Towards Software Defined ICN based Edge-Cloud Services

Ravishankar Ravindran‡, Xuan Liu†‡[0], Asit Chakraborti‡, Xinwen Zhang‡, Guoqiang Wang‡
‡Huawei Research Center, Santa Clara, CA, USA. {ravi.ravindran,asit.chakraborti,gq.wang}@huawei.com
{xinwenzhang}@gmail.com
†University of Missouri-Kansas City, MO, USA. xuan.liu@mail.umkc.edu

***Abstract*—ICN deployment will be based on the grounds of saving CAPEX/OPEX and/or enabling new services. This paper makes a case for the latter leveraging, emerging technologies such as network function virtualization (NFV) and software defined networking (SDN). We propose a framework to enable ICN based service platform as virtualized network functions to enable several edge-cloud services such as enterprise applications, big data analytic, or M2M/IoT services. This platform is generic to support several ICN protocols and corresponding real-time and non-real time services leveraging ICN features such as name based routing, caching, multicasting, and flexible security techniques. As an implementation of this architecture, we discuss how a scalable network based conferencing solution can be realized over the proposed ICN platform and compare it with a peer-to-peer design through a performance analysis.**

I. Introduction

Information-centric architectures surveyed in [9] are a paradigm shift in terms of how applications interact with the network. Information-centric networking (ICN) de-couples applications from the transport layer by first naming entities such as applications, services, and content and then binding consumers to them through ICN's name resolution layer. This de-coupling also assists with dynamic features like mobility, migration, and replication of named objects such as devices, content,or services. Recent literature questions the motivation for ICN [3], and suggests incremental HTTP to achieve ICN features. The comparison is not justified as ICN is envisioned as a future network protocol to work in both infrastructure and ad hoc situations to ahdnle extreme levels of dynamism, and to serve applications ranging from content distribution to IoT applications such as local device-to-device or V2V applications. ICN enables these through features such as topology independent name based routing, multicast/anycast, in-network caching, mobility support, and packet level security. In contrast, HTTP was primarily designed as an application layer protocol over a reliable host-to-host connection-oriented logic. However, incremental design allows backward compatibility, and de-risk deployment from a CAPEX and OPEX perspective. Hence, one of the ICN challenges is to realize ICN deployment in current networks in a manner which introduces the technology in a non-disruptive manner and allow experimentation to understand its usefulness.

[0]This work is done during Xuan Liu's internship at Huawei Research Center, Santa Clara, CA, USA

Two industry trends, network function virtualization (NFV) and software defined networks (SDN), enable agile infrastructure and resource management for operators. NFV gained attention with the operators demanding [2] a technology framework to realize network functions in software over commodity hardware achieving economy of scale offered by cloud computing technology. SDN allows for independent evolution of the control and forwarding plane. This enables a programmable forwarding plane capable of being tuned to meet the policies of several co-existing applications.

NFV and SDN can enable ICN based services in future networks. Towards this, we propose an incremental and generic ICN based platform that supports ICN applications as edge-cloud services ranging from enterprise applications, content distribution, and M2M/IoT services. The platform leverages NFV constructs that allow the ICN framework to be realized as a *virtual network functions* (VNF)[1]. The platform enables service virutalization through open-APIs to service owners to manage services (provision, scale, or migrate), while service flows are engineered by the operator or service owner through an SDN framework. Further, the platform enables a service-centric user-to-network interface (UNI) that allows rich service level ICN interaction between consumers and the ICN platform. The service layer extends ICN's primitives to express service requirements such as service discovery, service management, and context adaptation functions. The proposal is evolutionary by realizing it as an overlay over an IP infrastructure making it transparent to existing transport infrastructure. As a realization of the proposal, we discuss a network based conference solution for an enterprise that is aimed for interactive real-time multimedia applications capable of scaling to a large number of participants.

Remaining sections are laid out as follows: Section II discusses NFV and SDN briefly, and how ICN benefits. Section III introduces the ICN based edge-cloud service framework. Section IV discusses a network based conferencing solution as a service over the proposed ICN platform. Section V discusses results comparing the proposed network

[1]VNF definition from ETSI NFV-ISG: A Virtual Network Function is a virtual functional block (e.g. firewall) which provides exactly the same functional behavior and interfaces as the equivalent non-virtual network function (appliance based solution today), but is deployed as pure software inside the network functions virtualization infrastructure (NFVI, is the collection of all compute, storage, bandwidth resource within an NFV scope).

based conferencing solution to a peer-to-peer based approach from the literature, and Section VI presents our conclusion.

II. NFV and SDN

The objectives and features of NFV and SDN and its economic incentives for network operators have been discussed in [5] and [7], we briefly discuss them here.

The key features and challenges of NFV are:

- Cost efficient realization of network functions in software deployed over commodity hardware. Ideally, this could be any function from L2-L7, and network functions can be connected arbitrarily to meet service specific objectives.
- Leverage cloud computing framework to achieve service virtualization allowing dynamic scaling of computing, bandwidth, and storage resources through open-APIs.
- Challenges in this emerging area range from realizing software functions as reliable, secure, and with the same level of performance as hardware based realizations, to complex live migration and scale network functions on-demand across multiple domains. Further, not only do software functions require seamless integration with existing network management infrastructure, but also, new management extensions are required to monitor the virtualized infrastructure.

The key features and challenges of SDN are:

- The separation of control and forwarding planes that enable fine grained service control and adaptation to varying states of the forwarding plane or states of a service.
- Standardized APIs connecting services and controllers (called north-bound APIs), and controller and forwarding elements (called south-bound APIs) that allows independent evolution of the service, control, and forwarding planes.
- Challenges span several areas of SDN technology, primarily related to topics on controller management such as placement, fault tolerance, consistency of networks state and forwarding policies; and scalability of forwarding elements to handle flows from various work loads arising from deployment scenarios such as in an enterprise, data center, or provider environments.

A. Relating to ICN

Our visualization of the relationship between NFV, SDN, and ICN is shown in Fig. 1. The top of the hour glass is the service virtualization layer enabled by NFV, as it allows services to be provisioned, scaled, and migrated on-demand. In the context of ICN, these are ICN protocol and service instances. The bottom of the hour glass is the SDN, which allows ICN flows between consumers and ICN services to be subjected to service policies, which also includes context adaptation. As ICN naturally leverages computing and storage resources, these are as important of resources as the bandwidth typically associated with SDN today. The waist is the ICN

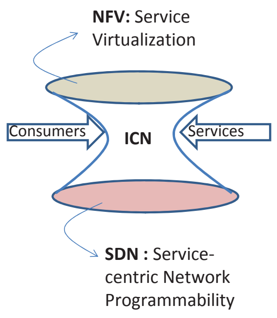

NFV: Service Virtualization
Consumers
ICN
Services
SDN : Service-centric Network Programmability

Fig. 1. ICN as A Service-Centric Narrow Waist

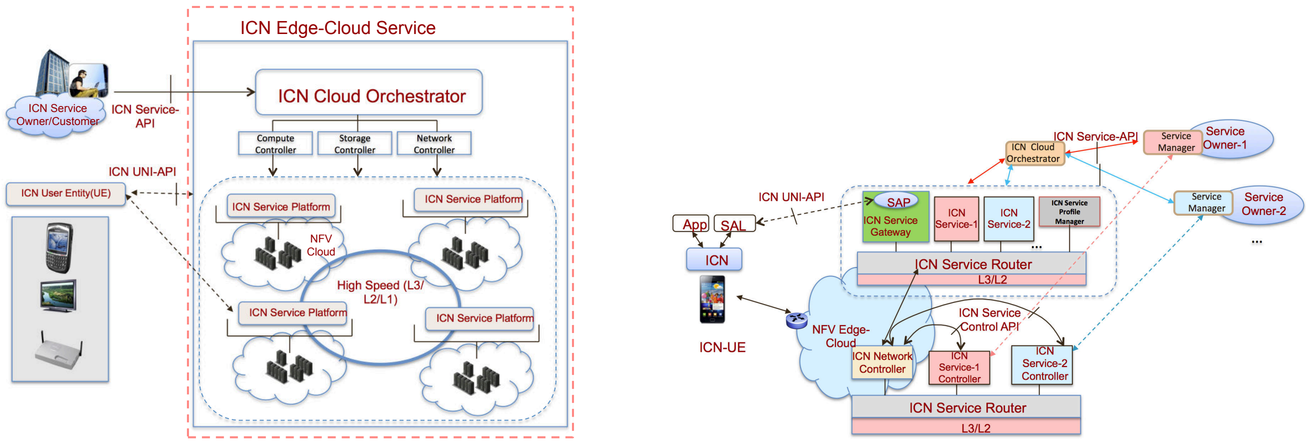
that connects consumers to services through expressive ICN-APIs that not only includes primitive *Get()* or *Put()* primitives[2] but extended to express contextual changes to allow real-time adaption of services to the consumer's context(s).

III. ICN based Edge-Cloud Framework

Experience from IPv6 deployment suggests operator's reluctance to deploy new technology unless it provides novel service opportunities or significantly reduces CAPEX and OPEX. This holds true for ICN as well; however, ICN distinguishes itself with a rich set of features to realize several new applications and services that are difficult to realize over IP today. Some of these include context-rich interest expression and in-network processing, large scale mobility through late-binding features, and equal support for both ad hoc and infrastructure based applications through de-coupling of the application and the transport layers. However, the challenge of introducing ICN in operator's domain still exists. Fortunately, the push for virtualized platforms based on NFV framework shall provide a low barrier entry and enable experimentation of ICN protocols and services as software functions, which by design can co-exist with other services. Particularly, ICN can as a serve glue to enable cooperation between operators and application service providers(ASP) such as NetFlix, Amazon etc, to deliver services from the network edge. This idea has also been explored as an overlaid service delivery solution called OpenADN framework [6]. Edge services allow service delivery with least latency while achieving high degree of service contextualization. Following are high level objectives of the proposed ICN based edge-cloud framework:

- A platform to enable different ICN protocols and related services both for experimentation and future deployment. The platform is aimed for information-centric applications such as conferencing or IoT applications that can leverage name based content dissemination, in-network caching, receiver-oriented interest and data multicasting, content level integrity, privacy, and provenance.
- A platform with open-APIs to stake holder entities such as ICN service owners (i.e in case of hosted services),ICN service controllers (operator's or third party controlled) for fine grained policy enforcement, and ICN consumers

[2]In the paper get/put, interest/response are used interchangeably implying ICN's receiver-oriented communication model.



(a) ICN as NFV based edge-cloud service.

(b) High level view ICN service platform.

Fig. 2. ICN based edge-cloud service framework.

(i.e. consumers of a particular ICN service, e.g., IoT applications, conferencing, V2V etc.).

- Adaptation to user dynamism through adaptation to changing context and achieving service level dynamism such as to scale, migrate, and replicate service resources on-demand.

A high level view of our framework is shown in Fig. 2(a). Future network based on NFV shall be an inter-connection of edge-clouds controlled by one or more providers, with intra- and inter-domain service level inter-connection to manage computing, storage, and bandwidth resources shared among heterogenous services. These edge-cloud realizations are expected to be deployed in the vicinity of the provider's central office (CO) or at points-of-presence (PoP), or as network-edge data centers, enabling local instantiation of ICN services to consumers, while providing global service delivery through a unified service control infrastructure. In the spirit of incremental deployment, the proposed ICN platform is realized as ICN router with ICN services local to the ICN router overlaid over current IP network, with multiple instances of the platform within a edge-network scope. All the ICN platform components can be realized as virtual network functions; however for performance reasons the ICN router itself could be non-virtual, while ICN services can be provisioned or scaled on-demand. We envision the platform to be comprised of three important components: *ICN cloud orchestrator*, *ICN service platform*, and *ICN service layer*. A more detailed view of this is shown in Fig. 2(b).

ICN Cloud Orchestrator: The cloud orchestrator interfaces with the ICN service owners to program the cloud resource through the *ICN service-API*. The service-API allows service owners to express ICN specific service requirements during provisioning and request dynamic changes to computing/storage/bandwidth resources to adapt to service load conditions. The ICN cloud orchestrator converts service requirements to compute storage, connectivity, and bandwidth requirements to meet service objectives. This API could also give feedback of service usage statistics to service owners to scale service requirements benefiting through a pay-as-you-go strategy. Further, service customers (as in an enterprise) uses the service-API to provision required resources as in a software-as-a-service (SAAS) set up, in which case a slice of the ICN service resource is dedicated for the customer's use.

ICN Service Platform: The ICN service platform is composed of VNF instances in the form of ICN protocol and service control functions. The protocol implementation, *ICN service router*, implements an ICN forwarding plane which conducts name based routing inter-connecting ICN service instances which can be distributed among several edge-cloud instances. The service platform's control plane functional components includes: *ICN service gateway* (ICN-SGW), interfaces with ICN consumers through the ICN UNI-API to resolve service requests to ICN service instance(s). The UNI-API also helps with service discovery and service context adaptation of the UE. The *ICN service profile manager* (ICN-SPM) is a database to resolve ICN service-ID to the location(s) of the *ICN service* instances. ICN services are instances of applications such as content distribution, conferencing, or IoT services, which executes its own service logic through interaction with several other service instances.

Another important control plane component is the realization of a SDN framework through the *ICN service controller* instantiated per-service to allow fine grained control of consumer interests and data response through name resolution service and context adaptation logic. The service controllers manages the name based routing policies of the ICN service router. Depending on the specific ICN protocol, the interest resolution to service instance or content can be online and per-hop or controlled in a centralized manner through a name resolution system (NRS). The service controller also handles context adaptation events of UE's expressed through the UNI-API. The service controllers interfaces with the *ICN network controller* through the *ICN service control API*. This API obtains service specific ICN topology view, statistics, and events to execute service functions.

ICN Service Layer: The service layer implements service functions to allow for the consumer's application to interact with the ICN service platform through the ICN UNI-API. A high level view of the service layer is shown in Fig. 3. The end host *service access layer* (SAL) handles service management functions such as service discovery, resolution, publish, and context management. In addition, ICN transport layer functions for reliability and flow control can be implemented in SAL; however applications could limit interaction with SAL for service bootstrapping and management functions. At the network end, the ICN service gateway contains the *service access point* (SAP) that interprets user requests and invokes service management tasks which includes discovery, resolution, publishing, or handling context changes. These tasks can be handled locally or with the help of other SDN control components and the SPM. ICN service instances can also leverage ICN-SGW's service management functions to enable features like service orchestration[3]. This interaction could also be used by ICN services to discover other services or request more resources on-demand to meet its service requirements without involving its own service control components or the service manager.

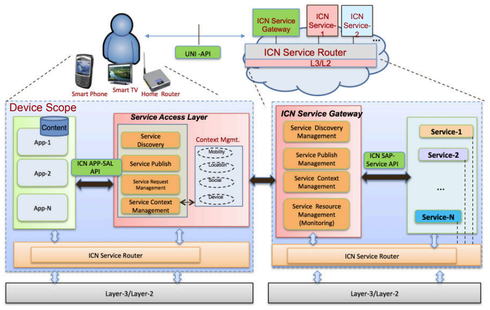


Fig. 3. ICN edge-cloud UNI service Layer

As discussed, the ICN service platform exploits cloud computing features of NFV to realize an ICN platform which can be generalized to instantiate several services corresponding to any ICN protocol. The services could be real-time or non-real time in nature.

We next discuss a solution to realize network based conferencing service for enterprises with the objective of scaling to many participants. We present this as a conferencing framework which any multimedia conferencing application can leverage.

IV. ICN Network based Conferencing Framework

A. *High level design*

This section discusses a proposal to realize conferencing as an ICN edge-cloud service because of its information-centric characteristics such as: generated content is in the context of the conference and shareable among participants and independent of the participant's device; data multicast

[3]Service orchestration is the concatenation of multiple services realized as a service graph to satisfy a user request.

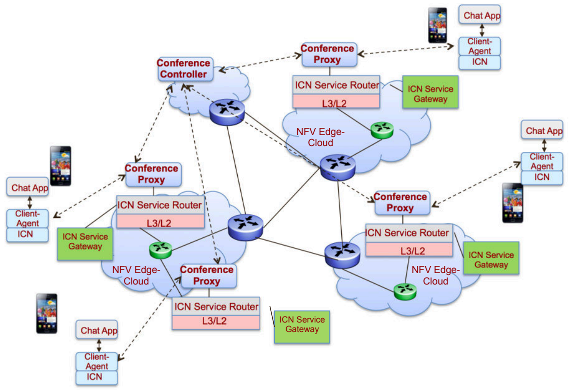


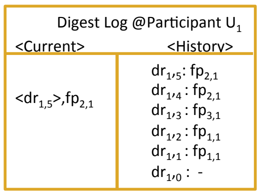
Fig. 4. ICN conference service

is natural as multiple participants within a site can share a single copy of data; and desirable content level security for integrity and provenance. Instantiation of such a service begins with an enterprise customer submitting a request to the ICN cloud orchestrator with information such as participant sites, maximum number of participants per site, service description parameters such as bandwidth requirements (determined by device, codec, etc.). The orchestrator initializes the ICN service VNFs, determines cache/storage, connectivity, and bandwidth requirements to sustain the QoE of the conference. Without delving into details of the service orchestration, we present design of our conferencing framework focussing on its scalability compared to the peer-to-peer design model called Chronos[10].

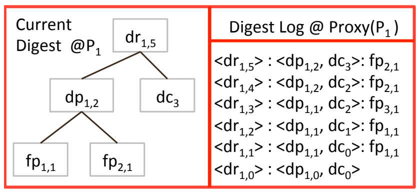
Chronos [10] proposes an NDN based peer-to-peer data synchronization solution. One of the scalability challenges with this design is the universal peer-level synchronization of the root digest (which represents the data set view) among all the participants which is inefficient to scale with an increasing number of participants or the rate of updates from the participants. Lack of scalability affects the QoE of interactive multimedia applications. In our design, scalable synchronization of the conference state is achieved through a two level hierarchy design. Through a network based conference system, shown in Fig. 4, new updates from participants are *PUSHED*, versus being *PULLED*, by each participant. Scalability is achieved using conference specific network functions, as local updates only need to be sent to the ICN conference service proxy instance (discussed later). Further the design also enables efficient recovery, as the state of the conversation is in the network, rather than at the end points

The functional components of this solution are shown in Fig. 5, and it includes three important components: client-agent, conference proxy, and the conference controller.

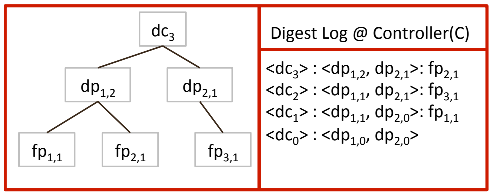
Client-Agent: Client-agent is local to the user entity (UE). Applications depend on the client-agent to obtain updates of the remote participants as well as to notify any local updates. Local updates trigger notification to the serving conference proxy (discussed next). The client-agent attaches to a conference proxy after it bootstraps; the discovery of the conference



(a) Digest log in client-agent.


Current Digest @P_1
$dr_{1,5}$
$dp_{1,2}$
dc_3
$fp_{1,1}$
$fp_{2,1}$
Digest Log @ Proxy(P_1)
$<dr_{1,5}> : <dp_{1,2}, dc_3>: fp_{2,1}$
$<dr_{1,4}> : <dp_{1,2}, dc_2>: fp_{2,1}$
$<dr_{1,3}> : <dp_{1,1}, dc_2>: fp_{3,1}$
$<dr_{1,2}> : <dp_{1,1}, dc_1>: fp_{1,1}$
$<dr_{1,1}> : <dp_{1,1}, dc_0>: fp_{1,1}$
$<dr_{1,0}> : <dp_{1,0}, dc_0>$

(b) Digest log in conference proxy.


dc_3
$dp_{1,2}$
$dp_{2,1}$
$fp_{1,1}$
$fp_{2,1}$
$fp_{3,1}$
Digest Log @ Controller(C)
$<dc_3> : <dp_{1,2}, dp_{2,1}>: fp_{2,1}$
$<dc_2> : <dp_{1,1}, dp_{2,1}>: fp_{3,1}$
$<dc_1> : <dp_{1,1}, dp_{2,0}>: fp_{1,1}$
$<dc_0> : <dp_{1,0}, dp_{2,0}>$

(c) Digest log in conference controller.

Fig. 7. Digest log in client-agent, proxy, and controller.

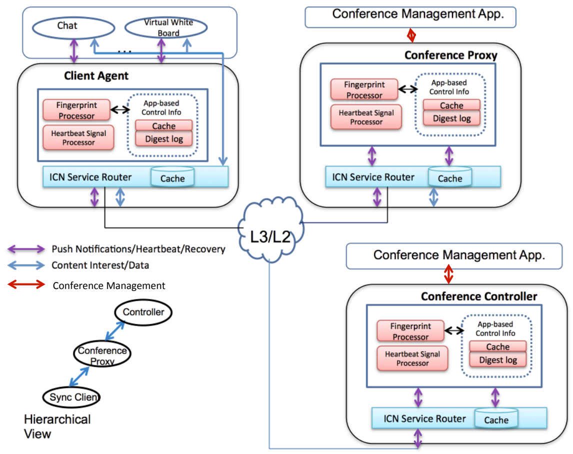
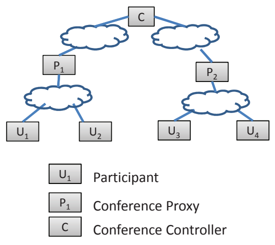
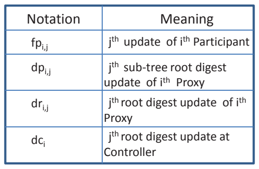


Fig. 5. ICN conference functional modules



(a) ICN conference example.



Notation	Meaning
$fp_{i,j}$	jth update of ith Participant
$dp_{i,j}$	jth sub-tree root digest update of ith Proxy
$dr_{i,j}$	jth root digest update of ith Proxy
dc_i	jth root digest update at Controller

(b) Digest log notation and meaning.

Fig. 6. ICN conference example.

proxy is through the SAL-SAP interaction. The functional components of a client-agent are shown in Fig. 5. The *fingerprint processor* handles the local application's new updates and also remote updates received from the conference proxy. A fingerprint is a concatenation of conference attributes such as conference name, participant-ID and sequence-ID (similar to [10]), and a digest of these attributes. The client-agent manages a *cache* and a *digest log* unique to each application it serves. The cache stores the recent fingerprint updates from the application, while the digest log is a temporal sequence of events reflecting both local and global updates. The client-agent receives digest updates from the conference proxy. The *heartbeat processor* lets the proxy know about its liveness, by periodically sending a heartbeat; in certain scenarios, the heartbeats could also piggy back on the notifications. For the simple topology shown in Fig. 6(a), a snapshot of the current digest and the digest log at the client-agent for U1 is shown in Fig. 7(a); here the current digest state is the latest digest received from the conference proxy, while the digest log keeps track of both the digest and the fingerprints.

Conference Proxy: This is realized as the virtual network functions as part of the ICN conferencing service. The conference proxy service handles the notification traffic of the conference, while the content traffic switches through the ICN service platform. The control traffic corresponds to the notification that the proxy handles from the local client-agents as well as the notification from the upstream controller (discussed next). The functional components of the conference proxy is similar to the client-agent and is shown in Fig. 5; the difference is in the scope of the digest tree and log it maintains. As shown in Fig. 7(b), the proxy manages a two-level tree. At any given point of time, the top level dr_{ij} is the root level digest that summarizes notifications from local client-agents and the controller. This root digest is periodically send to the client-agents, which identifies the global status of the conference. The second level tree records a second level digest, dp_{ij}, and this summarizes the notification from all the participants. This digest is notified to the local client-agent and controller, whenever an update is made to the digest tree.

Conference Controller: One possible design for the network based conferencing solution is to have one level of a hierarchy of conference proxys, and follow the design in [10] to synchronize the digest trees. This approach shall suffer from the same inefficiencies of the Chronos design, and hence, we chose another level of virtual network function to handle notification from the conference proxy that logically conducts similar operation as the conference proxy in terms of managing the current digest and the digest log. Referring to Fig. 7(c), the controller manages a single level tree for all the conference proxys it is serving. Here dc_j identifies the j^{th} update in series of events. Whenever a participant notifies a new update to the client-agent, this notification is sent to the proxy that updates its own digest tree resulting in an upstream notification to the controller; the controller then updates its tree and notifies the other remote proxies of the new update.

B. Design Considerations

Conference Name Space: This we discuss using the convention based on CCN [4]. There are two name spaces of interest: first, the one to exchange control information between the client-agent and the conference proxy and between the conference proxy and the controller. Second, is the name space for the applications to express interest for the conference content. A control name space is point-to-point between client-agent and proxy and proxy and the controller, and includes actions such as join/leave, fingerprint notifications, heartbeat, and recovery actions. A generic name space for this is $/entity\text{-}id/chat\text{-}room/\langle action - type\rangle/\langle parameter\rangle$; here $/entity\text{-}id$ is either the ID of the client-agent, proxy or the controller. The $parameters$ also vary based on the $action\text{-}type$ such as update notification, heartbeat, or recovery. For content exchange, the requirement here is global routing. In this case, for scalability and considering client-agents are clustered as part of the serving proxy, and proxies themselves are aggregated under a provider's ID, one possible name space is $/provider\text{-}id/proxy\text{-}id/user\text{-}id/chat\text{-}room/\langle sequence - id\rangle$. The $proxy\text{-}id$ in the name ensures data multicasting from the conference proxy instances. To summarize, the name space is governed by routing requirements and hence, has to be realized in a given situation.

Control Overhead: One of the objectives of network-based conferencing system is to reduce the control overhead. In our approach, each participant only synchronizes with a dedicated conference proxy via the client-agent, and the state of the conference proxies are synchronized with the conference controller. This way, broadcast based sync mechanisms (used in a peer-to-peer case) can be avoided. We can quantify the control overhead due to this mechanism by characterizing the update complexity of the digest tree at the conference proxy.

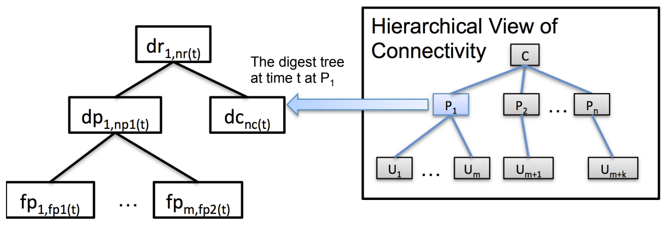


Fig. 8. State information showed by digest tree.

$$np_1(t) = \sum_{i=1}^{m} fp_i(t)$$

$$nr(t) = nc(t) + np_1(t)$$

$$nc(t) = \underbrace{np_1(t)}_{\text{Local update state}} + \underbrace{\sum_{j=2}^{n} np_j(t)}_{\text{Remote update state}}$$

Fig. 9. Characterizing the control overhead at conference proxy.

Fig. 8 shows a general form for the digest tree at proxy P1 at a given time t, which serves m local participants, and the most recent fingerprint for participant i is represented by $fp_i(t)$. The $np_j(t)$ represents the current number of local notifications received by the proxy j. $nr(t)$ indicates the remote notifications the proxy received from the controller, and $nc(t)$ shows the total number of notification received by the proxy from both local participants and remote participants. This relationship is formulated in Fig. 9. This formulation indicates the update load handled by a proxy node that increases linearly with the number of proxies. This is in contrast to the peer-to-peer case where the total the control messages in the network is $O(n^2)$, where n is the number of participants in the network.

Recovery from failures: Unlike Chronos, the recovery can be more efficient due to trusted anchor points in the network provisioned to manage the conference. Two types of recovery have to be addressed, one due to temporary network disruption such as a congestion scenario and the other due to long term network disruption, such as node or link failures. Both of these can be addressed by piggy backing sufficient state information in the notifications that can be used to enter a recovery mode due to lost or delayed notifications. One design enhancement during a normal operation is to piggy back the previous digest state along with the current digest in the notifications. For example, a notification from the conference proxy to client agent could look like $/client\text{-}agent\text{-}id/chat\text{-}room/notification/\langle prev\text{-}digest\rangle/\langle curr\text{-}digest\rangle$. In the normal state, the current root digest at the client-agent should match the $\langle prev\text{-}digest\rangle$; if this is not the case the client agent can try to recover the lost notification by entering into a recovery state with the proxy. Similar logic can be applied to interaction between the proxy and controller, and client-agent and proxy too.

Security Considerations: Considering information that is transacted during group communication is generally private, security considerations span both the control and forwarding levels. An appropriate authentication procedure is required to admit a conference participant. Conference management related control plane transaction (related to sharing control and forwarding plane name spaces) should be secure as well; this can be handled by the network based conference management entity hosted at the conference proxy, controller or by other third party. Security considerations between content producer and consumer have been discussed in Chronos. In addition provenance and integrity have to be maintained for transaction between client-agent and proxy, and, proxy and controller for which ICN based techniques [8] can be applied.

V. Simulation Evaluation

In this section, we evaluate the scalability of our proposed conferencing system in terms of the number of participants and frequency of simultaneous updates generated by the participants. We conduct the simulation using ndnSIM[1] based on CCN protocol. Fig. 11 shows the basic setup of the simulation. As per the conference design discussed in Section IV, we modeled a user chatroom application and as part of the conference framework, this included the client-agent, the proxy, and the controller. In particular, the chat application aims to simulate participants for a conferencing application. We also present a comparison with the name based peer-to-peer conferencing model from [10] by studying an NDN network with four routers and three participants (Fig. 10). The objective with the peer-to-peer case is to show the challenge in synchronizing

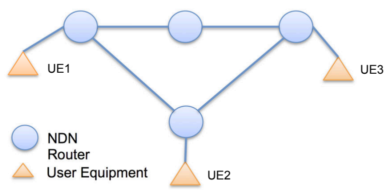


Fig. 10. Flat peer-to-peer scheme

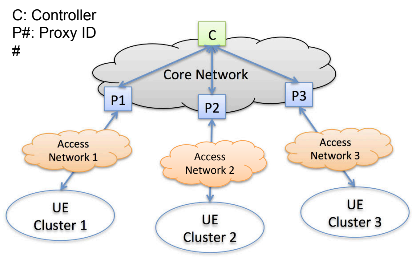


Fig. 11. Simulation set up

the conference state among the participants even in a small topology setup.

A. *Network based Conferencing*

In real-time interactive services, the timeliness of content arrival is critical for a good QoE. Furthermore, in a group setting, all users should converge to the same view with minimum delay. Convergence time here is defined as the time difference between the time instant at which the content is generated and when the final participant receives it. We conducted the simulation by varying two different parameters: frequency of content generation, number of participants, and network conditions. The simulation setup for the three scenarios is shown in Table I. The content generation rate follows Poisson distribution for all the scenarios.

Our simulation on the network based conferencing model includes two segments of the transport network: the local access network and the core network. For the local network we used a two-level tree topology, and the participants within the same local network are randomly connected to the routers in the local access network; whereas, for the core network, we used two types of topologies: a 3-by-3 grid network (Topo-1) and an Abilene network (Topo-2).

1) Content Generation Frequency: In Case I, we study the efficiency of network based conferencing by evaluating its convergence time with a fixed update rate of 0.5 contents/sec. Fig. 12(a) and Fig. 12(b) show the convergence time among all the participants for a particular new content generated by a remote participant. For this, we choose one participant from each cluster (participant ID = 1, 78, and 135) that generates new contents, and we trace the time when all other participants receive this content. We observe that the the participants within the same cluster as the sender receives the new content earlier than those in the remote clusters. This is because, for local participants the proxy immediately pushes the notification, while the same is received after the notification traverses the controller and the remote proxy for the remote participants.

TABLE I
SIMULATION SCENARIOS

Case	Participant Size	Topology Type	Link Capacity Gbps	Rate (λ) (content/sec)
I	150	Topo-2	0.1	[0.5,1,2,5,10]
II(a/b)	150	Topo-1 Topo-2	[0.1,0.2,0.5,1.5,5,10]	0.5
III	[60,150,300]	Topo-2	1	[0.5,1,2,5,10]

We also observe that the convergence time among the remote cluster participants did not vary much; the difference here is due to the propagation delay in the respective topologies.

Next we study convergence time performance due to multiple and almost simultaneous content updates. In this situation, we randomly pick one participant (ID = 15) and trace its contents generation during the first 2.5 seconds of the simulation time and record the time when everyone else receives these new contents; this is shown in Fig. 12(c). The “star” in the graph indicates the content generation time. We observe that between 2.3 sec and 2.5 sec seven new contents are generated and the time to notify and receive new content only depends on the topology, similar to the case above.

The convergence time is also affected by network conditions, particularly the link capacity, this is represented as Case II(a/b). For this, we fix the rate of the new content generation at 0.5 contents/sec and vary the link capacity in the network from 0.1Gbps to 5Gbps. For the given load the convergence time stabilizes when link capacity exceeds 1Gpbs. However, as expected, at a lower link capacity (0.1Gbps) the queueing delay causes an increase in the convergence time.

Next we study the convergence under varying loads by increasing the frequency of the content generation. Here we vary the arrival rate λ from 0.5 to 10 contents/sec. To avoid the link capacity bottleneck, we first set the link capacity as 10Gbps. Overall, we observe that the fast content generation has less impact on the convergence time. This is because the push model design allows new update notifications almost at the same rate of the new content generation. To study the case where the link capacity is the bottleneck, we study by setting the link capacity to 0.1 Gbps for the Abilene topology, and vary the content rate. We observe in Fig. 14 that under constrained link capacity, the queueing delay in the network increases causing overall an increase in the convergence time.

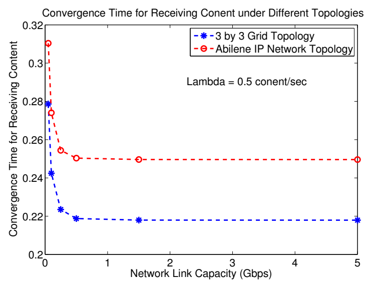
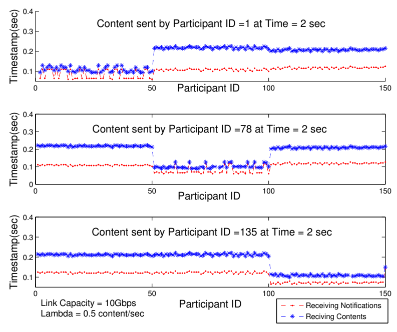
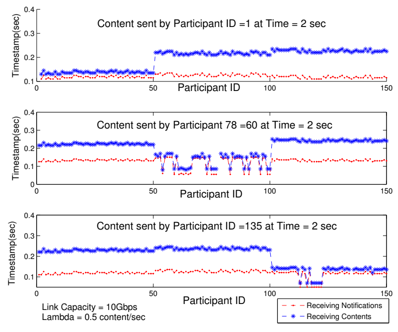


Fig. 13. Performance under various link capacity.


(a) Topo-1: 3-by-3 grid topology.


(b) Topo 2: Abilene topology.

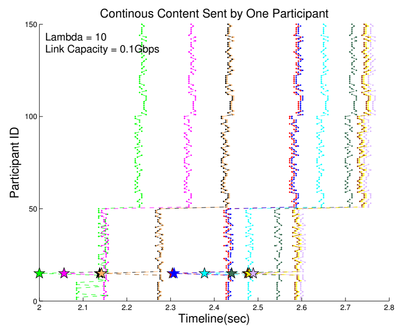

(c) Multiple updates v.s. Convergence performance

Fig. 12. Single update v.s. Multiple updates

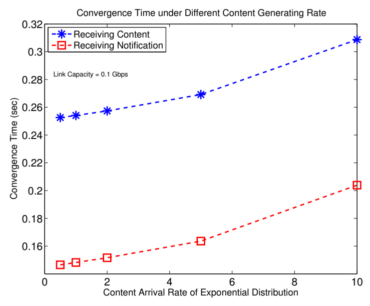


Fig. 14. Convergence time under different content generating rate.

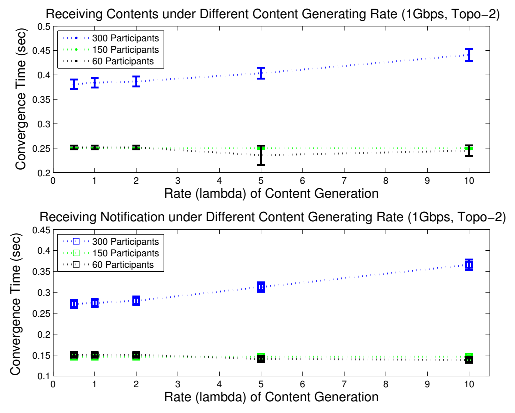


Fig. 15. Evaluation of scalability.

2) Cluster Size: In this scenario (Case III), we evaluate the convergence performance by varying the cluster size. We consider 20, 50, and 100 participants, which aggregates to 60, 150, and 300 participants. Fig. 15 shows the 95% confidence interval for the convergence time; here, the link capacity is set to 1Gbps. For each setting of the number of participants, we vary the content generating rate. We observe that, under sufficient link capacity and for the 60 and 150 case, the convergence time did not show significant difference. But, when we increase the number of participants to 300, convergence time increases, this is because of the high link loads on the access links that results in high queueing delay. Overall, we observe that the push model scales to a large number of participants.

B. Comparison with Peer-to-Peer Scheme

To compare the proposed hierarchical network-based conference system, we simulated the topology shown in Fig. 10, where link delay and capacity are set to 10ms and 1Mbps respectively. In this scheme, there were no proxy nodes or controller to handle new content notifications. Every participant in the network periodically broadcasts sync interests [10] to pull a new update. Following is the evaluation of this scenario.

1) Frequency of Sync Interest: In the peer-to-peer scheme, unlike the network based approach, the generation of sync interests is regulated by the consumers, which is independent of the content generation process. So, first we evaluate the impact of this effect by varying the ratio of the frequency of sync interest to the frequency of the content generated by fixing the latter. Fig. 16 shows the percentage of total contents received by all participants during a 10 second time window. A randomly chosen participant generates new content every one second (2 content/sec), while the sync interest rate varied as a multiple of content generation frequency as from 1 to 10. We observe that when the ratio is set to its lowest setting of 1 (i.e. same as content generation rate), only half of the new content generated in the network is received by the remote participants; this is because every sync interest that is issued by a remote consumer represents its current status, and if there are multiple simultaneous updates the consumer continues to fall behind as time advances. Thus, the only way to catch up with the content generation rate is to request at much higher rates than the content generation rate. Hence, we see when the ratio is high, consumers get in sync with the producer but also contributes to a high control overhead.

2) Synchronization Among Participants: The previous section discussed the impact of sync interest frequency on convergence time. The success of a peer-to-peer based design depends on designing an efficient synchronization mechanism among participants. This issue is discussed in [10] and is remedied through an exclude filter or timeout mechanisms. Such mechanisms further compound the complexity of the

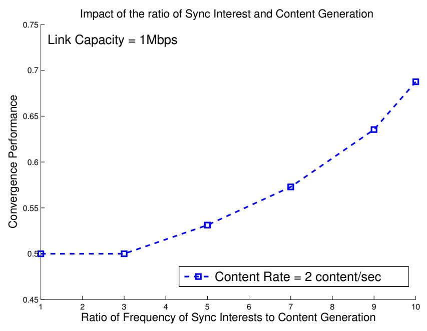


Fig. 16. Convergence performance for peer-to-peer scheme.

implementation as more configurations and good judgement is required at the consumer's end to synchronize the conference state to operate state with good QoE.

VI. Conclusion

This paper proposes an ICN based edge-cloud service framework leveraging NFV and SDN technologies. Ideally, this operator owned platform enables realization of any ICN protocol and services through open-APIs connecting consumers, service customers, and service owners. The paper discusses the functional components of the ICN service platform controlled by an operator in an overlay deployment in an NFV realization to orchestrate complex service realizations. As an example service, we discuss a network based conferencing solution which naturally benefits from ICN features. The performance of this service has been presented through simulation evaluation, and shown to scale in comparison to peer-to-peer solution. We are prototyping this framework and the conferencing solution as service over this platform.

References

[1] A. Afanasyev, I. Moiseenko, and L. Zhang. ndnSIM: NDN simulator for NS-3. Technical Report NDN-0005, NDN, October 2012.

[2] M. Chiosi, D. Clarke, J. Feger, C. Cui, J. Benitez, U. Michel, K. Ogaki, M. Fukui, D. Dilisle, I. Guardini, D. Lopez, F. Ruhl, P. Sen, and et al. Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. In *SDN and OpenFlow World Congress*, 2012.

[3] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: incrementally deployable ICN. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, SIGCOMM '13, pages 147–158, New York, NY, USA, 2013. ACM.

[4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT '09, pages 1–12, New York, NY, USA, 2009. ACM.

[5] A. Manzalini, R. Minerva, F. Callegati, W. Cerroni, and A. Campi. Clouds of virtual machines in edge networks. *Communications Magazine, IEEE*, 51(7):–, 2013.

[6] S. Paul and R. Jain. Openadn: Mobile apps on global clouds using openflow and software defined networking. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 719–723, 2012.

[7] S. Sezer, S. Scott-Hayward, P. Chouhan, B. Fraser, D. Lake, J. Finnegan, N. Viljoen, M. Miller, and N. Rao. Are we ready for SDN? Implementation challenges for software-defined networks. *Communications Magazine, IEEE*, 51(7):–, 2013.

[8] D. Smetters and V. Jacobson. Securing network content. Technical Report, PARC, 2009.

[9] G. Xylomenos, C. Ververidis, V. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. Katsaros, and G. Polyzos. A survey of information-centric networking research. volume PP, pages 1–26, 2013.

[10] Z. Zhu, C. Bian, A. Afanasyev, V. Jacobson, and L. Zhang. Chronos: Serverless multi-user chat over NDN. Technical Report NDN-0008, NDN, October 2012.