In-Network Compute Extensions for Rate-Adaptive Content Delivery in Mobile Networks*

Francesco Bronzino, Chao Han, Yang Chen†, Kiran Nagaraja, Xiaowei Yang†, Ivan Seskar and Dipankar Raychaudhuri
WINLAB, Rutgers University, North Brunswick, NJ 08902, USA
Email: {bronzino,chaohan,nkiran,seskar,ray}@winlab.rutgers.edu
† Department of Computer Science, Duke University, Durham, NC 27708, USA
Email: {ychen,xwy}@cs.duke.edu

***Abstract*—Traffic from mobile wireless networks has been growing at a fast pace in recent years and is expected to surpass wired traffic very soon. Service providers face significant challenges at such scales including providing seamless mobility, efficient data delivery, security, and provisioning capacity at the wireless edge. In the MobilityFirst project, we have been exploring clean slate enhancements to the network protocols that can inherently provide support for at-scale mobility and trustworthiness in the Internet. An extensible data plane using pluggable compute-layer services is a key component of this architecture. We believe these extensions can be used to implement in-network services to enhance mobile end-user experience by either off-loading work and/or traffic from mobile devices, or by enabling en-route service-adaptation through context-awareness (e.g., knowing contemporary access bandwidth). In this work we present details of the architectural support for in-network services within MobilityFirst, and propose protocol and service-API extensions to flexibly address these pluggable services from end-points. As a demonstrative example, we implement an in-network service that does rate adaptation when delivering video-streams to mobile devices that experience variable connection quality. We present details of our deployment and evaluation of the non-IP protocols along with compute-layer extensions on the GENI testbed, where we used a set of programmable nodes across 7 distributed sites to configure a MobilityFirst network with hosts, routers, and in-network compute services.**

***Keywords*—Internet architecture, mobility, in-network computing, cloud, video streaming, rate adaptation, video transcoding**

I. A Case For In-Network Compute

Mobile devices such as smartphones and tablets have overtaken personal computers to become the primary platform for accessing the Internet. Internet reports caution, however, that video constitutes a majority of the traffic and will increasingly dominate mobile data in the future [1]. With this growing trend, service providers face significant challenges in providing quality service while maintaining high network efficiency. Mobility further complicates the user-experience aspect due to variable link quality, temporary disconnections, and seamlessness when crossing physical networks. Service providers commonly adopt short-term strategies of limiting user network activity with expensive data plans and bandwidth throttling, and end up with outright poor or patchy service performance. While better spectrum management and capacity improvements promise to help, we argue that innovation in network protocols and in-network services can provide large improvements over current data delivery efficiency, particularly in mobile wireless networks.

Widely deployed middlebox solutions such as content proxies have helped since the early days of the Internet to opportunistically improve data delivery performance. In recent years, content delivery networks and cloud platforms boast further gains through strategic placement and geo-colocation of content and services with end-user. However, these solutions have often aimed to improve raw server-client RTTs, but do not address specific challenges of wireless networks or mobile devices in any meaningful way. CDNs and cloud applications are generally built as overlay services that can react to long-term differentiation in path qualities. While this may suffice for certain applications and when endhosts are on fixed wired networks, wireless networks aren't similarly benefited. A mobile device in a wireless network may have vastly different network experience in a short time span due to location, network load, or the particular access technology available(e.g., 3G, 4G or WiFi). For wireless networks, it is well understood that traditional content delivery methods, and in particular, connection-oriented transport protocols, have not provided satisfactory performance. We think that network-assisted approaches (e.g., storage and compute as integral components of the network) can provide a boost for content delivery in mobile wireless networks.

Streaming video, in particular, can benefit from an approach where the video bitrate can be dynamically adapted to match available bandwidth at the client access link using in-network compute resources. It is well known that the last-mile is usually the bottleneck link, and for wireless clients, the network performance could be highly variable under mobility. Solutions that address network performance variations today either buffer excessively at the client, or require client-side estimation of available bandwidth. While over buffering could be wasteful, accurate client-side estimations of available bandwidth is known to be problematic [2]. Furthermore, recent studies have shown that a multiplicity of rate-adaptive flows at access networks can lead to unexpected degradation and instability in delivery performance [3]. To address these challenges and to enable a host of novel value-add services (e.g., embedding of ads, localized alerts, regional subtitles, etc.), we argue for a network-assisted solution for rate-adaption. The placement of such an adaptation service close to the edge

*Research supported by NSF Future Internet Architecture (FIA) grants CNS-1040735, CNS-1345295, and CNS-1040043

/978-1-4799-6204-4/14$31.00

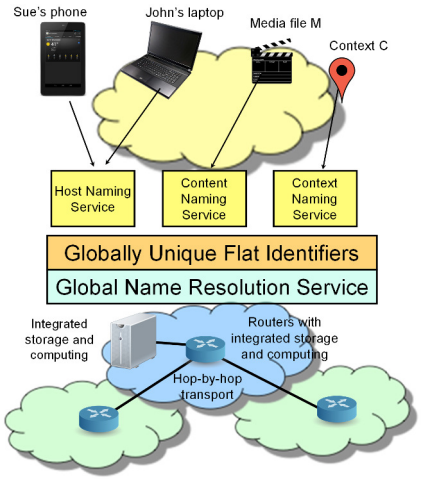

Fig. 1: MobilityFirst architecture.

would also enable the most accurate estimation of network performance as experienced by the client. As argued by others as well, we think the wireless service providers could make use of such in-network adaptation services as a way to effectively manage video flows at an aggregate level to handle traffic bursts, and to implement fairness, stability, and efficient delivery in the access network.

In this work, we leverage the compute-layer extensions to the data plane being explored within the MobilityFirst (MF) Future Internet Architecture (FIA) project [4], [5], [6] (overview in Section II-A) to implement in-network rate adaptation of streaming video delivery to mobile endpoints. We present details of the in-network computing architecture (Section III) and show through the rate-adaptation service example (Section IV) how in-network services can be deployed and addressed by endpoints. In Section V we describe the specifics of how we support in-network services in our MF prototypes and also implementation of the rate-adaptation service that we ran as part of the PacketCloud framework [6]. Finally, Section VI details our deployment of a prototype MF network over the GENI wide-area testbed and our experiences evaluating the video-adaptation service.

II. Background

A. MobilityFirst Name-based Network Architecture

MobilityFirst proposes a network architecture, protocol, and services with the principal goals of supporting seamless, at-scale *mobility*, and *trustworthiness* in the future Internet. Figure 1 shows the main components of the architecture: a fast and scalable global naming service (GNS), a hop-by-hop reliable data transport, and a routing fabric that is edge-aware and leverages in-network storage, all of which collectively target growing challenges at the mobile and wireless edge, but also the evolving computing landscape in the future Internet.

Name-based Network API: At the core of the architecture is the name-based networking abstraction that contrasts with the name-address conflated communication interface exported by Berkeley sockets and the TCP/IP stack. All network-attached objects in the new architecture enjoy direct addressability through long lasting unique network names or identifiers (we use GUIDs). This new GUID-centric network service API, first presented in [7] offers network primitives for basic messaging (*send, recv*) and content operations (*get and post*) while supporting several delivery modes innately supported by the MF network such as multihoming, multicast, anycast and DTN delivery. Combined with the GUID indirection and grouping (GUID mapped to one or more other GUIDs) concepts supported by the naming services, the new communication API can produce novel addressing and delivery capabilities only indirectly possible (and with certain in-efficiency) in today's IP architecture.

Support for at-scale mobility. Names for network objects are flat, self-certifying, Globally Unique IDentifiers (GUIDs) and apply to all network principals including hosts, services, content, and even abstract context - e.g., location). The protocol stack differs prominently from the current IP stack by including a GUID service layer which forms the new narrow waist and provides naming service by resolving GUIDs to their contemporaneous locators or network addresses. This resolution is enabled by a globally accessible distributed naming service (GNS), which is used by objects to both announce their latest location/address and lookup end points they wish to communicate with. We have explored and validated 2 alternate designs for the GNS: one an in-network service that closely integrates with the routing fabric[8] to minimize network latencies, and a second overlay implementation that optimizes through dynamic replica creation and placement of name services [9]. Both designs broadly meet our low resolution latency goals of less than 100ms on average for name lookup operations [8], [9]. Small resolution latencies are also crucial for dynamic or late-binding options in the network protocol, where en-route routers can (re)bind the latest network address of a device that is mobile.

Efficient data delivery to mobile networks. To address the poor performance of end-to-end transports in wireless conditions [10], [11], MF uses a segmented transport to progress data reliably hop-by-hop. Data is segmented into blocks that are cached at storage-enabled routers for in-network retransmission under losses. Experiments with this transport under a variety of wireless conditions have shown significantly improvements in fairness, throughput, latency and robustness [12], [13]. A new generalized intra-domain routing protocol is designed to flexibly find paths across wired, wireless and DTN segments [14], while accommodating mobile device disconnections with in-network storage. An edge-aware inter-domain routing that use smart propagation of capacity and load conditions at the edge of the network towards the core through *telescoping* updates, enables effective path discovery and informed forwarding decisions when delivery content to wireless networks, and to mobile and multihomed devices [15].

III. In-Network Compute Architecture

To enable future extensions to the network protocol, without expensive hardware replacements and disruption, MobilityFirst builds in an optional and dynamically pluggable *compute plane*. Examples of such need are additions of new service types, new principal types, new addressing structures, or extensions to the end-to-end security protocol. We envision service providers and network operators to be able to perform relatively simple upgrades in the form of software updates and addition/replacement of pluggable hardware modules to

extend the data plane functionality. Furthermore, we also postulate that such extensibility can enable third party application service providers (via the ISPs) to deploy either service end-points or service adaptors that are both closely integrated with the delivery path and best located to improve client experience.

A. Overview

Consider an end-to-end application service *S* that requires content delivery to mobile end-points. For simplicity, let us assume a server-client interaction model with a fixed server and a mobile client. Under good conditions, when the mobile client has good connectivity to the network core, the server responds to the client request by delivering the highest fidelity content (*C*). Under these conditions MF provides hop-by-hop transport under either unicast or multihomed delivery (e.g., when connected to both 4G and WiFi) as specified by the delivery service type during a send operation. Short-lived disconnections or minor variability in the access link can effectively be handled through a combination of in-network store-forward buffers in MobilityFirst routers (MFR) and contingency buffering at client ahead of actual consumption rate (e.g, video playback). If access conditions change significantly, however, and remain so for extended periods of time, it will become hard if not impossible to deliver the original high-fidelity content to the client in a timely manner. These conditions may occur under a burst of traffic or when the client moves to outer/poorer coverage area. In these cases, the compute plane extensions allow an authorized *adaptor* A_S service, regionally co-located with the client's network, to intercept and modify original data *C* to subsequently deliver *C'* that is commensurate to existing bandwidth constraints. A second demonstrative scenario for an in-transit adaptor service is for delivery content that is context-sensitive (C_X - for context *X*). Client mobility may redefine context (*X'*) thus requiring an adaptation to the delivered content($C_{X'}$).

B. Key Components

Enabling this dynamic adaptation requires compute services to be addressable, application providers to be able to deploy these services at appropriate points in the network, traffic to be correctly steered when beneficial, integrity of content and protocol continuity to be maintained post adaptation, and finally, applications should be provided a flexible service interface to invoke these services.

Service Addressing Generally, the compute layer will host services that are directly addressable by information contained in the data packet, i.e., the GUID of the service is explicitly specified in the packet as an extension header. However, service providers may also invoke specific compute services on packets based on header signatures alone, e.g., source-destination GUID or NA fields. These *transparent* services are similar to middlebox approaches today, which are routinely used to support application-layer proxies, content caches, traffic steering, and security functions. We allow for both directly addressable and transparent services (which may also have internally known GUIDs) in the MF compute plane.

Hosting Platform and Registration While routing fabric could potentially steer traffic to service endpoints located anywhere, packets will suffer least additional latency when the compute platform is co-located with the routing fabric. We envision both close hardware integrated solutions, where routers are embedded with a compute plane composed of general purpose CPUs, GPUs or pluggable accelerators , as well as co-located compute clusters or smart-sized clouds as possible platforms for hosting computer services. While the latter is less flexible and is challenging to scale, it presents the least overhead when the computations are limited and do not require access to other distributed state. Data compression or certain security checks are examples. When the compute service are more intensive, need to operate across a sequence of packets, or need access to distributed state, it appears that a co-located cloud is a better fit, as described in our PacketCloud instantiation [6]. In each case, MFR exports an interface for services to register a GUID addressable service with the forwarding plane. The attachment of the service instance will also be updated within the GNS to enable data packets that request the particular compute service to be routed correctly.

Geo-location and Routing We expect that one or more instances of an in-network compute service will selectively be deployed at locations seen as beneficial to the application provider, primarily due to expenses incurred in running on the platform provider. Since application services may be consumed by clients across the globe, it implies along certain paths these in-network adaptation services may be unavailable. Such clients are either served through end-to-end adaptation only, or alternatively, packets may be routed through paths that do have registered service instances and do not cause too much path stretch. This can be done in MF through native tunneling, where packets are first tunneled to the network hosting the service (known to application provider, and also registered in the GNS), and following adaptation are then routed to the destination client. In the non-tunneled default case, the packets are routed first to the target network of the client, where the compute plane service, if available, will act on the packet. With these, application providers can plan placement of their services according to benefits to their end-users, so long as ISPs in those geo-regions provide an open hosting platform.

IV. In-Network Rate Adaptation

In this paper we focus our attention on a specific use case providing details on how the service and content APIs could be jointly used to offer an in-network service that does rate adaptation when delivering video streams to mobile devices that experience variable connection quality. Bitrate adaptive streaming protocols have been widely adopted in most commercial solutions thanks to their flexibility in providing the desired service given variable network conditions. In particular, Dynamic Adaptive Streaming over HTTP [16] has been increasingly chosen as the standard go to solution thanks to its easiness of implementation, as it relies on the already existing HTTP infrastructure of webservers, proxies and caches. While it is easy to reckon the simplicity behind these protocols, they all rely on the ability of the client to estimate the available bandwidth, a task arguably very difficult under normal network conditions [2], and even more complex under wireless and mobile environments due to the high dynamicity caused by time-varying fading, shadowing interference and hand-off delays [17], [18]. Introducing an in-network service to provide the adaptiveness of the video stream behind these protocols, would allow to off load the task of deciding on the adaptive

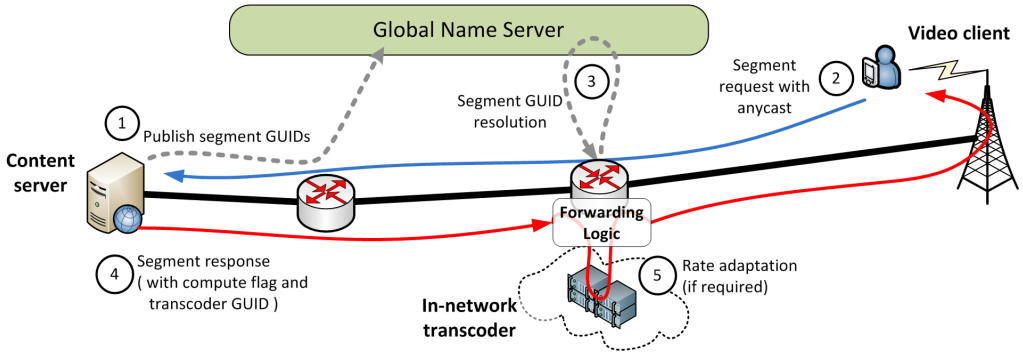

Fig. 2: Overview of rate-adaption use-case showing MobilityFirst's support for direct addressing of content and in-network compute services

stream to the components having the best knowledge about the available resources while maintaining most of the original good properties of these solutions; on the other hand, overly complicated services would discourage adoption.

To deploy an in-network service, the service developer, which for video streaming may either be the content distributor or the edge-network service provider, has to deploy, configure, and initialize the service instances at required network locations. To bring up an in-network service we refer the reader to the PacketCloud framework [6]. Once the service is up, the content server may pass further configuration such as listings of videos and available bitrates. Such metadata is transferred in the form of Media Presentation Description (MPD) files specified by the *MPEG-DASH* standard. To accomodate MF naming, we replace the content URLs in the MPD files with corresponding content GUIDs. Such config could be done out-of-band and apriori in preparation of handling the server-client flows. Though, the server may embed certain metadata, such as current bitrate, as parameters in the extension headers of the response chunk itself.

Leveraging and extending in-network functions. As previously introduced, MF offers native support for a variety of delivery services through the use of service type and service options. We have extended the basic set of name-based communication primitives to support the compute-layer service type and options, allowing application developers programmatic and in-band access to in-network services deployed on the compute plane. Figure 2 shows the flow of meta-data and data in a video streaming service implemented using the in-network extensions.

First step involves the server establish network presense for the content (and hence direct addressability). It does so by attaching the GUIDs for the video segments to the network:

mfattach*(sock, GUID-set)*

where GUIDs in *GUID-set* GUIDs will be associated with NA(s) corresponding to the host's one or more network attachment points, resulting in GNS mappings.Following this, the streaming proceeds as sequence of content requests and responses between client and server, starting with a request for video meta-data, i.e., the MPD file:

mfget*(sock, GUID, data_buf, size, svc_flag, opts)*

where GUID is a segment's GUID retrieved from the MPD file, and the svc_flag and opts encode any special delivery requests such as 'ANYCAST'. When the server receives a request (via *mfrecv*), it replies with the segment using MF's response API that matches the response with the corresponding *get*. To enable in-network rate-adaptation, the server requests the compute-layer delivery service, specifying the GUID of the transcoder service and the current encoded bitrate:

mfget_response*(sock, getId, data_buf, size, svc_flag, opts)*

where getId (returned during preceding recv) is used to match the *get* request, and svc_flag ('COMPUTING') and *opts* (in key-value pairs) encode details of the compute layer service to be invoked on the payload. The server adds the GUID of the compute service and the current encoded bitrate to the options parameter. Note that the response chunk may not always be steered to the compute service before reaching the client, and the decision will be based on the bandwidth available to the client device at the time of delivery. This is enabled through a routing-layer service implemented on the edge router.

V. Implementation

The in-network compute architecture consists of: a new network stack and socket API for hosts that implements the service interface used by the end hosts of the system, a software router that implements the hybrid GUID and NA based forwarding and storage-aware routing protocols, and a computing engine/platform that presents an open API for configuring and running in-network services.

Host Stack and API. The host stack has been implemented on Linux and Android platforms as a user-level process built as an event-based data pipeline. The stack is composed of a minimal end-to-end transport to provide message level reliability, the name-based network protocol including the GUID service layer, a reliable link data transport layer, and a policy-driven interface manager to handle multiple concurrent interfaces. The device-level policies allow user to manage how data is multiplexed across one or more active interfaces. The previously introduced socket API is available as a linkable library and implements the name-based service API which include the primitives *send, recv, and get* and a set of meta-operations available for instance to bind or *attach* a GUID to one or more NAs, configure transport parameters in the stack, or to request custom delivery service types such as multicast, anycast, multihoming, or in-network compute. Additionally to the general aspects just presented, the host stack, in coordination with the API interfaces, provides the functionalities to interact with the in-network service; we introduced in the

the previous sections different alternatives to provide tools for interacting with the network compute services; for this particular implementation, we exploited the option provided by the MF network protocol of flexibly introducing extension headers in the network header. The host stack is then in charge of accordingly fill in the fields of the extension header providing information regarding the carried video segments, such as bitrate and encoding information as passed from the application layer during the content operations invocations.

Video Client and Server. We use the presented host stack and API to implement a modified DASH system that exploits the in-network service. In order to implement a DASH video client that can rely on the in-network adaptation instead of implementing bitrate adaptation locally, we took advantage of the *VLC-DASH* plugin presented in [19] and modified it to display received segments independently of the delivered representation. We've implemented a basic DASH webserver using MF network API that handles requests for content (i.e., video segments and meta files) addressed by their GUIDs. In order for the plugin to work with the MF network and the implemented server, HTTP requests are forwarded to a proxy co-located on the same machine that translates URL request to MF content GUID requests. Mappings from video segments URLs to GUIDs is implemented using a local database, but we plan on developing a name resolution service for future experiments; a single GUID is used to identify segments independently from their bitrate.

Router. The software router is implemented as a set of routing and forwarding elements within the Click modular router. The router implements dynamic-binding using GNRS, hop-by-hop transport, and storage-aware routing. It integrates a large storage – an in-memory *hold buffer* – to temporarily hold data blocks when destination endpoints during short-lived disconnections or poor access connections. For dynamic in-network binding of GUID to NA, the router is closely integrated with the in-network GNRS. It attaches to a local instance of the distributed service, which is often co-resident on the physical device, but can also be hosted on separate co-located node. GUID-to-NA mappings once looked-up are cached by the router until TTL or expiry values established at the time the binding was created. The access routers implement a rate monitoring service that tracks the available bandwidth for each attached client. For the WiMAX network, the rate is obtained from the WiMAX base station which exports the most recent downlink bitrate allocated to each client by the scheduler based on a client's location, client offered traffic, and overall load on the BSS. We have implemented a similar rate monitoring capability for WiFi Access Points using standard 802.11 netlink configuration utilities. The monitored information is then used to select the eventual necessity invoking the transcoding service; this choice is implemented as a simple threshold logic where segments are forwarded if their required bitrate is not met by the available bandwidth. In the case segments need to be transcoded, the router forwards them to the transcoding service adding the available experienced performance to the chunk's extension header.

Cloudlet. Our hosting platform for compute services is based on the PacketCloud framework [6]. A cloudlet at a minimal is composed of a controller module, a pool of compute nodes, and a service interface that exposes management API for application providers to manage the lifecyle of their compute services. The controller can dynamically provision whole compute nodes per service or run them in virtual environments with clean isolation. For our Linux-based implementation, we use Linux Containers (lxc) as a light weight VM solution. The controller interfaces with the co-located router to register/de-register service GUIDs to control traffic steering between the router and the compute service. In our current implementation, traffic is steered over a TCP link set up between the software router and a basic TCP server running within the VM that hosts the service. The interaction protocol encodes the packet payload and some meta information such as input arguments supported by the compute extension headers. Lower overhead mechanisms to implement the interface are being considered.

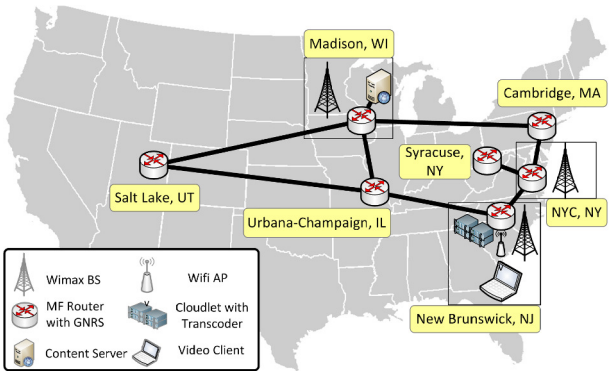

Fig. 3: Prototype components deployed on the GENI testbed

Rate-Adaptor Service. The in-network rate-adaptor service combines both caching and transcoding functions. Video segments that require transcoding to a different bitrate can either be transcoded in real time, or returned from the cache if a version of the segment in the target bitrate exists. The cache entries can be populated from both in-transit video segments as well as those that emerge from a transcoding operation. Limitations on both storage and compute resources present interesting tradeoffs across optimal reuse and minimizing the latency of transcoding in real-time. The transcoding functions are based on the ffmpeg multimedia framework. Once received segments reach the adaptor they get transcoded based on the statistics obtained from the chunk extension header, selecting between the different potential encoding bitrates.

VI. Prototype Evaluation

A. Deployment on GENI Wide-Area Testbed

The GENI testbed supports deep programmability by allowing experimenters to run custom network and software stacks on testbed nodes, and through flexible interconnection specification (incl. layer-2 links) [20]. It provides scale and a large geographic footprint by stitching together several academic and other non-commercial testbeds using Internet2's 10/100 Gbit backbone network and a host of other regional networks that connect up the individual institutions. The resulting nation-wide testbed with a variety of wired and wireless segments, aims to emulate, albeit in a limited way, the heterogeneity of the real Internet.

Experiment Setup. We deployed MF prototypes at seven GENI sites as shown in Figure 3. The routers, naming servers, and applications run on Xen VMs (total 14, 2 VMs per site)

Link	Link Type	Bandwidth (forward / reverse)	Rtt (ms)
Video Server (Wisconsin) — MFR (Wisconsin)	10G Ethernet (VMs shared)	4.86 / 2.58 Gbps	0.72 ± 0.11
MFR (Wisconsin) — MFR (Illinois)	Ethernet/Fiber	411 / 418 Mbps	8.9 ± 0.10
MFR (Illinois) — MFR (Rutgers)	Ethernet/Fiber	83.3 / 92.6 Mbps	78.7 ± 3.95
MFR (Rutgers) — Transcode Server (Rutgers)	10G Ethernet (VMs shared)	937 / 479 Mbps	0.51 ± 0.04
MFR (Rutgers) — Video Client (Rutgers)	WiMAX	9.49 / 1.05 Mbps	54.88 ± 2.87

TABLE I: Performance of links on the path from server at Wisconsin to VLC client at Rutgers.

each with 1 GB memory and one 2.09 GHz processor core. At the Rutgers site we also provision a raw node to run the transcoding server. Each MFR is configured with 1 or 2 interfaces depending on their role as core router or as an access/edge router, respectively. All routers have a core-facing interface connected to a layer-2 network that connects all seven sites. This was setup using a multi-point VLAN feature provided by Internet2's Advanced Layer-2 Service (AL2S). Routers at three sites (viz. Wisconsin, Rutgers, NYU) are configured with a second interface connecting to the local wireless network (WiMAX). Mobile wireless or emulated clients connect to MF network through this interface. Routers are each configured with 500 MB of hold buffer space, and have access to a GNS service instance co-located on the same node. The GNS service runs at all seven sites using a replication factor of k=3, and achieves a 95th percentile look latency of under 80ms.

B. Evaluation of In-Network Rate-Adaptation

We used the above setup to evaluate the in-network rate-adaptation service for a DASH video streaming application. The MF-enabled DASH server ran at the Wisconsin site, while the VLC client at Rutgers connected over WiMAX. Though the Wisconsin site has a WiMAX client network, we connect the content server over Ethernet to the access router to reflect the high-bandwidth uplinks for server deployments. Table I shows the performance of links along the server-client path, showing clearly that the client link is the potential bottleneck.

DASH Video Dataset. We use the DASH video dataset generated by Lederer et al. [21] using their DashEncoder. In particular, we use the DASH-ized files they generated from a 1080p version of the *Big Buck Bunny* animated movie available from here [22]. The DASH version of this video is available in six different segment lengths (viz. 1, 2, 4, 6, 10 or 15 sec) and at 15 to 20 different bitrate encodings (from 50 Kbps up to 8000 Kbps) for each segment length. In our experiments we use the fixed 480p resolution dataset with 2 second segment lengths. The fixed resolution allows fixed playback size and the shorter segment lengths allows for entire segment to be encapsulated in a single chunk for the hop-by-hop transfer.

Microbenchmarks. We measured the latency overhead of steering video chunks to the in-network transcoding service as the time taken to packetize the chunk received at the router, transmit it to the service, wait time, time to receive the response, and to chunkify the payload so it can be forwarded-on to the destination. Figure 4 shows the response time in relation to the original chunk size. Since actual transcoding delays vary substantially with the transcoding profile, we benchmark here the 'cached' transcode operation. Upon receiving a chunk and extracting the the encoded bitrate and

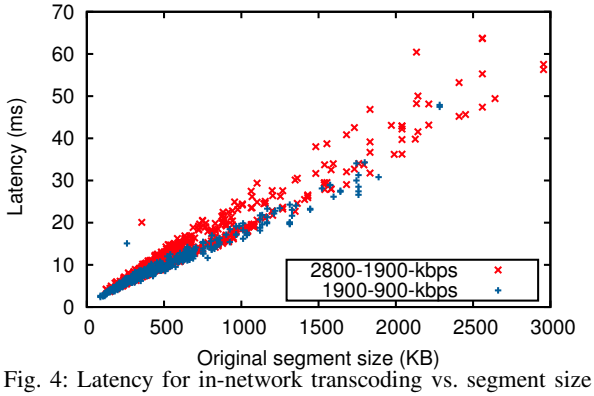

Fig. 4: Latency for in-network transcoding vs. segment size

the target bitrate parameters from the headers, the server responds with a pre-transcoded file of the requested bitrate. Therefore the overheads are primarily due to the messaging operations at the router and cloudlet server, and the cost to read the transcoded file from disk. In these benchmarks, we warm the cache sufficiently to enable files to be served from system buffers. As seen from the results for two sets of transcode operations (2800 to 1900 Kbps, 1900 to 900 Kbps), the overheads are in the order of few milliseconds to a few 10s of milliseconds for the larger segments. The big part of the delay is the transmission time, making a case for tighter integration of the compute service. These substantial delays also emphasize smart choice for segment sizes under variable client-access conditions, such as with a mobile client.

Emulated Mobility Experiment. We had previously described how the client link is monitored at the access router via the information exported by the WiMAX base-station. Any variation in available bandwidth due to mobility or load is available almost instantaneously at the router. However, for the set of experiments we ran, the client is fixed and hence sees little variation in its access link properties. We therefore emulated the client mobility by intentionally modifying the bandwidth reported by the measurement service to drop below the rate required to support smooth video streaming at the original encoded bitrate. During playback, the VLC client keeps requesting the server for video segments of the animation video as long as it does not fill its buffer measured in terms of seconds of video. Under favorable conditions, the access bandwidth available to the client is sufficient for streaming the highest bitrate content made available to the user, i.e. 1900 kbps. This is the initial stretch (until about 30 sec) seen in the client-side traffic plot of Figure 5. During this part, the video segments traverse the server-client path without being steered for in-network transcoding. Then as we emulate client mobility at about t=30 sec, the lower client bandwidth triggers steering of traffic to the transcoder, tagged with a target bitrate of 900 Kbps. The steered traffic and the response traffic with

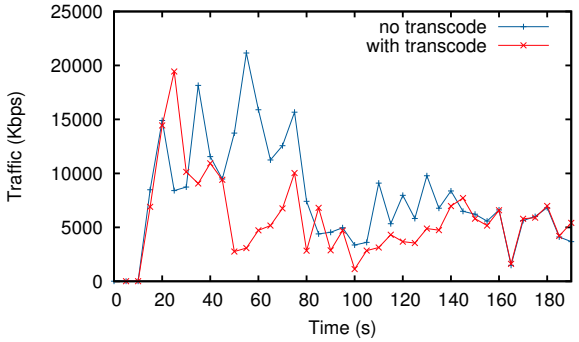

Fig. 5: Downlink traffic at client's WiMAX interface

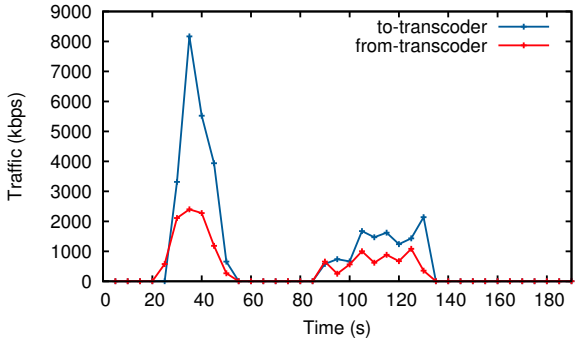

Fig. 6: Traffic between Rutgers MFR and transcoder

transcoded segments at the router-transcoder link is seen in Figure 6. We emulate a second mobility event at about t=85 sec which once again reduces the traffic on client's downlink. Note that since the client's downlink is actually fine, the reduction is not an artifact of drop in available bandwidth. Interestingly, however, the reduction during the second event is less dramatic. The variable bitrate content happens to be of lower size during this stretch. To handle such cases, it would be help to have an estimate of the required over-the-air bitrate at a finer granularity, perhaps at the segment level, to enable more careful steering of segments for transcoding. While we did not quantitatively evaluate video quality at the VLC client besides bitrate (the quality transitions were noticeable of course), we can report that we didn't see any buffering pauses.

VII. Future Work

In this paper we presented the details of the pluggable compute layer extensions we are exploring within the MobilityFirst project. To demonstrate its usefulness we presented details of our implementation of an in-network video transcoding service that we evaluated over the GENI testbed using a set of programmable nodes distributed across seven sites in the US. Early results from our prototype evaluation indicate feasibility and usefulness, however, a more detailed parametric study of the video service under variable network conditions and realistic mobility, and with consideration of user experience aspects, will help make a case for practical usefulness and convenience of our in-network rate adaptation service.

References

[1] "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013-2018," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf, 2013.

[2] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 225–238.

[3] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 2013, pp. 389–400.

[4] "MobilityFirst Future Internet Architecture Project," http://mobilityfirst.winlab.rutgers.edu/.

[5] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[6] Y. Chen, B. Liu, Y. Chen, A. Li, X. Yang, and J. Bi, "Packetcloud: an open platform for elastic in-network services," in *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture*. ACM, 2013, pp. 17–22.

[7] F. Bronzino, K. Nagaraja, I. Seskar, and D. Raychaudhuri, "Network service abstractions for a mobility-centric future internet architecture," in *Proceedings of the eighth ACM international workshop on Mobility in the evolving internet architecture*. ACM, 2013, pp. 5–10.

[8] T. Vu, A. Baid, Y. Zhang, T. D. Nguyen, J. Fukuyama, R. P. Martin, and D. Raychaudhuri, "DMap: A shared hosting scheme for dynamic identifier to locator mappings in the global internet," in *Distributed Computing Systems (ICDCS), 2012*. IEEE, 2012, pp. 698–707.

[9] A. Sharma, X. Tie, H. Uppal, A. Venkataramani, D. Westbrook, and A. Yadav, "A global name service for a highly mobile internetwork," in *Proceedings of the 2014 ACM conference on SIGCOMM*. ACM, 2014, pp. 247–258.

[10] S. Farrell, V. Cahill, D. Geraghty, I. Humphreys, and P. McDonald, "When TCP breaks: Delay- and disruption-tolerant networking," *IEEE Internet Computing*, vol. 10, no. 4, pp. 72–78, 2006.

[11] M. C. Chan and R. Ramjee, "TCP/IP performance over 3G wireless links with rate and delay variation." *Wireless Networks*, pp. 81–97, 2005.

[12] M. Li, D. Agrawal, D. Ganesan, and A. Venkataramani, "Block-switched networks: a new paradigm for wireless transport," in *Proc. of NSDI*, 2009.

[13] S. Gopinath, S. Jain, S. Makharia, and D. Raychaudhuri, "An experimental study of the cache-and-forward network architecture in multi-hop wireless scenarios," in *Proc. of LANMAN*, 2010.

[14] S. C. Nelson, G. Bhanage, and D. Raychaudhuri, "GSTAR: Generalized storage-aware routing for mobilityfirst in the future mobile internet," in *Proc. of MobiArch*. ACM, 2011, pp. 19–24.

[15] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri, "EIR: Edge-aware interdomain routing protocol for the future mobile internet," WINLAB, Rutgers University, Tech. Rep. WINLAB-TR-414, June 2013.

[16] T. Stockhammer, "Dynamic adaptive streaming over http: standards and design principles," in *Proceedings of the second annual ACM conference on Multimedia systems*. ACM, 2011, pp. 133–144.

[17] C. Müller, S. Lederer, and C. Timmerer, "An evaluation of dynamic adaptive streaming over http in vehicular environments," in *Proceedings of the 4th Workshop on Mobile Video*. ACM, 2012, pp. 37–42.

[18] H. Riiser, H. S. Bergsaker, P. Vigmostad, P. Halvorsen, and C. Griwodz, "A comparison of quality scheduling in commercial adaptive http streaming solutions on a 3g network," in *Proceedings of the 4th Workshop on Mobile Video*. ACM, 2012, pp. 25–30.

[19] C. Müller and C. Timmerer, "A vlc media player plugin enabling dynamic adaptive streaming over http," in *Proceedings of the 19th ACM international conference on Multimedia*. ACM, 2011, pp. 723–726.

[20] "Global environment for networking innovations (GENI)," http://www.geni.net.

[21] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 89–94.

[22] "Big Buck Bunny movie," http://bigbuckbunnymovie.org/.