ACACIA – Context-aware Edge Computing for Continuous Interactive Applications over Mobile Networks*

Junguk Cho
University of Utah
junguk.cho@utah.edu

Karthikeyan Sundaresan
NEC Labs America Inc.
karthiks@nec-labs.com

Rajesh Mahindra
NEC Labs America Inc.
rajesh@nec-labs.com

Jacobus Van der Merwe
University of Utah
kobus@cs.utah.edu

Sampath Rangarajan
NEC Labs America Inc.
sampath@nec-labs.com

ABSTRACT

There is widespread agreement that future continuous interactive (CI) applications will require edge computing capabilities from mobile networks. There is also widespread expectation that the emerging 5G network architecture, with its constituent technology components, will be the context in which this will be realized. Indeed many of the components that will be part of such an environment have been studied in standalone manner. However, the question of whether an end-to-end combination of these components would satisfy application requirements, or indeed, how these components would be combined into service offerings by mobile network providers, have not been meaningfully addressed. Towards addressing these challenges, we propose ACACIA- a service abstraction framework that enables CI applications on edge clouds in mobile networks. Evaluation of our prototype implementation shows that our holistic approach provides a 70% end-to-end application level latency reduction when compared with existing cloud and mobile solutions.

CCS Concepts

•**Networks** → **Mobile networks;** *Cloud computing; Wireless access networks;*

*Instructions for accessing an ACACIA profile in the PhantomNet testbed are available here: https://wiki.phantomnet.org/wiki/phantomnet/acacia

CoNEXT '16, December 12-15, 2016, Irvine, CA, USA
 ISBN 978-1-4503-4292-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/2999572.2999604

Keywords

mobile network; service offerings; context-aware mobile edge computing; NFV/SDN

1. INTRODUCTION

Future continuous interactive (CI) applications, like augmented reality (AR), virtual reality (VR), autonomous driving, etc [45, 21, 23], will have network requirements that cannot be provided by existing mobile networks. CI applications might require very low end-to-end latencies (low tens of milliseconds or less) [19, 37], and might similarly require computational resources to be close to the network edge [47]. Current 4G mobile networks are incapable of providing such delays since the core mobile network itself incurs a substantial delay (high tens of milliseconds or more) [33, 52]. Further, current mobile networks provide basic communication service abstractions aimed at human-to-human and human-to-machine type communication. To effectively support future CI applications *both* the underlying mobile network technology as well as the service abstractions provided by the network need to evolve.

There is widespread expectation that the emerging 5G mobile network architecture will address these concerns, that cloud platforms in the form of mobile edge computing (MEC) will satisfy edge computing requirements and that localization and context discovery frameworks will provide the means for CI applications to be realized. Indeed many of these necessary components have been proposed and studied in standalone manner. Mobile edge computing has been the subject of both academic and industrial efforts [31, 51, 22, 43]. Computational offloading has also been studied to facilitate CI applications [48, 47, 34, 38, 32]. User localization [25, 41, 50, 29, 40] and context discovery [5, 7, 17] frameworks have been proposed in support of CI applications. However, the question of whether an end-to-end combination of these components would satisfy application requirements, or indeed, how these components would be combined into service offerings by mobile network providers, have not been

meaningfully addressed. Given the real-time interactions between user, network and application, the key question we ask is *how should the three entities (user, network and application) be jointly orchestrated and combined in a service abstraction to deliver low latencies and enable CI applications in mobile networks?*

We answer this question through the design of our service abstraction framework called ACACIA. ACACIA enables CI applications on edge clouds in mobile networks. ACACIA leverages client context information through proximity service discovery to optimize network and application processing. ACACIA leverages SDN/NFV principles with LTE/EPC QoS mechanisms to steer CI application traffic, on demand, to closely located mobile edge clouds. We expect our work to inform the emerging 5G mobile network architecture, however, we note that the use of SDN and NFV mechanisms allow our approach to be readily realized with existing LTE/EPC mobile networks.

In designing ACACIA our key insight is that a user's context with respect to her environment is a critical factor in optimizing both the network and application aspects required by MEC-based CI applications. Specifically, a CI application using MEC-based services should be directed to an edge cloud instance in close proximity to the user where the required services are available (i.e., providing network level optimization). Similarly, the specific context of the user can be used to inform and optimize the application functionality. For example, a cognitive assistance face recognition AR application [30] might optimize its functionality by knowing where the user is (e.g., normal domestic environment vs. shopping mall) and knowing whether a person interacting with the user is someone the user interacted with before (i.e., family or friends in a domestic environment, vs. acquaintances associated with service outlets the user frequents). Similarly, a retail shopping AR application can benefit not only from knowing the relative location of a user (e.g., the appliance department), but also from knowing that the user is interested in a specific type of appliance (e.g., a microwave oven).

ACACIA leverages this insight to realize *Context-Aware MEC for CI applications over mobile networks*. Specifically, ACACIA leverages user context in the form of their interest in specific CI proximity "services" as well as proximity to landmarks to (i) create *on-demand* connections to MEC servers only when a client is interested in CI applications and MEC-based CI services are available, while maintaining an always-on connection to the core network, thus alleviating the control overhead in establishing such additional connections, (ii) automatically *classify* MEC traffic at the source (client) itself, and *re-direct* only the MEC traffic to the closest MEC server, thereby avoiding the need for traffic inspection and classification as well as the load of non-relevant traffic at the MEC server, and (iii) speed-up the application processing through *location-aware optimization*, that is especially useful in CI services like AR/VR.

We developed an end-to-end prototype of the ACACIA architecture, specifically using a software LTE/EPC platform, Open vSwitch, and an SDN controller to implement the decoupled control and data planes of ACACIA's LTE network function gateways, thus realizing a centralized control plane but a distributed data plane. Our prototype leveraged the LTE-direct D2D (Device-to-Device) [5] capabilities of commercial smartphones to provide the user context necessary to intelligently orchestrate and optimize the network as well as the CI service. With LTE-direct, ACACIA avoids incurring any infrastructure overhead and easily ties it to other components as a potential service offering by mobile carriers. While ACACIA can be applied to various low latency CI applications, to demonstrate its efficacy, we built a sample real-world AR application for engaged retail services that utilizes ACACIA's MEC and user-context optimization. Our retail application enables store owners to simply equip their sales people in different sections of the store with smartphones capable of LTE-direct, and customers are presented with an enriched shopping experience in real-time as they explore the store with their smartphones.

To our knowledge, ACACIA represents the first holistic end-to-end design to support CI applications in mobile networks. Using our prototype realization we performed extensive evaluation to validate our design. Our evaluations reveal that ACACIA provides a 70% end-to-end application level latency reduction when compared with existing cloud and mobile solutions, and a 60% reduction compared with a mobile edge cloud solution that only optimizes network latencies. We make the following specific contributions in this work:

- We design and build a service abstraction framework that leverages user context information, made available without any infrastructure support through LTE-direct capabilities, to intelligently orchestrate and optimize both the MEC network and CI service.
- Leveraging SDN/NFV with existing QoS bearer framework in LTE network, we empower an LTE core network (EPC) with MEC capabilities, without requiring any changes to LTE specifications, additional hardware/middleboxes and modifications to LTE base stations[1].
- We build and deploy an AR based interactive retail service on commercial smartphones which uses an existing LTE network enhanced with our framework.

2. RELATED WORK

Computation Offloading: To support mobile applications requiring heavy computation, several offloading frameworks have been proposed [48, 47, 34, 38, 32]. The Gabriel system [34] follows a long history of cloudlet related works [48, 47] and proposes a multi-tiered mobile offloading architecture in support of cognitive assistance applications. MAUI [32] proposed a system which supports fine-grained code offload from a smartphone to a server to maximize energy savings with minimal developer effort. OverLay [38] proposed a

[1]In the remainder of this paper we follow common practice by referring to current 4G long term evolution (LTE) radio access networks combined with evolved packet core (EPC), simply as "LTE" networks.

practical mobile augmented reality framework. It reduces the search space for computer vision algorithms based on human behavior and uses GPU processing to accelerate computation. These efforts have mainly focused on reducing application level computation latency and do not address the networking complexities that arise in realizing such offloading in mobile networks. ACACIA aims to enable these offloading solutions over mobile networks through a novel and practical mobile edge networking approach.

MEC: Several works [31, 51, 22, 43] have proposed mechanisms and architectures for Mobile Edge Computing (MEC) over mobile networks: (i) Traffic offload mechanisms located at the eNodeB or close to it (in middleboxes) can inspect traffic and route them to dedicated MEC servers [31, 51]. This approach incurs the cost of middlebox deployment and the processing overhead of traffic inspection and de/encapsulation of GTP tunnels. Further, deploying these solutions without modification to current LTE networks, requires monitors to sniff all control messages. (ii) Extension to small-cell home eNodeB deployments for local IP access (LIPA [8]) has been proposed to realize cloudlet [48] functionality in the home environment [22]. While this is conceptually similar to an MEC environment, it is not clear how (or whether) this approach generalizes beyond the home environment. (iii) The new industry initiative for MEC aims to develop an architecture and standards to allow flexible MEC realizations [43]. While details of their work are still emerging, their approach calls for the integration of computing elements with radio access network (RAN) equipment, which suggests the need for new standardized protocol interfaces to be developed. This in turn will require new equipment (or at best upgrades to existing equipment) before the approach can become reality. ACACIA's MEC approach differs from the above approaches in that it does not require new protocol interfaces, extra hardware or heavyweight monitoring to realize MEC in current LTE networks. In addition, it provides low latencies for mobile users as well as reduced network overhead to effectively realize CI applications based on service availability and user interests.

Service Discovery: There are various proximity service discovery techniques [5, 7, 17] like LTE-direct [5], iBeacon [7], and Wifi-Aware [17] that can be used to provide user context. Further, there is a rich set of literature in indoor localization [25, 41, 50, 29, 40]. From a mobile provider's perspective, there is synergy in using LTE-direct as part of its service offerings. Hence, ACACIA employs LTE-direct for proximity service discovery and coarse indoor localization. However, ACACIA's focus extends beyond service discovery and localization to more importantly, optimize and enable context-aware CI applications.

While the above works clearly indicate active research in various areas (computation offloading, MEC, service discovery and localization) in isolation, it is not clear how these components need to be combined synergistically to provide an end-end service offering for CI applications. Addressing the challenges that arise from realizing this vision, is the objective of our work on ACACIA.

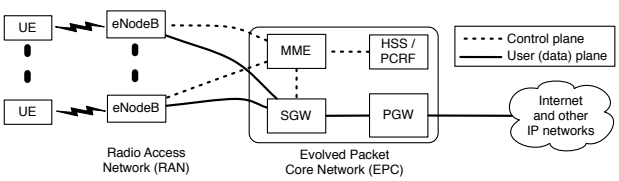


Figure 1: LTE/EPC Architecture

3. BACKGROUND

Acronym	Definition
EPC	Evolved Packet Core
eNodeB (eNB)	E-UTRAN Node B
UE	User Equipment
MME	Mobility Management Entity
HSS	Home Subscriber Server
PCRF	Policy and Charging Rule Function
SGW	Serving Gateway
PGW	Packet Data Network Gateway
GTP	GPRS Tunneling Protocol
TEID	Tunnel EndPoint Identifier
TFT	Traffic Flow Templates
QCI	QoS Class Identifier
MEC	Mobile Edge Computing
SCTP	Stream Control Transmission Protocol

Table 1: Frequently used acronyms

LTE/EPC Architecture. The 4G mobile network architecture consists of the Radio Access Network (RAN) and the Evolved Packet Core (EPC) as shown in Figure 1. While the RAN includes eNodeBs (base stations) that serve the user equipment (UEs), the EPC consists of both control-plane components that *manage* the devices and data-plane components that *forward* the data traffic. The control-plane components mainly consist of the MME (Mobility Management Entity), the HSS (Home Subscriber Server) and the PCRF (Policy and Charging Rule Function). The data-plane components consist of the Serving Gateway (SGW) and the Packet Data Network Gateway (PGW) which primarily perform forwarding of user data traffic. In addition, the SGW serves as an anchor point in case of handover between eNodeBs and contains buffers for paging functionality. The PGW acts as the Internet gateway for the data traffic and also enforces operator-defined policies (QoS), packet filtering and accounting. After a UE is attached to the mobile network, packets from the UE are forwarded from eNodeB to the two GWs and vice versa for traffic towards the UE.

Role of SDN/NFV in LTE Networks. With advances in Network Function Virtualization (NFV) and Software Defined Networking (SDN), it has been widely advocated [42, 46, 6] that the two GWs should be split into control-plane entities (GW-C) and user-plane entities (GW-U). The GW-Cs interact with other control-plane entities like MME using 3GPP standard interfaces and setup the forwarding paths

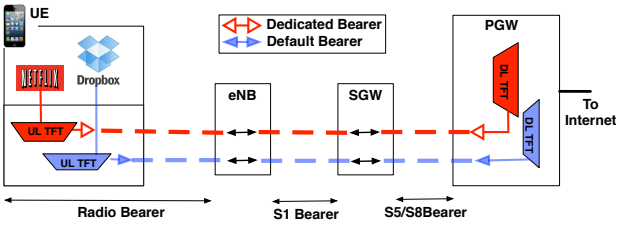


Figure 2: Bearers in LTE/EPC [49]

on the GW-U entities using OpenFlow. This split architecture has several advantages. It allows for (i) better scaling of control and data planes independently, (ii) fine-grained load balancing across the SGW-Us and the PGW-Us by the SGW-C and PGW-C and (iii) flexible management of multiple SGW-Us and PGW-Us (distributed across data centers) by logically or physically centralized SGW-C and PGW-C.

Bearers in LTE/EPC. When a UE registers with the network, a default data-path called a *default bearer* is created between the UE and the PGW to ensure always-on connectivity. Each bearer is composed of three segments: (i) radio bearer between UE and eNodeB, (ii) data bearer (S1 bearer) between eNodeB and SGW and (iii) data bearer (S5/S8 bearer) between SGW and PGW as shown in Figure 2. The data bearers are carried over GTP based UDP tunnels that are differentiated by a unique GTP Tunnel EndPoint Identifier (TEID). In addition to the default bearer, *dedicated* bearers can be set up for specific traffic flows, which are identified using traffic flow templates (TFTs) [3] in both up (in UE) and downstream (in PGW) directions (essentially a five-tuple packet filter). In addition to providing a basic connectivity abstraction, a bearer can also be associated with a QoS Class Identifier (QCI) [4], which specifies the priority, packet delay budget and packet loss rate associated with the bearer's traffic. Figure 2 shows an example instantiation with both default and dedicated bearers. In this example the dedicated bearer, with better QoS metrics, is associated with traffic flow for a Netflix application, while other traffic, e.g., associated with Dropbox, is carried on the default bearer.

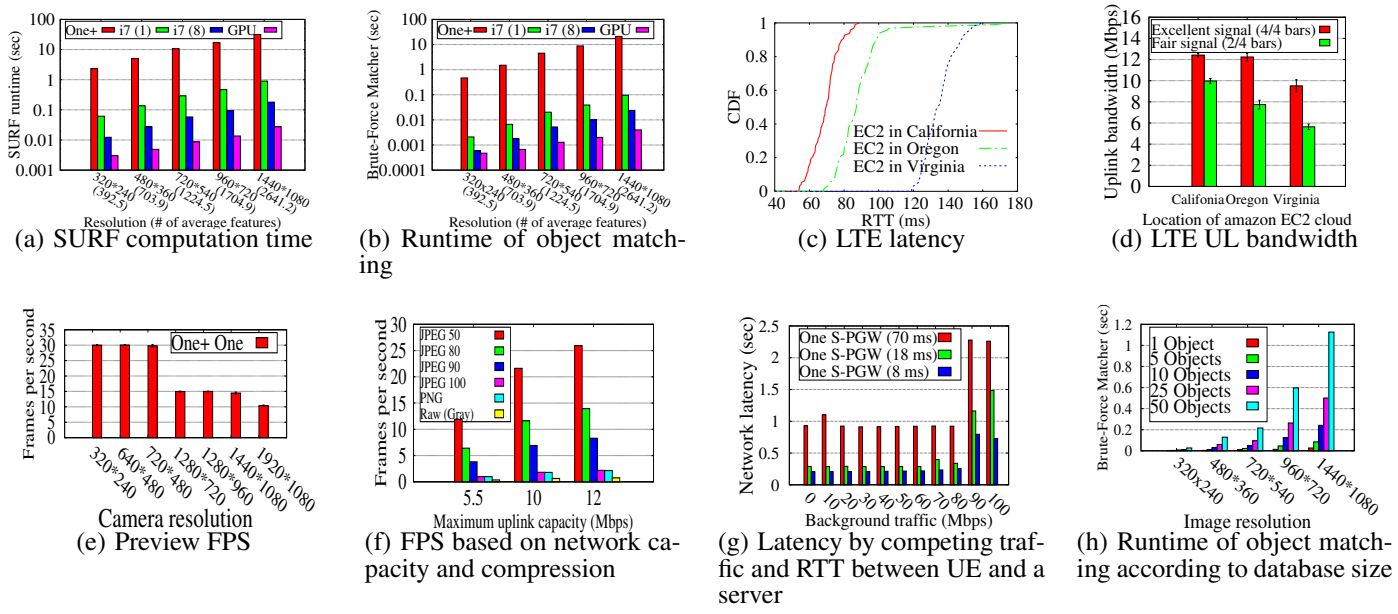
LTE-direct. LTE-direct [5] is a proximity service discovery techniques using a device-to-device (D2D) paradigm. It employs *Publish* and *Subscribe* messages on the radio (LTE) channel to advertise and discover nearby services. It is particularly attractive owing to the ubiquitous availability of LTE on smart devices (no additional infrastructure required), and provides superior range and robustness. In addition, it leverages the eNB for timing information, resource allocation as well as user authentication without requiring explicit and complex initial setup procedures [5]. Resources for service discovery are allocated in the uplink part of the LTE spectrum, which is lightly loaded compared to the downlink. At periodic intervals (e.g., 5 or 10 sec), the eNB allocates resource blocks (RBs) for LTE-direct transmissions as part of its uplink frames. This has a negligible impact on the uplink capacity (utilizes $< 1\%$ of uplink resources [5]). All information related to service discovery is stored and handled in the LTE modem itself. While a service discovery message from a publisher is broadcasted on a periodic basis, application specific information (binary codes and their mask that express user's interest) stored at the subscriber is used for filtering and matching the service discovery message. If matching happens in the LTE modem, the service discovery message is forwarded to the appropriate application and translated by the application. Otherwise, it is filtered out in the LTE modem. Handling service discovery entirely in the modem allows for scalability (hundreds of devices [5]), security and fast discovery.

4. MOTIVATION

To understand the various factors impacting the latencies perceived by CI applications over mobile networks, we experimented with augmented reality (AR) as a representative CI application. AR-based applications overlay useful information like text, audio and video over the live scene of the user's current environment in real-time, and find numerous use-cases (object information in retail shops, navigation in museums, cognitive assistance, etc.). Hence, they require tight end-to-end latencies (low tens of ms) for satisfactory interactive user experience [22, 38].

Computation Offloading. We measure the latency incurred by running two important AR-related operations, namely feature detection and description (finding features (keypoints) from an image and describing them (descriptors) using the SURF algorithm [27]) and object matching (using keypoints and descriptors to compare them against in the database) by using the OpenCV [12] library. Figure 3(a) and Figure 3(b) show computation time for SURF detection and description and for object matching respectively as a function of image resolution on four different devices: (i) the latest One+ One smartphone device, (ii) a single i7 core server, (iii) an eight i7 core server and (iv) a server with a GeForce GTX TITAN GPU processor. When AR operations are executed on the smartphone, the computation time is quite high, e.g., 2 seconds even with the lowest image resolution, suggesting the unfeasibility of deploying the AR application on the smartphone itself. Running AR operations on the server on the average shows 36x (1-core), 182x (8-cores) and 1087x (GPU) reduction in latency for feature detection and description computation and 223x (1-core), 852x (8-cores) and 3284x (GPU) latency reduction for matching operations. These results indicate the necessity of using the offloading compute resources of CPU and GPU to realize the latencies demanded by AR applications.

Low Network Latency. The always-on, universal availability and stable operation of LTE in licensed spectrum make it a promising candidate for mobile devices to offload computation to the cloud. However, cloud servers being located far from mobile devices and centralized S/P-GWs employing hierarchical routing in the core network [33, 52] result in high and unpredictable latencies. For example, Figure 3(c) shows the round-trip-time (RTT) measurements in a commercial LTE network from a smartphone in the US midwest to Amazon EC2 servers in different locations. Consistent with previously reported LTE measurements [36], our



(a) SURF computation time (b) Runtime of object matching (c) LTE latency (d) LTE UL bandwidth

(e) Preview FPS (f) FPS based on network capacity and compression (g) Latency by competing traffic and RTT between UE and a server (h) Runtime of object matching according to database size

Figure 3: Factors impacting AR applications

measurements show the RTT between the smartphone and the EC2 server in California is the shortest and around 70 ms, with other EC2 servers delivering much higher median latencies. Considering human sensitivity to latencies higher than roughly 25-50 ms for AR applications, the LTE network latency warrants optimization and can account for a substantial portion of the end-to-end latency [22].

High Network Bandwidth. In addition to latency, network bandwidth is also important for AR applications. AR applications typically generate a large volume of data from a camera in a smartphone, which require high bandwidth to upload the data to the cloud. Figure 3(d) shows the measured uplink bandwidth from a smartphone connected to a commercial LTE network in the midwest US to Amazon EC2 servers in different locations. The highest uplink bandwidth between the smartphone and an EC2 server in California is about 12 Mbps, which is consistent with previous results [36]. We conducted an experiment to determine whether these data rates are sufficient for AR applications. First we measured the maximum frames per second (FPS) generated by the camera preview mode in a One+ One smartphone. Figure 3(e) shows FPS based on the configuration of camera resolution. At HD resolution (1920*1080), the device generates 10 FPS. Second, we calculated the (upload) frame rates (FPS) possible via the the current LTE network. The results are shown in Figure 3(f) for different image compression approaches and as a function of the available LTE network capacity. In uncompressed mode (Grayscale image) the smartphone cannot even send one frame per second, even with the highest uplink capacity. With a reasonable compression ratios, e.g., JPEG 90 (higher number indicates less compression), the device can send 8 frames per second, which is relatively close to the FPS generated from its camera for a HD scene. While one could resort to compression, this would significantly affect the accuracy of the object detection/matching component of the AR application due to lossy compression. This clearly indicates that without guaranteeing sufficient uplink bandwidth for AR-applications, one cannot realize the ideal FPS needed to run the AR applications satisfactorily.

We note that mobile edge computing aims to address computation offloading and latency reduction, and by bringing cloud processing to the edge of the mobile network might also positively impact network capacity constraints. However, as we consider below, other challenges remain.

Traffic Differentiation. MEC provides a means to bypass the core network bottleneck by handling the traffic closer to the wireless access. While proximity of the MEC server to the UE helps, it is equally important to control the traffic load that is directed to the MEC server and hence needs to be processed/classified by it. Figure 3(g) presents the network latencies observed as a function of the background traffic load that is handled by core network GWs serving the MEC server. Proximity of the server to the UE is varied by emulating different RTTs to the server. It can be seen that even for small RTTs (MEC server located close to the eNodeB), if the AR traffic competes with other background traffic, this can have a significant impact on the end-to-end network latency (e.g., in our experiment 90 Mbps of background traffic increased the end-to-end delay to approximately 800 ms which makes AR applications impractical.). Hence, it is critical to have mechanisms that differentiate between traffic, and identify only those that require the use of MEC resources and direct them to the appropriate MEC servers.

Control Overhead. The feasibility of creating dedicated bearers in LTE offers the potential to create such a bearer between the mobile device and an MEC server in the edge cloud. However, the creation of every such bearer incurs

additional overhead in the form of control messages. This control overhead is exacerbated by the fact that LTE tears down the bearers associated with a device when the device goes idle (i.e., have no data to send for 11.576 seconds [35]), and then needs to reestablish the bearers when there is data activity (known as an LTE radio promotion event). The total number of control messages (and bytes) involved with such a "release and reestablish" sequence in an NFV/SDN LTE architecture is 15 messages (2914 bytes) in our testbed setup (Composed of: SCTP 7 (1138), GTPv2 protocol 4 (352), OpenFlow 4 (1424)). Assuming such a dedicated MEC bearer is established every time the default bearer is established, this could translate to 2.58MB of control traffic per day per device, assuming bearer creation based on the behavior of popular applications (i.e., 929 times per day) [24]. For a worst case (upper bound) scenario where bearers are recreated every time after LTE radio promotion event, this number can be as high as 20MB per device per day (i.e., 7200 times.) Thus, to keep such control overheads low, one has to be judicious in the creation of separate (additional) bearers (say, for MEC) only when it is warranted. Considering the huge number of mobile users, the overheads are not negligible.

Application Optimization. In addition to the network, often most CI applications involve heavy-weight operations that incur substantial latency even in a server machine. In the case of AR applications, this would correspond to the object detection/matching operations. Figure 3(h) indicates the latency incurred in searching for an object from a database of various sizes running on an eight i7 core server. It clearly shows that the impact of pruning the database can have a significant impact on the end-end latency, making it a critical component for optimization as well.

Thus, we find that to deliver the tight latencies demanded by CI applications at end users, one needs to not only optimize the network but also the application. In the next section, we demonstrate how to accomplish this by intelligently leveraging *user context* along with SDN/NFV capabilities of the network to realize our service abstraction framework called ACACIA.

5. ACACIA'S DESIGN

Figure 4 shows an overview of the ACACIA mobile edge computing (MEC) service framework for context-aware continuous interactive (CI) applications. As shown, ACACIA assumes a core mobile network consisting of both centralized network functions (e.g., using LTE nomenclature, standard evolved packet core (EPC) functions), as well as distributed mobile edge cloud instances. Following current technology trends, the mobile edge cloud instances, in addition to hosting application servers, are capable of realizing virtualized core network functions (i.e., NFV/SDN-based EPC functions). Finally, ACACIA uses the D2D-based LTE-direct proximity discovery functionality.

Given this context, ACACIA enables context-aware CI applications by orchestrating the following functional components: (i) *User context discovery:* The ACACIA device

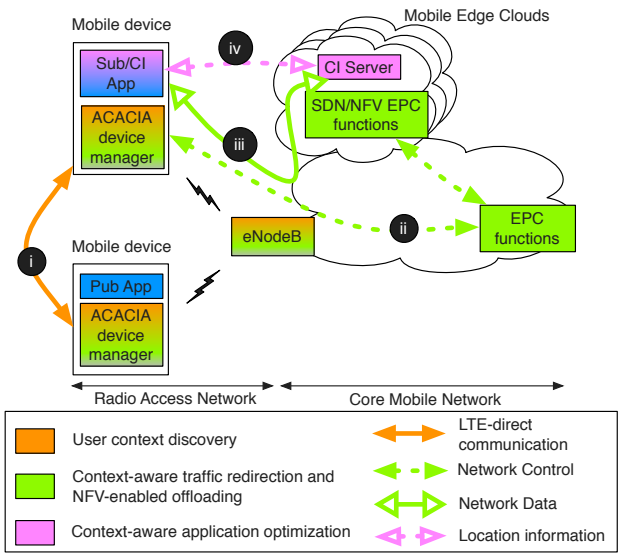


Figure 4: ACACIA Overview

manager enables applications on mobile devices to make use of the publish-subscribe capabilities of LTE-direct proximity services to advertise (publish) and learn about (subscribe) the availability of "services" that are of interest to users. The semantics of the messages associated with these service advertisements are application specific, but an "interest match" by a subscriber will trigger a CI application to provide the user with more information. (ii) *Context-aware traffic redirection:* When CI applications require mobile edge cloud resources to realize their functionality the ACACIA device manager will establish connectivity to the mobile edge cloud. The ACACIA device manager interacts with EPC functions in the mobile core network, which makes use of the proximity service discovery information to establish connectivity to the appropriate CI server on a mobile edge cloud that would satisfy the latency requirements of the CI application. (iii) ACACIA *Mobile Edge Network:* ACACIA combines existing LTE/EPC QoS bearer framework with novel SDN and NFV enabled mechanisms to realize the standards-compliant MEC data-plane offloading for only CI traffic. (iv) *Context-aware application optimization:* With connectivity to the mobile edge cloud in place, the ACACIA device manager provides location information for the CI application, which can be used to optimize its functionality. Thus, ACACIA is capable of enabling a variety of CI related use cases.

5.1 ACACIA Use Case

ACACIA can help various CI applications in significant ways. Consider an enriched retail shopping application as an example, where a retail shop uses ACACIA's service framework provided by a mobile operator. The service framework would encompass a number of components: First, an augmented reality (AR) server hosted in the mobile edge cloud. Second, the ability to use LTE-direct as a means to trigger access to the AR server. LTE-direct is used through a paired

set of applications, one used by the retail store, the other used by customers.

When the sales personnel in the retail shop start work, they open the retail shop appplication in their smartphones and select their section or products from its User Interface (UI). As a result, their smartphones periodically publish the section or product information over the air with LTE-direct. Assume a customer, who wears Google glasses or has a smartphone, and comes to the shop to buy a laptop. When she enters the shop, she opens a retail shop application which has the same UI as that of salesmen and selects 'laptop' as her interests. (Note that in a more sophisticated setting, this information might be obtained based on the user's recent internet search activity.) Her device starts subscribing to laptop related information through LTE-direct. When she comes closer to the laptop section, her retail application receives the notification in the form of an alarm and a vibration (or a notification via her wearable device). It means the "service" which corresponds to her "interest" is available in this area of the store.

After the notification, her device runs an AR application. The AR application sends the live scene of her environment to an AR server in the mobile edge cloud closest to her. The AR server sends back information relevant to the scene such as prices, reviews, and current sales using text, audio or video. Using this information, she can make an informed decision and be aware of any current incentives (sales) available to her. While the AR application runs, it also sends information on nearby "sections" to the AR server based on messages it receives through LTE-direct from other publishers. This information is then transformed to approximate user "location" in the AR server, to optimize the latter's processing of the AR application.

In this context, the mobile carrier's service abstraction is realized via its LTE mobile network, its mobile edge cloud and the LTE-direct functionality. In addition, the mobile provider might provide libraries to use LTE-direct in the service application (e.g., a retail shop application in our example) and the ACACIA device manager to seamlessly orchestrate the service application, LTE networks and user contexts. The retail store would be responsible for developing the pair of applications (for customers and employees) using the LTE-direct libraries, as well as the server application running in the mobile edge cloud.

5.2 User Context Discovery

ACACIA leverages LTE-direct to discover nearby CI services based on user's interests. Publishers (any portable device with the LTE-direct feature) can publish any information relevant to their service called *service discovery message* through a small message and broadcast it periodically over the air. The service discovery message includes the service name (e.g., a retail shop name) and its detailed information (e.g., section/products) from a service application. Subscribers (any portable device with LTE-direct feature) can decide whether they want to receive such broadcasted service discovery message by selecting their interest in specific CI applications. Different LTE-direct publishers (e.g., different retail stores) will have different LTE-direct service names (managed by the mobile carrier providing LTE-direct services). This will allow LTE-direct subscribers to distinguish between different publishers.

5.3 Context-aware Traffic Redirection

ACACIA deploys a service or daemon called the ACACIA device manager that runs on mobile devices and plays two important roles. It first works as a proxy to transfer the service discovery message requests and responses between the CI applications and the LTE modem. More importantly, it is responsible for managing network connectivity of CI applications to use their CI servers in the mobile edge cloud on demand based on user contexts. When a user starts a CI application, the CI application first connects to the ACACIA device manager and registers itself. Then, the subscriber selects her specific interests (e.g., a laptop in our retail use case) through the UI of the CI application. When there is a match (of user interests) with the service discovery message received by the LTE modem for the CI application, the ACACIA device manager sends a request for network connectivity between the CI application and CI server to MEC Registration Server (MRS). The MRS is a core network component (i.e. Application Function (AF) in 3GPP terminology) in ACACIA, which manages the CI services and starts creating/deleting the network connectivity to the CI server in the mobile edge cloud instances. The first service discovery message from the ACACIA device manager, which has the information on the service, is used to locate the closest CI server in the MRS. When the user finishes using the CI application, the ACACIA device manager sends the request to delete network connectivity to the MRS, which results in the deletion of network connectivity for CI server in mobile network. Then it unregisters the ACACIA device manager, which deletes all relevant CI application information. The detailed procedure for network management between the CI application and its server is explained in Section 5.4.

The above design has two advantages. First, since the ACACIA device manager requests creation/deletion of network connectivity to the MRS based on user context *on demand*, it avoids maintaining two always-on bearers (default and dedicated), which would cause the unnecessary control overhead identified in Section 4. Second, it allows decoupling the CI application logic from MEC connectivity setup logic. Thus, application developers do not have to worry about provisioning the MEC connections closer to the UEs, as this is transparently handled by the ACACIA device manager, the MRS and mobile network components.

5.4 ACACIA Mobile Edge Network

ACACIA achieves the network connectivity between CI application and its server and selective CI traffic offloading to the appropriate CI server as shown in Figure 5. It leverages the user context-aware ACACIA device manager and the QoS bearer framework (i.e., a dedicated bearer) in LTE to selective steer traffic towards the MEC-based server. Specifically, traffic classification happens on the mobile device, where CI application traffic is sent on a dedicated bearer

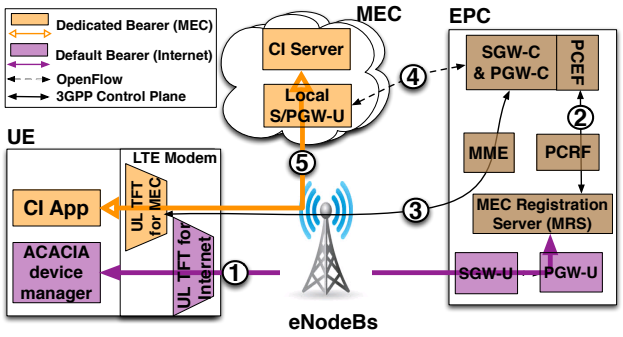


Figure 5: ACACIA Mobile Edge Network Architecture.

associated with the CI application and server. ACACIA uses split mobile GW functionality (i.e., GW-C and GW-U), to terminate the dedicated bearer on a set of GW-Us resident on the MEC. (This is realized using SDN and NFV functionality.) We denote GW-Us that serve the CI server and handle dedicated bearers as *local* GW-Us. Other GW-Us handle the default bearers, which connect UEs to the internet.

① *Request:* When the ACACIA device manager receives a match through LTE-direct, it sends a request to setup a dedicated bearer to MEC Registration Server (MRS) through the default bearer. The request includes the UE's IP address and service information.

② *Create:* Once the request is successful, the MRS signals the PCRF (Policy Control and Charging Rules Function) with the service ID, and flow information including IP addresses of UE and CI server corresponding to service information. The PCRF dynamically generates policy rules, which consist of service ID, QCI, and flow information from the MRS. The PCRF invokes the PCEF (Policy and Charging Enforcement Function) in PGW-C with the policy rules, which results in a network-initiated dedicated bearer activation [2].

③ *Set-up:* PGW-C, SGW-C, MME, eNB, and UE then exchange control messages to setup a dedicated bearer between these various components. To setup data plane for the dedicated bearer between *local* GW-Us, PGW-C and SGW-C should provide IP addresses of *local* GW-Us (not those of GW-Us serving default bearers) as the F-TEID (Fully Qualified Tunnel Endpoint Identifier) values in the *Create Bearer Request/Response* control messages. In addition, the MME sends the received IP address of the *local* SGW-U from SGW-C to eNB in the *Bearer Setup Request* control message, which ensures that only the MEC traffic is routed through eNB and *local* SGW-U. The eNB then sends the new Radio Bearer ID, Radio Bearer QoS, and IP address of the CI server as the TFT values in *RRC Connection Reconfiguration* control message to the UE. This ensures that the traffic for the MEC server uses a radio bearer (i.e., UE and eNB) that corresponds to the dedicated bearer, based on the traffic filtering function (UL TFT) that exists in the LTE modem.

④ *Route:* After successful exchange of control messages, SGW-C and PGW-C insert SDN (OpenFlow in our set-up) flow rules into the *local* SGW-U and PGW-U based on information such as TEID, and IP address of GW-Us and eNB from control messages. At this point the CI applications can use the local CI server (⑤).

ACACIA design presents a cleaner and more efficient solution than other MEC approaches explained in Section 2. It is a standards compatible and does not require the modifications to eNBs, standard LTE interfaces and protocols. Further, traffic classification happens in LTE modem through the LTE/EPC QoS bearer mechanism, which avoids the middlebox deployment for selective traffic offload in the network. ACACIA guarantees short latencies due to closer CI server and avoids the competing traffic between CI traffic and others in centralized GW-Us. In addition, in our ACACIA realization, the GW-Cs are decoupled into a 3GPP control plane and the OpenFlow control plane, thereby allowing them to scale independently.

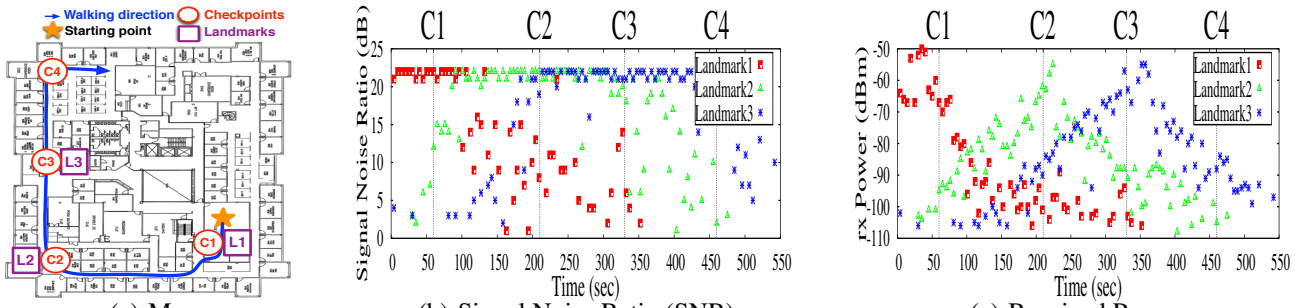
5.5 Context-aware Application Optimization

While all CI applications benefit significantly from ACACIA's abilities explained in the previous sections, certain classes of CI applications, like AR, require further optimization of their application processing to meet their end-to-end latency requirements. As shown in Section 4, this arises from their need to *match* real-time scenes (e.g., faces, objects, etc.) from the client with a large database of apriori tagged images from the environment so as to provide relevant information back to the client in real-time. Towards addressing this latency, ACACIA leverages LTE-direct capabilities to help localize the client in the environment, and uses the client's location information to speed up the CI application.

Note that the requirements on localization accuracy are not very stringent, as the focus is to leverage location information for database pruning. For example, in a retail shop AR application, if the AR application server knows the approximate location of a user, say at the granularity of sections (e.g., clothes vs. shoes, etc.) this will be sufficient.

LTE-direct based Indoor Localization: LTE-direct provides received power (rxPower) level and signal to noise ratio (SNR) with a service discovery message when a subscriber receives it. We conduct a basic experiment to investigate the utility of such auxiliary information since there are no commercialized LTE-direct so far.

Choosing the right parameter: Figure 6(a) presents a map showing the locations of three landmarks (publishers) and the motion path of a subscriber. The landmarks periodically broadcast their distinct service messages over the air using LTE-direct. The subscriber registers with all three services in his CI application and then starts moving from landmark1 to landmark3. Figures 6(b) and 6(c) plot the measured SNR and rxPower values respectively. Figure 6(c) clearly shows the strong correlation between the rxPower and the subscriber's distance to the landmark, with the former peaking as the subscribers gets closer to each landmark. However, the SNR case does not exhibit such strong correlation, owing to the limited dynamic range that is used for decoding (i.e., 25 dB span compared to 50 dB span in rx-



(a) Map (b) Signal Noise Ratio (SNR) (c) Received Power

Figure 6: LTE-direct

Power). Thus, ACACIA leverages rxPower as a more reliable parameter for indoor localization.

Tri-lateration: Once the rxPower values are obtained from the client through LTE-direct, ACACIA employs tri-lateration to localize the client. Tri-lateration requires the location of landmarks (e.g., (x,y) coordinates) and the distance between a subscriber and each of the landmarks as inputs, and returns the estimated coordinate of user's current location as output. While the location of the landmarks is known a-priori, the distance between a subscriber and each of the landmarks needs to be obtained. Here, a linear regression model for the path loss between a user and landmark is constructed for the given environment, which is a one-time overhead. When a subscriber gathers real-time rxPower information (through LTE-direct) from landmarks that it can hear from, the model is used to predict its distance from each of those landmarks. This information is then used by the tri-lateration algorithm [28] to predict the user's coordinates.

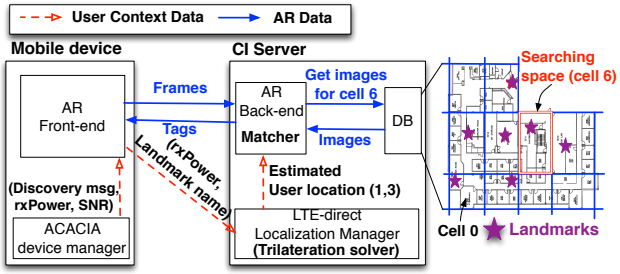


Figure 7: ACACIA CI application Architecture

Leveraging Fine-grained User Context: Figure 7 shows the interaction between the AR application on the user and CI server in the network. The AR front-end application on the user device continuously uploads frames from its camera to the AR back-end application on the CI server running on the mobile edge cloud in mobile network. In addition, whenever the front-end application receives a service discovery message from a landmark, it sends the rxPower and name of the corresponding landmark to the LTE-direct localization manager, also residing on the CI server. The LTE-direct localization manager aggregates such information from the user and uses it to estimate the user's current location by using Tri-lateration, which is then forwarded to the AR back-end application regularly. The AR back-end application maintains a geo-tagged database (Fig. 7), wherein the environment is geographically partitioned into different areas/segments, and the images/scenes from a particular segment (along with their AR-related information) are stored under the same geo-tag. On receiving the location information of the user, the back-end application determines the portion of the database (geo-tag) that covers the user's location. Then, it compares the frames uploaded from the user in real-time with only those from the database that have the identified geo-tag (e.g., cell 6 in Fig. 7). This significantly prunes the database and reduces the time required for matching images, thereby substantially decreasing the response time of the AR application. A noteworthy aspect of using such fine-grained user context (i.e., indoor location), enabled through LTE-direct capabilities of user devices, is the elimination of the need for any additional infrastructure.

6. IMPLEMENTING ACACIA

We have implemented a prototype of ACACIA on two different testbeds; the PhantomNet mobile testbed [26] as well as a private LTE testbed. Both testbeds were equipped with LTE-direct enabled One+ One smartphones (available with Release 12 LTE), Open vSwitch (OVS) [11], Ryu controller [14], OpenEPC-based LTE/EPC software [20] and eNodeBs. The main difference between the two testbeds is the eNodeB hardware. PhantomNet uses a small cell eNodeB from ip.access (scaling up to tens of users), while the private testbed uses a commercial grade eNodeB macrocell (scaling to a hundred users). We also build an AR-based retail application as a representative CI application.

6.1 Network Components

We use OpenEPC software [20] for LTE core network components (e.g., MME, GW-Cs, PCRF, PCEF, HSS and AF) and for processing 3GPP standard control plane. We use OVS as the GW-Us and the Ryu (SDN) controller to control the GW-Us by equipping them with GTP tunneling functions. The Ryu controller dynamically manages GTP

flow rules with GW-Us with IP addresses and TEID from GW-Cs for GTP tunneling.

Dedicated bearer: *(i) MEC Registration Server (MRS) & PCRF:* We refactor the Client-RX module in OpenEPC as the MRS. We configure appropriate *policy rules* (e.g., application ID, QCI) for AR applications in the PCRF databases. *(ii) Control messages for dedicated bearer:* We extend the OpenEPC software to support dedicated bearers. We first extend the PCEF module in OpenEPC to distinguish between the default and dedicated bearer creation requests. Second, we modify each GW-Cs module by setting the IP addresses of MEC GW-Us in Fully Qualified Tunnel Endpoint Identifier (F-TEID) in dedicated bearer request messages to select the correct GW-Us for reaching the MEC server. Finally, we implement the generation of correct control messages (e.g., TFT, ARP), which pass from the MME to eNB, in the MME module.

SDN: *(i) Ryu controller:* We use Ryu OpenFlow controller to control the GW-Us. We extend the Ryu library to support GTP flow management, allowing the generation of GTP en/decapsulate flow rules. *(ii) Open vSwitch:* We use OVS 2.0 and extend it to work as GW-Us by adding GTP encapsulation/decapsulation mechanisms in kernel-space. We extend OVS to processes GTP packets in a kernel-resident fast-path once a packet is matched in the user-space using OpenFlow tables (called slow path in [44]). We use the logical port concept [39], which simplifies header manipulation in the OVS to process GTP packets.

6.2 ACACIA Device Manager

We implement the ACACIA device manager as an Android Service in the smartphone. It defines a *ServiceInfo* class, which implements the *Parcelable* interface [13] to exchange class instances between CI applications and ACACIA device manager. This allows for the expression of user interests (e.g., the landmark name) and discovery messages (landmark name, rxPower, SNR) corresponding to the class instance. To support this communication between the ACACIA device manager and CI applications, ACACIA uses the *Messenger* class [9], which is one of the interprocess communication methods in the Android operating system. In addition, it manages the creation/deletion of dedicated bearer requests to the MRS, when the first interest match is detected by LTE-direct or the user terminates the use of the CI application.

6.3 AR-based Retail Application

We have built a retail GUI application to interface with the ACACIA *device manager* and a prototype of an AR application which consists of a front-end Android application and a back-end object matching server by using OpenCV 2.4.10 [12].

(i) Service Discovery GUI application: The GUI application presents potential landmarks/services (e.g., sections or objects in a retail store) to users and records the user's interest in specific services and contains a localization handler which is connected to the LTE-direct localization manager at the CI server. When it is started, it first binds to the ACACIA device manager and communicates with it using the *ServiceInfo* class. When service discovery happens based on the user's interest, the ACACIA device manager returns the discovered information to the CI application. Then, the localization handler forwards the landmark's name and rxPower to the LTE-direct localization manager at CI server.

(ii) AR front-end: The AR front-end application continuously reads frames from the smartphone's camera. To reduce the latency in uploading frames to the AR back-end server and to save uplink bandwidth, it initially configures the frame size to an appropriate resolution to avoid resize computation overhead and encodes the grayscale frame with JPEG format.

(iii) LTE-direct Localization Manager: We implement a LTE-direct localization manager at the CI server by extending the trilateration solver [16]. When the manager starts, it reads the metadata from a file. This includes information on the number, location and names of landmarks, and the model parameters (α, β) for linear regression to convert rxPower values to distance between a user's location and each landmark. When it receives rxPower and landmark updates from a user, it runs the trilateration algorithm to estimate the user's current location. After estimating the user location, it passes this information to the AR back-end application to help prune the search space in the database.

(iv) AR back-end: The AR back-end server processes uploaded frames from the AR front-end and matches it with objects in its database. When it is started, it first reads the current database stored in YAML format [18]. Our database is populated with 105 objects emulating a retail store and is partitioned based on sections like food, toys and so on. Each object is stored in the database as a set of: object name, an annotated tag, SURF [27] keypoints and descriptors from the image of object. The AR back-end server first decodes an encoded frame from the AR front-end and runs the SURF algorithm to extract SURF feature keypoints and descriptors from it. Using this information, it starts the matching process with objects currently stored in the database. Here, the database is pruned based on the user's location information provided by the LTE-direct localization manager. During the matching procedure, it uses several steps to improve the matching accuracy even though it increases runtime. In each step, it compares the output with the threshold and then decides whether to proceed to the next step or return a "no-match" response. First, it performs the ratio test with the two best matches from a brute-force k-nearest matcher for the two images. Second, it performs the symmetry test to check whether the best matches from two brute-force k-nearest matchers are the same or not. If they are not the same, the best match is discarded. Last, it validates the matches using RANSAC (random sample consensus) to return the correct estimates (matches) as inliers and incorrect one as outliers.

7. EVALUATION

We first evaluate the various components in ACACIA in isolation, followed by the impact of their optimization on application latency.

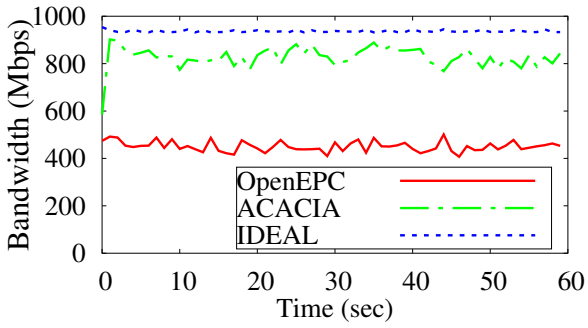
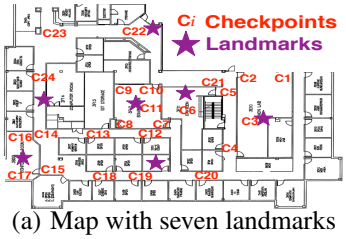


Figure 8: Data plane comparison

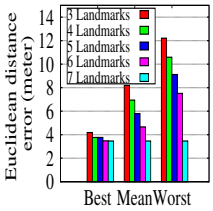
7.1 Micro-benchmarks

Validating ACACIA*'s standard compliance.* We verify ACACIA's standards compliance by testing its inter-operability with several closed source, standards based eNodeBs and UEs. Specifically, we successfully tested ACACIA against the following combinations: (i) eNodeB: ip.access small cell; UE: One+ One (Android 5.0), Nexus 5 (Android 4.4) and Huawei dongle (connected to a Ubuntu 14.01). (ii) eNodeB: commercial macrocell; UE: One+ One (Android 5.0), Nexus 6 (Android 5.0) and Pantech dongle (connected to Ubuntu 12.04). We test each combination with the QoS-based dedicated bearer provisioning for the MEC server in ACACIA. Our results illustrate the standards compliant operation of ACACIA. The time to set up a dedicated bearer in ACACIA depends on the location of the network components in the mobile core network.

SDN Data Plane. We compare our OVS-based GW-U implementation with a (non-split) OpenEPC GW implementation which executes entirely in user-space. (We used OpenEPC release 5 which does not have a functional split GW implementation.) Figure 8 shows the network throughput for ACACIA and the vanilla OpenEPC implementation when the GWs process GTP packets from Iperf TCP test on the mobile client. The figure also shows the maximum achievable throughput in this setup.



(a) Map with seven landmarks



(b) Accuracy according to # of landmarks and positions

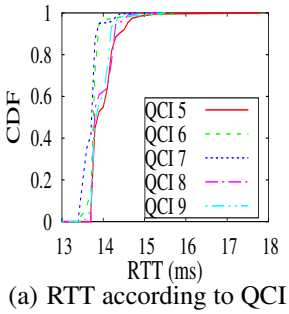
Figure 9: LTE-direct localization accuracy

LTE-direct based Device Localization. We conduct trace-based evaluations to study the accuracy of ACACIA's indoor localization component. The traces were collected from LTE-direct enabled One+ One smartphone devices. Figure 9(a) shows a map indicating the locations of seven landmarks (publishers) and several checkpoints (subscriber locations), which are used for location estimation. We employ Euclidean distance to evaluate localization accuracy error.

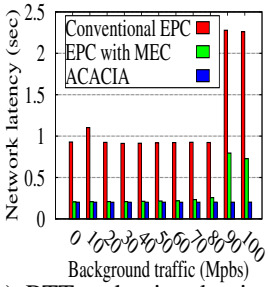
When a subscriber can hear several landmarks, it uses various combinations of signals from the landmarks to estimate the location. Figure 9(b) shows the results. As the number of landmarks increase, the accuracy of localization estimation increases, with seven landmarks showing the best accuracy. In addition, the position of landmarks, especially when the number of landmarks is small, impacts the accuracy of localization. This manifests in the large difference between the best and worst case errors in the case of small number of landmarks, and decreases considerably for a large number of landmarks. Thus, to obtain good accuracy, it is essential to either deploy a large number of randomly placed landmarks, or a small number of well-positioned landmarks.

Our localization error with LTE-direct is around 3 meter on average, which may be high for certain applications. However, note that, ACACIA leverages user's location information to track down the relevant segment of the store, which will in turn be used to prune the search space for computer vision based AR applications. While our current accuracies translate to appreciable application speed-up for AR applications (Section 7.3), ACACIA can benefit from other sophisticated fine-grained localization techniques.

7.2 Impact of Traffic Redirection



(a) RTT according to QCI



(b) RTT reduction by isolation

Figure 10: Redirection effects

RTT Reduction. To measure the RTT between a UE and MEC server, we use a One+ One smartphone as the UE, and LTE ip.access as the eNB with ACACIA's MEC platform (including GW-Us and MEC server). We use different QoS Class Identifier (QCI) [4] values when setting up a dedicated bearer for MEC server and measure RTT by using Ping command. All experiments are conducted with the best LTE signal (full signal bars) in One+ (signal strength: -70 dBm, -74 ASU).

Figure 10(a) shows the end-to-end RTT between UE and the MEC server. 95% of the RTTs are within 15 ms, with RTT between eNB and MEC accounting for a meagre 1.6 ms. Thus, latency is significantly reduced compared to the median latency over current LTE network (e.g., 70 ms) [36].

This makes the case for offloading the computation of CI applications from the smartphone in the presence of MEC.

RTT Reduction from Isolation of CI Traffic. We conduct experiments, where the AR application serves as the MEC/CI traffic, while Iperf is used for generating background traffic at varying rates. We also introduce controlled delays between eNB and SGW and between SGW and PGW to emulate network latencies between UE and MEC server under different core network architectures (conventional and MEC).

Figure 10(b) shows the combined benefit from (i) isolating and redirecting AR traffic (non-split vs. split control-data plane architecture), and (ii) location of the MEC server. (70 ms vs. 13 ms) The conventional core networks employ the non-split architecture, where all traffic from the UE goes through the same S-GW and P-GW. The results indicate that until the network capacity is saturated (around 90 Mbps), the location of the MEC server plays the dominant role in the end-end application latency. However, once the network capacity is saturated, the background traffic has a large impact to increase latency significantly in the conventional architecture. However, ACACIA's ability to isolate/redirect AR traffic helps maintain stable, low latencies for the AR application.

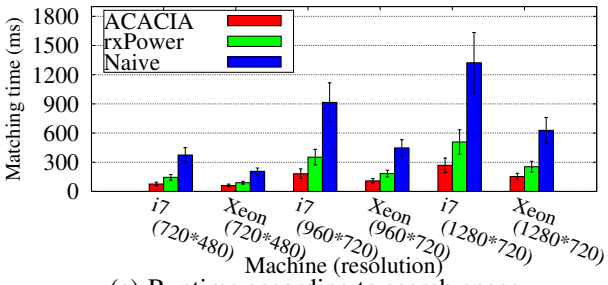
7.3 Impact of Application Optimization

AR front-end resize and compression. Given the volume of data, even when using grayscale images, our AR application uses compression. We measure compression time and the compression ratio. To compress raw grayscale images (1280*720, 960*720 and 720*480) with JPEG 90, on average takes 53ms, 38ms and 23 ms on the One+ One device and shows 5x, 5.8x, and 4.7x size reduction.

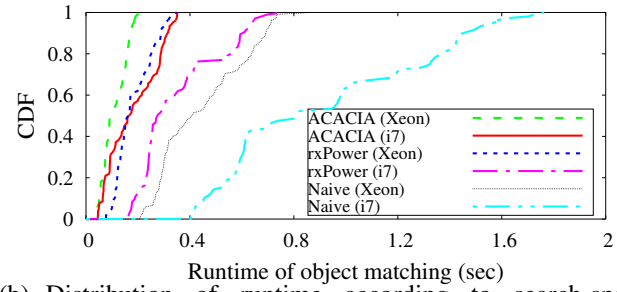
Application Search-space Optimization. We evaluate the reduction in the AR application search-space by leveraging user location information. We consider three schemes. *Naive* approach searches over all objects (in the entire floor) in the database. *rxPower* searches a smaller part of the database (e.g., sections in the floor) based on user's proximity to landmarks from which it received the highest and second-highest rxPower signals. Lastly, ACACIA uses LTE-direct based indoor localization and searches only a much smaller portion of the database (e.g., sub-sections in the floor) based on location information from *LTE-direct localization manager*.

We divide the floor into 5 sections and 21 subsections, as shown in Figure 9(a). Our AR database has 105 objects that are tagged at a sub-section level. We select 24 objects, which are located at checkpoints shown in Figure 9(a) and generate 5 frames per object from the AR application running on One+ One phones at those locations. For the last two approaches, we also measure rxPower from the 7 Landmarks at each of those checkpoints. We use i7 (8 cores) and Xeon processor (32 cores) as the AR server to see impact of processing capability with varying size of frames as inputs.

Figure 11(a) shows the average runtime latency for object matching alone in AR. Clearly, ACACIA incurs the shortest time by being able to reduce its search-space to 2-6 subsections out of 21 with the help of user location information.



(a) Runtime according to search-space

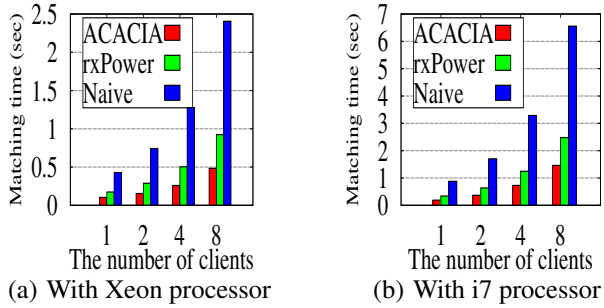


(b) Distribution of runtime according to search-space (960*720)

Figure 11: LTE-direct localization impact

ACACIA shows up to 5.02x and 1.93x reduction in average run-time for matching operations compared to *Naive* and *rxPower* respectively. In high resolution, ACACIA shows better performance than the other approaches since high resolution requires longer processing time for the matching operation. The Xeon processor, with a larger number of cores and OpenCV's support for parallel computations, shows a much better performance. Figure 11(b) shows the CDF of the runtime latency with 960*720 resolution. Without the help of user location information, matching takes over 1 second for some images on i7 (8 cores) processor.

Further, while ACACIA and *Naive* return correct matches in all instances, *rxPower* returns one false negative (object exists in DB, but is not matched/found) in boundary area (C13 in a map) from multiple landmarks as it only searches for objects in the database corresponding to the two landmarks that return the highest rxPower values.



(a) With Xeon processor (b) With i7 processor

Figure 12: Runtime of object matching based on # of users

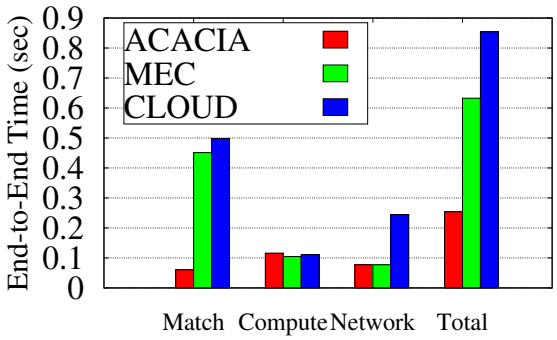


Figure 13: End-to-End comparison

Figure 12(a) and 12(b) show the runtime latency for object matching with 960*720 resolution frames as the number of clients increases with i7 and xeon processor respectively. As the number of clients increase, the runtime is almost doubled. So, ACACIA shows higher reductions compared to the other two cases as the number of clients increase. Experiments at other resolutions (720*480 and 1280*720) show the similar results.

In this experiment, the AR server does not show false negative or false positive since the server has 4 stages which improve accuracy.

7.4 End-to-end Evaluation

We conduct experiments to evaluate end-to-end latency. A One+ One device connects to an LTE ip.access eNodeB and we set the camera input to a 720*480 resolution in the AR application. We evaluate the end-to-end latency by using a subset of the objects in our image database.

In this experiment, we compare ACACIA with (i) an *MEC* implementation which does not have search-space optimization and (ii) *CLOUD* which is conventional EPC and without search-space optimization. Figure 13 shows the results. ACACIA shows a 7.7x reduction for match compared to the other approaches and a 3.15x reduction for network latency compared to *CLOUD*. *Compute* includes the compression of frames on the smartphone, and decompression and SURF computations of keypoints and descriptors in the AR server. There is no significant difference between the different approaches. *MEC* shows a 25% end-to-end reduction compared to *CLOUD*. ACACIA shows a 60% and a 70% end-to-end reduction compared to *MEC* and *CLOUD* respectively.

8. DISCUSSION

A real deployment of ACACIA. A realistic deployment of ACACIA would require Device-to-Device capability in eNodeBs and smartphones, e.g., in our implementation we use LTE-direct. We expect smartphones and eNodeBs to support this functionality in the near future since it is a part of 3gpp release 12 [1]. In addition, since LTE-direct works with smartphones supporting LTE, we expect ACACIA will not require significant additional infrastructure for either the mobile provider or service providers. In the core network side, ACACIA requires split mode GWs (GW-Cs and GW-Us), which differs from current centralized hardware-based GWs. While different GW realization will be needed, our approach does not require changes to the LTE/EPC interface (i.e., does not require 3gpp specification changes). Further, considering current trends [15, 10], we expect software-based split mode GWs will soon be deployed in mobile network. ACACIA does not require change the rest of the mobile core network components (e.g., MME, PCRF, etc.).

We expect that the ACACIA architecture will be able to support a variety of business models. A possible mapping could involve the mobile provider offering an infrastructure service to service providers, including the ACACIA device manager, a device-to-device library and MEC clouds. A service provider could then use this infrastructure to provide different services and/or applications, e.g., retail AR applications and servers as in our prototype.

Other proximity discovery techniques with ACACIA. ACACIA can use other proximity service discovery techniques, e.g., bluetooth ibeacon [7] and WiFi Aware [17]. These technologies also use a pub-sub model and provide similar features like service discovery messages and power level information. To support these technologies, the ACACIA device manager will need to be extended with similar approaches to what we use for LTE-direct.

ACACIA *without proximity service discovery.* It is possible for ACACIA to function without proximity service discovery. In this case ACACIA would need a different trigger to request network connectivity to an appropriate MEC server. For example, launching a specific application might serve as the trigger to activate ACACIA functionality.

9. CONCLUSION

We identified the critical need for edge computing as well its synergistic optimization with both the user and application, towards enabling continuous interactive (CI) applications in mobile networks. To this end, we proposed ACACIA - a service abstraction framework that adopts a holistic end-to-end approach to enabling low latency CI services over mobile networks offered by mobile network providers. ACACIA leverages client context information, namely proximity to landmarks and user interests, along with the SDN/NFV capabilities of the core network, to optimize both network and application processing latencies. The benefits of ACACIA are showcased through a highly-responsive augmented reality based retail service application. Our results show significant improvement over existing mobile networks, and even over a basic mobile edge cloud approach. This validates our holistic optimization strategy. At the same time our results suggest the need for further improvements, likely for both radio access network latencies as well device computational power, before scalable CI applications will be feasible.

Acknowledgements: We would like to thank our shepherd Vijay Sivaraman and our reviewers for their feedback on earlier versions of this paper. This work was initiated when the primary author was an intern at NEC Labs America, and is supported in part by the National Science Foundation under grant numbers 1343713 and 1305384.

10. REFERENCES

[1] 3GPP Release 12. http://www.3gpp.org/specifications/releases/68-release-12.

[2] 3GPP TS 23.401. http://www.3gpp.org/DynaReport/23401.htm.

[3] 3GPP TS 24.008. http://www.3gpp.org/DynaReport/24008.htm.

[4] 3GPP TS 24.008. http://www.3gpp.org/DynaReport/24008.htm.

[5] Creating a Digital 6th Sense with LTE Direct. http://tinyurl.com/jj9tdg4.

[6] Flexible and Elastic User-data Plane with OpenFlow v1.4.0. http://www.openepc.com/specials-and-tooling/cpup-split-with-openflow/.

[7] Getting Started with iBeacon. https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf.

[8] Local IP Access and Selected IP Traffic Offload (LIPA-SIPTO). http://www.3gpp.org/ftp/Specs/html-info/23829.htm.

[9] Messenger Class. http://developer.android.com/reference/android/os/Messenger.html.

[10] NEC Virtualized Evolved Packet Core. http://tinyurl.com/nuefq28.

[11] Open vSwitch. http://openvswitch.org/.

[12] OpenCV. http://opencv.org/.

[13] Parcelable Interface. http://developer.android.com/reference/android/os/Parcelable.html.

[14] Ryu controller, a component-based software-defined networking framework. http://osrg.github.io/ryu/.

[15] The Journey to Packet Core Virtualization. http://resources.alcatel-lucent.com/asset/174234.

[16] Trilateration. https://github.com/lemmingapex/Trilateration.

[17] Wi-Fi Aware. http://www.wi-fi.org/discover-wi-fi/wi-fi-aware.

[18] XML/YAML Persistence. http://tinyurl.com/jzc5fdr.

[19] The tactile internet - itu-t technology watch report. http://www.itu.int/dms_pub/itu-t/oth/23/01/T23010000230001PDFE.pdf, Aug 2014.

[20] OpenEPC. http://www.openepc.com/, 2015.

[21] 5GPPP. 5G Vision - The 5G Infrastructure Public Private Partnership: the next generation of communication networks and services. https://5g-ppp.eu/wp-content/uploads/2015/02/5G-Vision-Brochure-v1.pdf, 2015.

[22] AGARWAL, S., PHILIPOSE, M., AND BAHL, P. Vision: the case for cellular small cells for cloudlets. In *Proceedings of the fifth international workshop on Mobile cloud computing & services* (2014), ACM, pp. 1–5.

[23] ALLIANCE, N. 5g white paper-executive version. *White Paper, December* (2014).

[24] AUCINAS, A., VALLINA-RODRIGUEZ, N., GRUNENBERGER, Y., ERRAMILLI, V., PAPAGIANNAKI, K., CROWCROFT, J., AND WETHERALL, D. Staying online while mobile: The hidden costs. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies* (2013), ACM, pp. 315–320.

[25] BAHL, P., AND PADMANABHAN, V. N. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, Ieee, pp. 775–784.

[26] BANERJEE, A., CHO, J., EIDE, E., DUERIG, J., NGUYEN, B., RICCI, R., VAN DER MERWE, J., WEBB, K., AND WONG, G. Phantomnet: Research infrastructure for mobile networking, cloud computing and software-defined networking. *GetMobile: Mobile Computing and Communications 19*, 2 (2015), 28–33.

[27] BAY, H., TUYTELAARS, T., AND VAN GOOL, L. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*. Springer, 2006, pp. 404–417.

[28] BORENSTEIN, J., EVERETT, H. R., FENG, L., AND WEHE, D. Mobile robot positioning-sensors and techniques. Tech. rep., DTIC Document, 1997.

[29] CHEN, Y., LYMBEROPOULOS, D., LIU, J., AND PRIYANTHA, B. Fm-based indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 169–182.

[30] CHEN, Z., JIANG, L., HU, W., HA, K., AMOS, B., PILLAI, P., HAUPTMANN, A., AND SATYANARAYANAN, M. Early implementation experience with wearable cognitive assistance applications. In *Proceedings of the 2015 Workshop on Wearable Systems and Applications* (New York, NY, USA, 2015), WearSys '15, ACM, pp. 33–38.

[31] CHO, J., NGUYEN, B., BANERJEE, A., RICCI, R., VAN DER MERWE, J., AND WEBB, K. Smore: Software-defined networking mobile offloading architecture. In *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges* (2014), ACM, pp. 21–26.

[32] CUERVO, E., BALASUBRAMANIAN, A., CHO, D.-K., WOLMAN, A., SAROIU, S., CHANDRA, R., AND BAHL, P. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services* (2010), ACM, pp. 49–62.

[33] DONG, W., GE, Z., AND LEE, S. 3g meets the internet: Understanding the performance of hierarchical routing in 3g networks. In *Teletraffic Congress (ITC), 2011 23rd International* (Sept 2011), pp. 15–22.

[34] HA, K., CHEN, Z., HU, W., RICHTER, W., PILLAI, P., AND SATYANARAYANAN, M. Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services* (2014), ACM, pp. 68–81.

[35] HUANG, J., QIAN, F., GERBER, A., MAO, Z. M., SEN, S., AND SPATSCHECK, O. A close examination of performance and power characteristics of 4g lte networks. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (2012), ACM, pp. 225–238.

[36] HUANG, J., QIAN, F., GUO, Y., ZHOU, Y., XU, Q., MAO, Z. M., SEN, S., AND SPATSCHECK, O. An in-depth study of lte: effect of network protocol and application behavior on performance. In *ACM SIGCOMM Computer Communication Review* (2013), ACM, pp. 363–374.

[37] ITU. Tactile Internet. http://www.itu.int/en/ITU-T/techwatch/Pages/tactile-internet.aspx.

[38] JAIN, P., MANWEILER, J., AND ROY CHOUDHURY, R. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International*

Conference on Mobile Systems, Applications, and Services (2015), ACM, pp. 331–344.

[39] KEMPF, J., JOHANSSON, B., PETTERSSON, S., LUNING, H., AND NILSSON, T. Moving the mobile evolved packet core to the cloud. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2012 IEEE 8th International Conference on* (2012), IEEE, pp. 784–791.

[40] KOTARU, M., JOSHI, K., BHARADIA, D., AND KATTI, S. Spotfi: Decimeter level localization using wifi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), ACM, pp. 269–282.

[41] LAU, E.-E.-L., AND CHUNG, W.-Y. Enhanced rssi-based real-time user location tracking system for indoor and outdoor environments. In *Convergence Information Technology, 2007. International Conference on* (2007), IEEE, pp. 1213–1218.

[42] LI, L. E., MAO, Z. M., AND REXFORD, J. Toward software-defined cellular networks. In *Proceedings of the 2012 European Workshop on Software Defined Networking* (Washington, DC, USA, 2012), EWSDN '12, IEEE Computer Society, pp. 7–12.

[43] PATEL, M., NAUGHTON, B., CHAN, C., SPRECHER, N., ABETA, S., NEAL, A., ET AL. Mobile-edge computing introductory technical white paper. *White Paper, Mobile-edge Computing (MEC) industry initiative* (2014).

[44] PFAFF, B., PETTIT, J., AMIDON, K., CASADO, M., KOPONEN, T., AND SHENKER, S. Extending networking into the virtualization layer. In *Hotnets* (2009).

[45] RODRIGUEZ, J., Ed. *Fundamentals of 5G Mobile Networks*. John Wiley & Sons, 2015.

[46] SAID, S. B. H., SAMA, M. R., GUILLOUARD, K., SUCIU, L., SIMON, G., LAGRANGE, X., AND BONNIN, J. M. New control plane in 3gpp lte/epc architecture for on-demand connectivity service. In *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on* (Nov 2013), pp. 205–209.

[47] SATYANARAYANAN, M. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Comp. and Comm. 18*, 4 (Jan. 2015), 19–23.

[48] SATYANARAYANAN, M., BAHL, P., CACERES, R., AND DAVIES, N. The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing 8*, 4 (Oct 2009), 14–23.

[49] STEFANIA, S., ISSAM, T., AND MATTHEW, B. Lte-the umts long term evolution: From theory to practice. *A John Wiley and Sons, Ltd* (2009).

[50] VARSHAVSKY, A., DE LARA, E., HIGHTOWER, J., LAMARCA, A., AND OTSASON, V. Gsm indoor localization. *Pervasive and Mobile Computing 3*, 6 (2007), 698–720.

[51] WANG, K., SHEN, M., CHO, J., BANERJEE, A., VAN DER MERWE, J., AND WEBB, K. Mobiscud: A fast moving personal cloud in the mobile network. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges* (2015), ACM, pp. 19–24.

[52] ZARIFIS, K., FLACH, T., NORI, S., CHOFFNES, D., GOVINDAN, R., KATZ-BASSETT, E., MAO, Z. M., AND WELSH, M. Diagnosing path inflation of mobile client traffic. In *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362* (New York, NY, USA, 2014), PAM 2014, Springer-Verlag New York, Inc., pp. 23–33.