Democratizing Cellular Access with CellBricks

Zhihong Luo
UC Berkeley

Silvery Fu
UC Berkeley

Mark Theis
UC Berkeley

Shaddi Hasan
Virginia Tech & Facebook

Sylvia Ratnasamy
UC Berkeley

Scott Shenker
UC Berkeley & ICSI

ABSTRACT

Markets in which competition thrives are good for both consumers and innovation but, unfortunately, competition is not thriving in the increasingly important cellular market. We propose CellBricks, a novel cellular architecture that lowers the barrier to entry for new operators by enabling users to consume access on-demand from any available cellular operator — small or large, trusted or untrusted. CellBricks achieves this by moving support for mobility and user management (authentication and billing) out of the network and into end hosts. These changes, we believe, bring valuable benefits beyond enabling competition: they lead to a cellular infrastructure that is simpler and more efficient.

We design, build, and evaluate CellBricks, showing that its benefits come at little-to-no cost in performance, with application performance overhead between −1.6% to 3.1% of that achieved by current cellular infrastructure.

CCS CONCEPTS

• **Networks** → **Network architectures**; **Mobile networks**;

KEYWORDS

Cellular architecture, Host-driven mobility, Democratization

ACM Reference Format:
Zhihong Luo, Silvery Fu, Mark Theis, Shaddi Hasan, Sylvia Ratnasamy, and Scott Shenker. 2021. Democratizing Cellular Access with CellBricks. In *ACM SIGCOMM 2021 Conference (SIGCOMM '21), August 23–27, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3452296.3473336

1 INTRODUCTION

Cellular networks play an increasingly important role in the Internet ecosystem: they serve over 5B subscribers, source over 50% of web traffic, and are expected to see dramatic growth due to new applications enabled by 5G, IoT, and edge computing [21, 26]. Given their central role, it is vital that the cellular market be open to innovation and competition. Unfortunately, this is not the case today as the cellular market is dominated by a small number of providers; *e.g.,* 3 carriers account for over 98% of US subscribers [87] and there is mounting concern that the monopolistic nature of this market will negatively impact innovation, pricing, security and, ultimately, the user's experience. [43, 45, 47, 59, 64, 65, 71, 96]

In this paper, we explore an alternate cellular architecture that allows a potentially large number of competing cellular providers to coexist. We start with the observation that, to lower the barrier to entry, we must ensure that providers of *any* scale – small to large – can compete equally within the cellular ecosystem.

We use "scale" to refer to the geographic area that a provider covers. E.g., a small-scale provider might offer coverage over a modest geographic area spanning just one or a few cell towers (*e.g.,* in a campus, mall, city downtown or rural area) while a large-scale provider might offer nation-wide coverage (as today's leading providers do). By "compete equally", we mean that a user should have no reason to discriminate between providers based on the geographic scope of their infrastructure. Instead, we'd like to enable a user to consume cellular service from *any provider that is available at that place and time* with no concern for whether that provider offers coverage in *other* locations. Doing so levels the playing field for all providers: small or large, new or incumbent.

This ideal scenario described is very different from current practice. Today, a user's choice of provider is influenced by the provider's coverage area (*e.g.,* [74]), in addition to price and other factors. Thus, the viability and success of a provider depends on its deployment scale. Building a cellular network is slow and capital intensive; hence expecting new entrants to roll out a large-scale network before they can enter the market significantly raises their barrier to entry. While independent smaller-scale mobile operators do exist, they are often relegated to a secondary role: they largely serve niche markets, rely on roaming agreements with nation-scale mobile networks, or only provide private local networks.

As we'll discuss in §2, the current bias towards large-scale providers is not just an accident of history; rather, it is deeply ingrained in the design choices of the current cellular architecture. To reverse this, we propose a new cellular architecture, CellBricks, that is explicitly designed to accommodate providers of any scale. To achieve this, our architecture departs from current cellular designs in two important aspects. First, it removes the traditional requirement that users have a trusted relationship with the cellular network they are attached to, and instead enables users to consume (and pay for) service on-demand from any infrastructure operator. CellBricks achieves this by moving certain user management functions (*e.g.,* accounting, authentication) out of the cellular infrastructure and refactoring them between the user and an external "broker" service (§3). Secondly, it moves support for mobility from the network to the user device so that a user can experience seamless mobility even if she frequently switches between (potentially smaller-scale) providers, and so that she can do so without relying on complex network support and inter-provider roaming agreements.

SIGCOMM '21, August 23–27, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8383-7/21/08.
https://doi.org/10.1145/3452296.3473336

Although CellBricks was originally motivated by the goal of enabling competition, we find that our design offers two additional benefits: simplification and efficient capacity scaling. By removing in-network support for user management and mobility, the cellular core in CellBricks is significantly simpler than in the current (notoriously complex) cellular architecture (§3). By allowing users to connect to *any* cellular network, CellBricks allows more efficient use of spectrum and infrastructure. This benefit is particularly valuable as 5G requires much denser deployments of base stations than previous generations, which amplifies existing coverage issues.

In summary, the benefits of CellBricks are threefold: (i) lowering the barrier to entry for new providers, (ii) simplifying cellular core infrastructure, and (iii) enabling more efficient use of cellular spectrum and infrastructure.

We design and implement CellBricks as an extension to open-source cellular platforms (Magma [38], srsLTE [48]). We evaluate CellBricks via a combination of experiments on a small-scale testbed and emulation over existing cellular and wide-area networks. We demonstrate that CellBricks is compatible with existing radios, introduces negligible overhead (between -1.61-3.06%) on application performance, and scales to a large number of users under different radio conditions.

In this paper, we focus on the technical feasibility of a cellular architecture that is more open to new entrants. We recognize that our proposal also gives rise to questions around incentives, spectrum, *etc.* We discuss these briefly in §3 but leave an in-depth exploration of such issues to future work.

Roadmap. In §2, we elaborate on why the current cellular architecture fails to accommodate small/mid-scale providers. We present the overall approach, design, implementation, and evaluation of CellBricks in §3, §4, §5, and §6 respectively. We discuss related work and conclude in §7.

Ethics Statement: This work does not raise any ethical issues.

2 BACKGROUND AND MOTIVATION

2.1 The Current Cellular Architecture

Today's cellular networks comprise two main operational components: the Radio Access Networks (RAN) and the cellular "core" (called EPC in LTE, or 5GC in 5G). The RAN includes cell towers (called eNodeBs) that communicate over a radio interface with user equipment (UE). The RAN forwards traffic from UEs to the core which then forwards the traffic onward to the Internet.

The RAN defines how data is encoded and transmitted over the air between the cellular tower and a user device. Our architecture does not modify the RAN and hence we do not discuss it further. The cellular core implements a range of functions related to user authentication, mobility management, traffic classification and prioritization, usage accounting, and so forth. These functions are implemented as hardware or software appliances that may be deployed in a provider's Central Office, an edge data center, or (more recently) the cloud [69]. Importantly, the core serves as the *mobility anchor* for UEs: a UE's IP address, for example, is assigned by the core when the UE connects to the network, and this address remains the same as a UE moves between different towers. We do modify the cellular core in CellBricks and hence elaborate on it briefly (see [10] for details).

The cellular core includes: (i) *Control plane* functions that implement standardized signaling protocols for communication with UEs, (ii) *User plane* functions that implement packet forwarding, including classification and prioritization to enforce QoS levels, counters for accounting, *etc.*, and (iii) *Management plane* functions that maintain subscriber information and perform authentication and policies.

When a user connects to a mobile network it first goes through an "attachment" process which involves using standardized signalling protocols to communicate with the cellular core's control plane. This signalling triggers a series of management functions within the core including (i) authenticating the device, (ii) looking up its subscription plan, (iii) configuring the appropriate user plane functions based on this subscription plan (e.g., configuring rate limits, packet classification rules and priorities), and (iv) creating one or more logical tunnels between the UE and the core to handle traffic. Once this attachment process is complete, the mobile network (radio and core) can process the user's traffic.

This attachment process is not repeated when a user moves from one cell tower to another within the same provider. Instead, the relevant components in the RAN and cellular core will coordinate to ensure that the communication state for that UE – *e.g.,* its tunnel state, QoS rules – are correctly applied to traffic arriving to/from the UE's new tower. This "handover process" is implemented by migrating the tunnels that carry the UE's traffic such that traffic continues to flow through the same elements in the cellular core, including the IP gateway connecting the core to the Internet.

Participants. Traditionally, the two main participants in a cellular network are the user with her UE and the Mobile Network Operator (MNO). The MNO owns and operates cellular infrastructure and also provides user support services such as sales, billing, customer care, marketing, *etc.* The user typically enters into a contractual agreement with one MNO which serves as her default or "home" provider. To provide broad coverage, an MNO may enter into contractual agreements with other MNOs and when a UE "roams" outside the coverage area of its home network, it can consume service from one of these other MNOs, with the roaming UE's traffic typically routed back through its home network.

Mobile *Virtual* Network Operators (MVNOs) are service providers that do not own RAN infrastructure, but instead provide user-facing services (sales, billing, *etc.*) while relying on business agreements with some number of MNOs to provide use of their RAN. Two well-known MVNOs in the US are Google Fi [49] (which uses the T-Mobile and US Cellular networks) and Cricket [101] (which uses the network of its owner, AT&T). In this scenario, the user contracts with an MVNO, and the MVNO in turn contracts with MNOs.

2.2 Limits of Today's Cellular Architecture

We argue that the above cellular architecture is fundamentally at odds with empowering smaller scale providers. There are two key reasons for this which we elaborate on below.

(1) Scaling coverage requires pre-established agreements. A user U can only obtain service from an MNO M with which it has a pre-established contractual agreement. This agreement may be direct (*i.e.,* between U and M) or indirect (*i.e.,* U has an agreement with N and N has an agreement with M that authorizes M to serve U). These agreements are how an MNO provides service to its

users outside its own footprint, and how MVNOs establish their coverage. Similar to peering in wide-area routing, these agreements establish *trust* between two entities based on which they cooperate in authenticating and billing users; *e.g.,* via inter-provider roaming protocols. Unfortunately, this approach scales poorly when we have many smaller providers because these inter-provider agreements are manually established and carry high transaction costs.

Today, the overhead of establishing such agreements is acceptable because each MNO has a large deployment and hence one only needs a small number of agreements to ensure broad coverage – *e.g.,* Google partners with two MNOs for its Fi service. But in an environment with many smaller-scale providers, the number of agreements required to ensure broad coverage would quickly become untenable. [1]

(2) Seamless mobility requires coordination between cell towers. A handover is the process of migrating a UE from one tower to another within one provider's network. Today, this involves cooperation between the towers and cellular core to ensure that a UE maintains its IP address and its active sessions are not disrupted. In current networks, because an MNO has a large deployment, handovers are the norm while crossing provider boundaries is rare and hence users mostly enjoy "seamless" mobility.

However, ensuring seamless mobility in a network of many smaller-scale providers is more challenging: in this case, switching towers will more frequently imply switching providers and preserving a UE's IP address when it crosses provider boundaries would be incredibly complex. Hence, simply carrying over today's cellular design to our context would lead to frequent IP address changes, thereby disrupting TCP connections and degrading the user experience.

In summary, the essential properties of a cellular network – seamless mobility and broad coverage – are difficult to achieve if we simply apply today's design to an infrastructure made up of many providers of any scale. This motivates us to revisit existing designs to eliminate the above problems.

3 OVERVIEW

We propose a new cellular architecture called CellBricks that starts with the MVNO architecture but systematically alters it to avoid the problems discussed in §2.2. CellBricks involves three entities: (i) users and their associated UEs, (ii) brokers, and (iii) cellular access providers of any scale, which we refer to as brick-Telcos (bTelcos).[2] Similar to MNOs, bTelcos own and operate cellular infrastructure (towers, core appliances, etc). Brokers act as intermediaries between users and bTelcos: a user enters into a contractual agreement with a broker and the broker is responsible for representing the user to bTelcos. From a user's perspective, she subscribes to cellular services from her broker and need not be aware of the specific bTelco her device is attached to, which will vary over time.

Up to this point, our architecture might appear identical to that of MVNO services. The key point of departure is that our architecture does *not* require a pre-established agreement between brokers and bTelcos. Thus, bTelcos have no pre-established agreements with users or brokers, and unlike MVNOs, brokers can provide service to their users over any available bTelco infrastructure. To our knowledge, CellBricks is the first architecture that allows both users *and* brokers to dynamically leverage untrusted access providers.

At a high level, we envisage that operation in such a network proceeds as follows.

(1) On-demand authentication and authorization. A UE (denoted U) may request service from a bTelco (T1) when it comes within range of T1. The request identifies the user's broker (B) and T1 forwards the request to B together with parameters describing the terms of service (*e.g.,* QoS and billing options) that T1 can provide. B authenticates both U and T1 (using a protocol outlined in §4.1). If B decides to authorize the request, it informs T1 of this and T1 can start providing cellular access to U. As part of this process, B and T1 might also negotiate additional features such as the need for lawful intercept (as defined in [4, 8, 36]).

(2) Billing. Periodically, U and T1 independently send verifiable and tamper-proof *usage reports* to B. These reports might summarize both the bandwidth used and connection quality that U received. At some later time, T1 bills B based on the usage reports. Compensation is realized in the same manner as other online financial transactions, building on standard techniques for online authentication and payments. Note that we mediate the *process* of payments but do not dictate the actual pricing scheme which is left open to innovation.

(3) Mobility. Later, U may come in the range of bTelco T2 and may wish to switch from T1 to T2. To do so, U simply repeats the same authentication and authorization steps with T2 as it did with T1 and then switches to T2. As a result of the switch, U's IP address may change. To handle this situation without requiring coordination between bTelcos, we employ a host-based mobility procedure (§4.2) that does not disrupt users' application-level sessions.

Realizing CellBricks raises many technical challenges, which we discuss shortly. However, if feasible, we believe that CellBricks offers the following benefits:

(i) No "scale bias": a bTelco can generate revenue by providing service to users within its radio range, irrespective of the scale of its deployment.

(ii) Few pre-established contractual agreements: a bTelco can begin providing service without requiring contractual agreements with users, other bTelcos, or brokers. In our example, neither U nor B have a pre-established relationship with T1. Instead, B and T1 authenticate each other on-demand and (as we'll describe) do so using standard public key cryptography techniques. As we explain later, a bTelco only needs a certified public key and an ability to settle payments; these requirements are standard for online merchants.

(iii) Simplification: CellBricks is a simpler cellular infrastructure to implement and operate. All mobility in CellBricks is host-driven, with bTelcos implementing no particular in-network support for mobility. CellBricks makes no distinction between switching between towers (handovers) or providers (roaming). This eliminates the need for coordination between bTelcos, and even among towers within the same bTelco, while users move about. Further, user authentication is managed by brokers using standard, widely-deployed public key cryptographic techniques. Figure 1 summarizes

[1] With ≥300,000 cell towers in the US [30], if all MNOs deployed 100 towers complete coverage would require 3,000 contracts per MNO *vs.* the few today. This is clearly impractical or at least, raises the barrier to entry.

[2] Brokers are similar to current MVNOs and bTelcos to MNOs. However, we introduce some fundamental differences in their role and functions and hence introduce new terminology to avoid confusion.

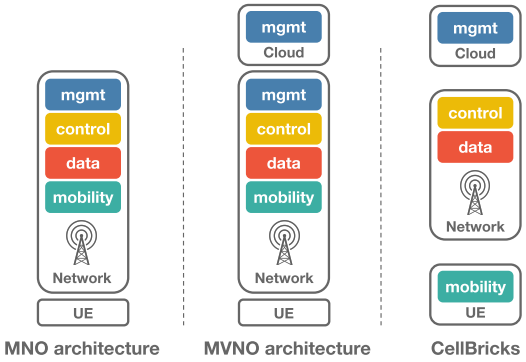

Figure 1: The network here refers to both RAN and cellular core infrastructure. The Cloud contains those portions of the cellular service typically run in a datacenter (e.g., subscriber database). The MVNO arch. requires in-network support for management because they still rely on usage accounting and authorization implemented in the core.

the "division of labor" in each architecture. From an operational perspective, by not requiring a pre-defined trust relationship between brokers and bTelcos, CellBricks removes costly integration and testing procedures commonly required today for establishing roaming and network sharing arrangements among operators.

(iv) Infrastructure efficiency: Rising demands on existing cellular infrastructure are driving *network densification*, with nascent 5G networks requiring larger numbers of smaller cell sites to deliver their promised network capacity. Deploying dedicated radio infrastructure for each provider is capital-intensive and inefficient. In contrast, CellBricks facilitates low-friction infrastructure sharing, allowing any number of brokers to take advantage of a bTelco's deployment. More generally, CellBricks gives greater power of choice to users, brokers, *and* bTelcos: users and brokers can use any bTelco while bTelcos can simultaneously serve multiple brokers. Moreover, this choice can be exerted in a fine-grained manner allowing for a range of policies (*e.g.,* selecting bTelcos based on their historical performance).

(v) Seamless integration of private networks: A growing number of *private* cellular networks serve specific populations or use-cases – *e.g.,* enterprise campus or industrial IoT contexts [15, 35] – and there is interest in integrating these private networks with public cellular networks in a controlled manner [52, 94]. E.g., allowing an employee to seamlessly transition from her MNO to the enterprise's private network. This is not easily achieved with today's cellular architecture but is naturally accommodated in CellBricks.

3.1 Discussion

Although our primary goal is to evaluate the *technical feasibility* of CellBricks, we briefly address a few questions regarding adoption that a reader may have at this point. That said, there are many open questions regarding the market structure and business incentives surrounding CellBricks that are beyond the scope of this paper.

1) What about spectrum? CellBricks requires no changes to the Radio Access Network (RAN) and bTelcos can use any spectrum available to them. Trends in the spectrum regulatory environment are favorable to new entrants, providing them several options for obtaining spectrum. *E.g.,* in the US, the Citizen's Broadband Radio Service (CBRS) [13] provides 150MHz of spectrum in the 3.5GHz band on a dynamically shared basis, allowing wireless operators to deploy networks without costly exclusive spectrum licenses; many commercial deployments of CBRS-based LTE and 5G mobile networks are already underway [50]. Other countries have adopted regulatory constructs that allow new entrants to operate in licensed, but unused, cellular spectrum [12, 29].

It is also feasible that new providers can simply license spectrum from incumbent providers, where this is mutually beneficial [11]; *e.g.,* where the incumbent has no existing or planned infrastructure. Hence existing providers can use new entrants in a franchise-like model, leasing the right to operate in the incumbent's spectrum in certain areas. Our proposal provides a technical foundation for businesses to take advantage of these innovative licensing schemes, while remaining compatible with existing licensing frameworks.

2) What are the incentives for the various stakeholders? For *new bTelcos*, building and operating a cellular network represents an opportunity to participate in a profitable and growing market [18]. CellBricks merely makes this opportunity more accessible to new entrants. Further, because bTelcos are inherently multi-tenant (that is, a single bTelco cell site can support multiple brokers), bTelcos can serve more customers with the same infrastructure, enabling financially profitable operation in a wider range of contexts.

Brokers in CellBricks are equivalent to today's MVNOs or the consumer-facing side of an MNOs, and share similar incentives: the business opportunity of participating in a growing market and the broader benefits of improved user access [38, 76, 90]. As long as demand for cellular service exists, mobile operators will compete to meet that demand. CellBricks simply removes architectural barriers that currently limit this competition. Further, unlike MVNOs today who are at the mercy of their underlying MNOs, CellBricks brokers could easily switch between bTelcos, if necessary, to seek favorable commercial terms.

What about incumbent providers? While it might appear that they have little incentive to embrace our architecture, we speculate that this may not be universally true. Building and operating a radio network is the most capital intensive portion of a mobile network's operation which is exacerbated by 5G's need for dense deployments [19]. With our architecture, existing MNOs can leverage bTelco infrastructure without massive financial investments while still benefiting from their ownership of spectrum (akin to a franchise model). MNOs today already embrace sharing passive infrastructure (*e.g.,* towers) to solve densification and rural expansion [9]; our architecture simply allows them to do so more extensively. Despite these potential benefits, it is still quite likely that incumbent providers would find CellBricks more of a threat to their dominant positions than an opportunity for more efficient infrastructure. Fortunately, CellBricks can be incrementally deployed with no change to, or cooperation from, legacy operators. Specifically, a CellBricks broker could have contractual agreements with some (legacy) MNOs while also leveraging new CellBricks-compatible bTelcos. In this incremental deployment model, MNOs continue to run their legacy protocols and UEs run both legacy and SAP authentication protocols in a dual-stack mode.

Finally, users benefit from bTelcos in the short term through improved coverage and reap the benefits of a more competitive market in the long term.

3) Won't brokers be the new monopoly? We believe this is unlikely to be a concern. First, the barrier to entry for starting a broker is low, requiring no investments in cellular infrastructure or long-term agreements with bTelcos. Instead, the main requirement for a successful broker is the ability to attract users and provide customer support. Many players meet this requirement: content providers, online retailers (*e.g.,* Amazon), traditional retailers (*e.g.,* Costco), credit institutions (*e.g.,* Visa), and non-profit entities such as governments. Second, the broker market is likely to remain competitive because it is easy for users to switch brokers or even sign up with multiple brokers.

4 DESIGN OF CELLBRICKS

To realize CellBricks we must address three questions: (i) How do we ensure secure attachments in the absence of mutual trust between bTelcos and users/brokers?, (ii) How do we minimize disruption to users' connections when switching between bTelcos? and (iii) How do we ensure secure billing and QoS enforcement in the absence of mutual trust between bTelcos and users/brokers? Next, we describe our solution for each of these. For ease of exposition, we use U to represent the UE, B the broker, and T the bTelco.

4.1 Secure Attachments

The foremost challenge is secure attachments, *i.e.,* to ensure secure authentication and authorization in the absence of mutual trust between bTelcos and users/brokers.

Design Rationale. We begin by noting that the process by which a UE attaches to a cellular network can be decomposed into three steps [37]. The first is to establish radio-layer connectivity with the tower, for which we simply reuse existing techniques. The second is authentication which, today, means mutual authentication between a UE and MNO, and is implemented using a shared secret key that is pre-established between the UE (via its SIM card) and the home MNO [22]. The last step to set up the parameters of the service (*e.g.,* QoS settings). For CellBricks, we must revisit the last two steps as we cannot build on the assumption of a trusted relationship between the UE(U)/broker(B) and the bTelco (T). Instead, our requirements for CellBricks are: (i) mutual authentication between U and its B, (ii) mutual authentication between T and B,[3] and (iii) authorization, by which we mean that T must obtain irrefutable proof that B authorized it to service this U; this is required as T need not trust B.

We propose an approach that moves away from shared secrets and instead relies on public-private key cryptography, as is common in online services today, to achieve these goals. We assume all entities – Us, Bs, and Ts – have an associated public key and that B and T keys are signed by a Certificate Authority (CA). Under these assumptions, we design a *secure attachment protocol (SAP)* that achieves our security goals using standard public-key authentication techniques. Our SAP protocol is efficient, requiring only a single round-trip from the U to T to B, and back, compared to two round-trips between U and MNO in the current architecture.

SAP protocol. Briefly, the SAP protocol is invoked when U moves to a different T and involves the following procedures and message exchanges (detailed procedures can be found in the code blocks in Fig.2 and Fig.3):

[3]Since the U trusts B, it is sufficient that the broker authenticates the T and we do not need additional direct authentication between the U and T.

SAP: UE Procedures

Invoked by the UE upon attaching to a new bTelco.

Arguments:

id_x	identifier of entity x
authVec	authentication vector
authReqU	auth. request UE sends to bTelco
authRespU	auth. response received from bTelco
pk_B	broker's public key
sk_U	UE's secret key
n	random nonce generated at UE

Procedures:

1. **Set** authVec = (id_U, id_B, id_T, n)
2. **Encrypt** authVec with pk_B → **authVec***
3. **Sign** authVec* with sk_U → **$sig_{authvec}$**
4. **Send** authReqU = ($sig_{authvec}$, authVec*, id_B) to bTelco

Upon receiving authRespU:

5. **Verify** $sig_{authrespU}$ with pk_B; do 6. if succeed
6. **Decrypt** authRespU with sk_U; use ss, defined in broker procedures, to configure NAS security context

Figure 2: A summary of the steps run at the UE, as part of the secure attachment protocol.

(1) Message from U to T: U crafts a message requesting service from T. The message contains an *authentication vector*, which includes the identifiers of the T, B, and U itself; plus a nonce. An identifier could be the digest of the owner's public key; or the IMSI [31] (if U), IP address, or domain names (if B and T). The nonce is generated as a random string at U and serves to protect against replay attacks. U encrypts this message with B's public key, signs it, and sends it to T. Because T never observes a cleartext identifier for U, it cannot act as an "IMSI catcher" [89].

(2) Message from T to B: T augments the request received from U with the service parameters related to QoS (described later). T signs the augmented request, and forwards it to B.

(3) Message from B to T: When B receives a request, it authenticates both U and T and decides whether to approve the request based on U and T's profiles. If approved, B returns a message that contains two signed and encrypted (sub-)responses: (i) *authRespT* that includes identifiers of U and T, a shared secret *ss*, and QoS parameters (the last two are described next); (ii) *authRespU* that includes identifiers of U and T, *ss*, and the U-generated nonce. On receiving B's message, T authenticates B by validating B's signature in *authRespT*; this response serves as the authorization for T to serve U. Then, T replies to U with *authRespU*.

(4) Message from T to U: On receiving *authRespU*, U authenticates B by validating B's signature in *authRespU*; this response helps U confirm that its access to T is now authorized.

As a summary, U is responsible for identifying itself to T and B; T forwards authentication messages between U and B, and B authenticates and authorizes both U and T. Finally, both U and T will use *ss* in the responses to set up their security contexts following the existing security procedures. Note that SAP's security context is identical to that used in EPS which includes keys for protecting AS and NAS messages, as well as NAS counters and identifiers. One could refer to [67] for details. Briefly, the shared secret *ss* is used as the master key (also known as KASME [103]) in the security

SAP: bTelco Procedures

Invoked by the bTelco upon getting a SAP request.

Arguments:

authReqU, authRespU, pk_B

authReqT	auth. request bTelco sends to broker
authRespT	auth. response from broker to bTelco
qosCap	QoS capability definitions
sk_T	bTelco's secret key

Procedures:

Upon receiving authReqU from UE:

1. **Lookup** broker (e.g. its IP) with $authReq.id_B$
2. **Set** authReqT = (authReqU, qosCap)
3. **Sign** authReqT with $sk_T \rightarrow$ **$sig_{authreqT}$**
4. **Send** ($sig_{authreqT}$, authReqT) to broker

Upon receiving ($sig_{...}$, authRespU, $sig_{...}$, authRespT) from broker:

5. **Verify** $sig_{authrespT}$ with pk_B; do 6. if succeed
6. **Reply** ($sig_{authrespU}$, authRespU) to UE; **Decrypt** authRespT; set UE profile and NAS security context

SAP: Broker Procedures

Invoked by the broker upon serving a SAP request.

Arguments:

authReqT, authRespT, authVec*

sk_B	broker's secret key
ss	shared secret used by UE and bTelco
qosInfo	QoS information for the UE session

Procedures:

Upon receiving ($sig_{authreqT}$, authReqT) from bTelco:

1. **Verify** $sig_{authreqT}$ with pk_T and do 2. if succeed
2. **Decrypt** authVec* with sk_B
3. **Lookup** id_U and id_T; **Verify** $sig_{authvec}$ with **pk_U** and id_T matches **pk_T**; do 4. if both succeed
4. **Encrypt** (id_B, id_T, ss, n) with $pk_U \rightarrow$ authRespU; (id_B, ss, qosInfo) with $pk_T \rightarrow$ authRespT
5. **Sign** authResp $\rightarrow$ **$sig_{...}$**, authRespT $\rightarrow$ **$sig_{...}$** with sk_B
6. **Reply** ($sig_{...}$, authRespU, $sig_{...}$, authRespT) to bTelco

Figure 3: A summary of the SAP procedures that run at the bTelcos (top) and the brokers (bottom).

mode control (SMC) [7] procedures to derive keys for ciphering and integrity protection of their AS and NAS messages [6], which we reuse otherwise unmodified from today's standard. After the security context is established, we reuse unmodified session establishment procedures to provide U with access to public networks, during which T assigns an IP address to U.

In addition, SAP is also used to communicate various service parameters such as *e.g.,* QoS settings and whether lawful interception is to be invoked [4, 5]. While today's network implements both the policy and mechanism for these features, CellBricks decouples these, with policy decisions made by B and communicated to T which implements them. This is done by augmenting the authentication protocol to include QoS parameters (*i.e.,* qosCap and qosInfo in Fig.3, we omit the other possible policy parameters for brevity) and other service parameters, and the set of parameters can also be dynamically updated. Specifically, we have T inform B what QoS options (qosCap) it can enforce and that B can then send specific parameter values (qosInfo). Doing so requires a standardized approach to expressing these parameters, *e.g.,* for QoS, we propose to adopt the existing 3GPP definitions of QoS parameters [5]. We will describe how a broker ensures that bTelcos correctly enforce the QoS in §4.3.

SAP is designed such that U only requires *a small set of static parameters* for attachment; specifically, U's key pairs and B's public key. This state can be embedded in the U's SIM card, in exactly the same way that the shared secret used for authentication today is embedded and distributed to users. Note that U's public keys are used only for interactions with B, who issues U's key pair in the first place, hence no certificates are needed for U's public keys. Moreover, B can revoke U's public key by simply invalidating the key in its database. For T and B, we assume their public keys and corresponding certificates are distributed and maintained using standard PKI techniques, akin to existing Internet services.

Lastly, in an effort to understand what security properties SAP can offer and what it cannot, we discuss the security of SAP in the context of several common attacks in our technical report [24].

4.2 Seamless Mobility

Since CellBricks enables a potentially large number of smaller-scale bTelcos, switching between towers often implies switching bTelcos: How do we minimize the disruption to the user's connections in the face of these frequent switches?

Design rationale. In today's networks, most mobility events involve *handovers* in which a user switches from one tower to another within the same provider's network and support for such handovers is embedded in the network; *i.e.,* towers coordinate using signaling protocols to ensure that a user's traffic is routed through a consistent gateway in the cellular core. This ensures that the device retains its IP address as it moves between towers and hence that its active TCP connections are not disrupted.

This network-driven coordination is difficult to implement in CellBricks as it requires cooperation and interoperability between bTelcos, adding both operational and technical complexity which in turn raises the barriers to entry for a new bTelco. Instead, we propose a host-driven approach that essentially eliminates the concept of a handover: a user simply detaches from one cell tower and independently attaches to a new tower (run by the same or different bTelco) via the SAP protocol. This approach is simple as it requires no network support or coordination between towers.[4] Similar to today's network-driven UE-assisted handover, where UEs conduct performance measurements to help the network make handover decisions, our UE-driven handover can benefit from network assistance too. For instance, UE-driven handover can perform smarter cell selection based on the list of neighbor cells learned from the network. We believe such UE-driven, network-assisted handover is feasible and promising as demonstrated in today's Wi-Fi roaming [14, 84, 91] and Wi-Fi-cellular handover [70, 92, 104].

Handling IP changes. As a result of our host-driven approach to mobility, U's IP address may change as it switches towers which raises an important question: how do we avoid disrupting U's connections? We observe that the reason the current handover process

[4] While we do not preclude coordination across towers in a *single* provider, we're intrigued by the possibility of removing this complexity from the network entirely and hence evaluate this extreme design point in §6.

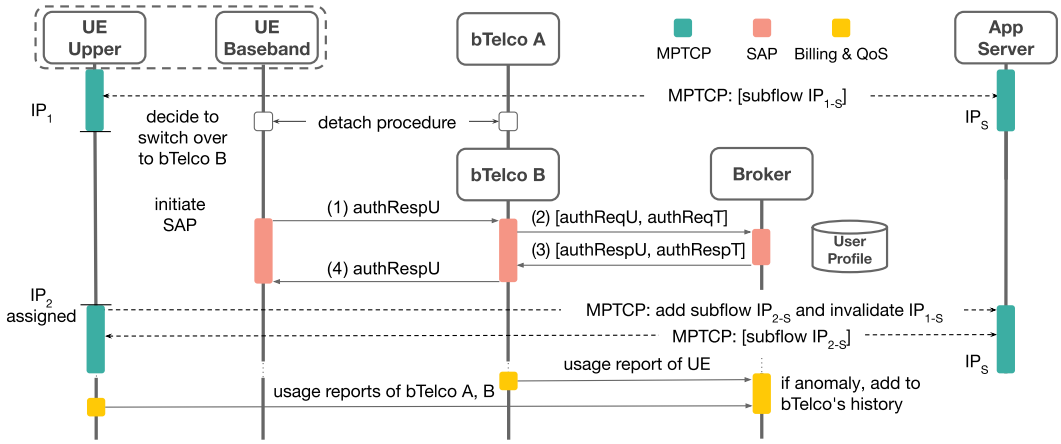

Figure 4: An overview of the attachment, mobility, and billing/QoS process in CellBricks depicting the key events and message exchanges happen during the SAP, MPTCP, and billing protocol over time. Note that authReqU/T and authRespU/T are defined in Fig.2 and Fig.3.

must retain U's IP address is to avoid breaking U's open transport-layer (typically TCP) connections. But is Layer 3 (L3) the right network layer at which to address this problem? For CellBricks, we argue not. A host's IP address today reflects its location in the Internet's routing topology and the administrative domain to which it belongs; we want to preserve this property. Instead, we want higher layers – specifically, the transport layer – to be capable of adapting to changes in the endpoint's IP address.

Fortunately, new transport protocols such as MPTCP [102] and QUIC [60] already provide such support, though motivated by different use-cases than ours.[5] These protocols have explicit connection identifiers within their L4 header and use IP addresses only for packet delivery. This separation allows the use of multiple underlying IP addresses for the same connection. Simply adopting these protocols, which are already standardized and widely deployed [42, 57], solves our problem with no additional change and no network support.

For instance, MPTCP introduces the notion of a subflow – a flow of TCP segments operating over an individual path. A single MPTCP connection can operate with multiple subflows that can be dynamically added and removed over the lifetime of the connection [102]. Fig.4 illustrates how a MPTCP connection with a single active subflow reacts to IP changes in the context of bTelco detachment and attachment. In brief: (i) At the end of the detachment procedure, the baseband processor at U deletes the radio bearer (used at bTelco A) and informs the OS kernel that the IP address of the UE's network interface is no longer valid (as a result, the interface's IP is typically set to `0.0.0.0`); then (ii) the MPTCP stack at the UE is notified about the address invalidation and will watch for a new address until reaching a predefined timeout (default to 60s) while the existing subflow (IP_{1-S}) stays inactive. If the timeout is reached, the MPTCP connection will be torn down. (iii) Once the UE securely attaches to the bTelco B, a new "bearer" (a tunnel connecting the UE to a gateway to the Internet), is created using the UE's new IP address. Once the network interface regains a new address, the UE's MPTCP stack uses its new source IP and initiates a three-way handshake to create a new subflow (IP_{2-S}); the UE also informs the server side of the connection to remove the previous subflow (IP_{1-S}) via the REMOVE_ADDR option. Once IP_{2-S} is established, the UE and server can resume exchanging packets over the MPTCP connection.

We believe this *host-based* approach is the right long-term solution: architecturally, it respects Internet design principles and layering, can be deployed with no support from the network, is supported by major operating systems (*e.g.,* Windows, Linux, X, Android, iOS) and is seeing growing deployment [20, 98] including for multi-access in 5G [34]. There are solutions besides MPTCP and QUIC – *e.g.,* HIP [66] and SCTP [88] – that can also handle IP changes. We leave an exploration of these options to future work.

Finally, although host-driven mobility is our preferred approach, it requires support at both endpoints. To support incremental deployment while these protocols are ubiquitously deployed, our strategy (used in our prototype) is simply to fallback to TCP when MPTCP is unavailable and rely on the application and/or L7 protocols (*e.g.,* SIP re-invite [83]; HTTP range headers [40]) to efficiently restart failed connections.

4.3 Verifiable Billing and QoS

The last piece of the puzzle is: how do we ensure secure and verifiable billing and QoS enforcement in the absence of mutual trust among the UE, broker, and bTelco?

Design rationale. We focus on ensuring accurate *accounting*, by which we mean the ability to obtain an accurate record of the network resources a UE consumed at a bTelco. Such accounting is the foundation on which billing between the various parties – T-to-B, and B-to-U – can be implemented and we leave the question of pricing open to innovation. This approach matches today's architecture where accounting within the cellular core supports a range of service plans; *e.g.,* based on flat-rate pricing, usage caps, *etc.*

In current networks, accounting is based on measuring traffic statistics in the "packet gateway" of the cellular core (PGW in 4G, UPF in 5G). Even though operators could miscount or over-bill, users generally trust their results because of their reputations as large carriers and their contractual agreements. With untrusted bTelcos, we instead need an accounting protocol that is *tamper-resistant* and *verifiable*.

[5] MPTCP's original goal was to improve the performance of a single connection by leveraging the multiple paths between a source and destination.

Verifiable Billing and QoS

Invoked by the broker at the end of each billing cycle.

Arguments:

$tr_{\{U,T\}}$	traffic report from UE or bTelco; it includes usage (e.g. dl_bytes) and QoS (e.g. dl_loss) info
thresh	anomaly threshold
dsp	discrepancy of reported usage

Procedures:

Once received both tr_U and tr_T:

1. **Set** thresh = (tr_U.qos.dl_loss + ϵ) × tr_T.usage.dl_bytes
2. **Set** dsp = tr_T.usage.dl_bytes - tr_U.usage.dl_bytes
3. **If** dsp > thresh, **increment** the # of anomaly at the bTelco

Upon receiving a reputation query for bTelco T:

4. **Report** the # of anomaly occurred at T

Figure 5: A summary of the steps run at the broker to enable verifiable billing and QoS.

We assume that bTelcos are not malicious in the sense of wanting to disrupt a user's service but that they could be motivated to lie about resource consumption if it increases their revenue *and* if they believe they can lie without being detected. This latter seems reasonable given the capital costs of setting up a cellular tower: to see a profit, a bTelco will need to remain operational for some period of time but if it is suspected of cheating then a broker might simply choose not to use the bTelco. The same may be said of UEs and their Brokers. We call this a "dishonest but not malicious" threat model. These assumptions are analogous to many customers and retail businesses - large and small - in the real world.

Verifiable billing and QoS. At a high level, our approach is to have T and U independently measure the traffic volume and QoS for U's session. Then, we have them periodically send encrypted and signed traffic reports that contain those measurements to U's broker B. A traffic report includes the following information: *(i) Session identifier* which uniquely identifies a session between U and T; *(ii) Relative timestamp* within the session, which is used for B to align U's and T's reports; *(iii) Usage metrics* that accounts for traffic volume consumed at the uplink (UL) and downlink (DL) in bytes; *(iv) Duration* for phone call and events such as SMS messages; *(v) QoS metrics*, as defined by the 3GPP standard, including the average bit rates, packet loss, and packet delay etc., reported separately for both the DL and UL [1].

The challenge is that T's traffic reports are untrusted, and T might have an incentive to inflate the usage values. Therefore, U will independently measure its own traffic statistics and periodically sends a traffic report R_u to B. Since U may have an incentive to *deflate* the usage values, we ensure that R_u cannot be tampered with by U and hence B can trust R_u. We discuss this further below.

This simple approach sets up the right incentive structure: dishonest reporting by either Us or Ts will manifest as a discrepancy between their reports and, while small discrepancies are expected and tolerated, a large or persistent discrepancy will be viewed as anomalous. We assume B and T store a history of report summaries and anomalies and hence, over time, build up a *reputation system* based on which either party can decline to cooperate - U/B can switch to using a different T, while T can decline to serve U/B. We elaborate on design considerations for this reputation system in what follows.

Reputation System We focus on addressing the use of a reputation system to enforce correct resource accounting.[6] To do so, B maintains: (i) a per-bTelco aggregate *reputation score* and (ii) a list of its own users that are suspected to have tampered with their device. We expect the latter to be a small number, because implementations at the UE side are embedded into baseband firmware and hard to tamper afterwards, which allows the broker to carefully review the firmware implementation and ensure its correctness. Likewise, T maintains an aggregate reputation score per broker.

Reputation scores can be derived from the UE's and the bTelco's traffic reports in a manner that is left open to innovation. Here we present one design based on simple heuristics but imagine that in practice brokers can implement more sophisticated strategies. Fig.5 describes this design where B compares the discrepancy of reported DL usage against a *threshold value* that is calculated based on the UE's reported DL loss rate and a fixed tolerance ratio ϵ. (*e.g.,* derived from the acceptable link loss rate). When the discrepancy is greater than the threshold, B considers this as an anomaly and records this incident (a "mismatch"). B then derives the reputation score based on the number of mismatches, weighted by the degree of mismatch. We leave an exploration of exactly how to do this weighting to future work. Note that for T, it discovers discrepancies with U's reports either indirectly from the broker's final settlement or directly by requesting U to also send it a copy of the traffic report.

Given the reputation scores, B can decide whether to authorize an attachment according to the reputation score of the bTelco as well as whether the user is on the suspect list. Likewise, T can decide not to service any users belonging to a broker with a poor aggregate score. The exact policy that each broker and bTelco will adopt is open to innovation.

In terms of security properties, CellBricks's reputation system is generally vulnerable to the same failure modes as any reputation based systems, and at the same time benefits from existing countermeasures. Interested readers could refer to our technical report for more discussion on this issue [24].

Our final requirement is to ensure that the traffic report from the UE cannot be tampered with by the user (since the user may have an incentive to undercount, just as the bTelco has an incentive to overcount). A tamper-resistant accounting protocol at the UE has two components: a secure measurement function that accurately records a user's traffic statistics, and a protocol that safely communicates these statistics from the UE to the broker.

We propose to embed the measurement function in the UE's baseband, which today implements all cellular functions and is assumed to be tamper-resilient [61].[7] To ensure these measurements cannot be tampered with once extracted from the baseband, we propose to sign and encrypt the measurement report on the baseband.

The above changes can be implemented as just a *firmware* upgrade to the existing UE baseband and introduces little overhead because: (i) usage and QoS metrics are already available in today's

[6] We believe that one could extend the reputation system to enforce QoS but leave a design of this to future work.

[7] Some tools [62, 79] can read (but not write) the modem's internal state, and some attacks can steal security credentials and sensitive information [55, 105]. However, none of these can overwrite modem statistics.

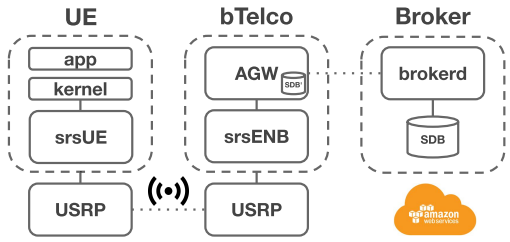

Figure 6: An overview of our testbed. AGW: access gateway; SDB: SubscriberDB.

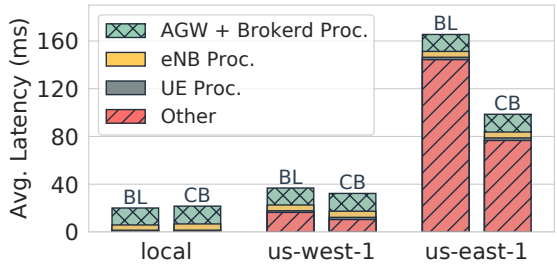

Figure 7: Latency breakdown by module in the Magma baseline (BL) and CellBricks (CB) during an attachment request.

baseband processor (*e.g.,* PDCP counters for bytes sent/received and RLC metrics for packet loss [2]), (ii) today's baseband processor already implements encryption. Moreover, these operations are only performed once every reporting cycle, which we anticipate will be on the order of many seconds or even minutes.

5 PROTOTYPE IMPLEMENTATION

We prototype CellBricks using existing open-source cellular platforms, given changing baseband firmware requires certificates from baseband and/or device manufacturers. As described in Fig.6, the prototype includes four components: UE(s), the base station (eNodeB), the cellular core, and our broker implementation (brokerd). Our testbed has two x86 machines: one acts as UE and the other as bTelco (eNodeB + EPC). We connect each machine to an SDR device (USRP B205-mini [95]) which provides radio connectivity between the two machines. On each machine, we run an open-source LTE suite (srsLTE [48]) with the UE machine runs the srsUE stack and the eNodeB machine runs the srsENB stack. We extend srsUE with the UE-side changes mentioned in §4. We install an MPTCP-enabled Linux kernel (v4.19) on the machines and run apps in docker containers. The containers use the network stack of the host machine, allowing applications to run unmodified because MPTCP is largely backward compatible with the existing socket API.

For the cellular core and broker, we build on Facebook's Magma [38][8], an open-source software platform that serves as an extensible mobile core network solution. The two main components we extend are the access gateway (AGW) and the orchestrator (Orc8r). The AGW implements the core network (EPC), and the Orc8r implements a cloud service that configures and monitors the AGWs. We extend AGW to support our secure attachment protocol: we define new NAS messages [6] and handlers and implement these as extensions to Magma's AGW and srsUE. Finally, we implement the broker service (called *brokerd*) as part of Magma's Orc8r component deployed on AWS. Brokerd maintains a database of subscriber profiles (called SubscriberDB) and implements the secure attachment protocol, processing authentication requests from bTelcos. Our prototype does not include the reputation system. We defer its implementation and evaluation to larger-scale CellBricks deployment in future.

In summary, we introduce two new protocols to the existing 3GPP standard (the SAP and our accounting protocol), as well as a new dependency on end host transport protocols (*e.g.,* MPTCP) to support mobility. We modify only the UE and core network to achieve this, which importantly allows reuse of unmodified commercially available cellular base station equipment. In total, our extensions to Magma includes 2,493 LoC in the AGW (in C) and 263 LoC in Orc8r (in Python); we add 940 LoC (in C) to the srsUE.

[8]CellBricks's implementations and evaluation results can be found in [23].

6 EVALUATION

We evaluate CellBricks using a combination of two approaches: (i) benchmarks of our prototype testbed (§5), and (ii) emulation over existing cellular and wide-area networks. The former is limited in scale but validates end-to-end correctness and demonstrates compatibility with existing radios, user devices, and base stations. The latter allows us to answer what-if questions regarding application performance under real-world conditions with real applications.

CellBricks raises two main performance questions: (i) how much overhead does our attachment protocol (that includes three parties) introduce compared to the existing cellular protocol (that includes two parties)? and (ii) does CellBricks's host-based approach to mobility impact application performance relative to today's in-network approach? We evaluate these questions in what follows.

6.1 Prototype Performance

Methodology: We measure the end-to-end latency due to our attachment protocol, measured from when the UE issues an attachment request to when attachment completes. Note that we do not take into account the potential cell selection time, as the target cell information is usually known prior to attachments. From the E2E latency, we first remove the time spent in the RRC (radio) and lower layers since this value depends largely on the choice of radio hardware and protocol implementation and these components remain unmodified in CellBricks. Moreover, because we use software-based implementations of the RRC and and lower layers, the latency through the radio stack is higher than it would be in a typical hardware-based RRC ($\approx 130ms$) which could mask other system overheads that we introduce. To understand where the remaining time is spent, we instrumented the relevant components of our prototype – Access Gateway (AGW), SubscriberDB, Brokerd, eNodeB and UE – to measure the processing delay at each.

In our experiments, the UE, eNodeB, and AGW are always located in our local testbed and we run experiments with the subscriber database (SubscriberDB) and Brokerd either hosted on Amazon EC2 [16] or our local testbed. Running in the cloud matches current deployment practice in which certain core network components are run in the carrier's datacenter or on public clouds. For each setup, we repeat the same attachment request using both unmodified Magma and Magma with our modifications to implement CellBricks. We repeat each test 100 times and report average performance.

Results: Fig.7 shows the attachment latency (after removing the time spent in the RRC and lower layers) for three different placements of the SubscriberDB and Brokerd for both unmodified Magma (our baseline, denoted BL in the figure) and Magma with our modifications to implement CellBricks (denoted CB). In each case, we also show the breakdown of latency at each module. The portion labeled "Other" is simply the leftover latency once we remove the time spent in each of the above modules from the attachment latency; this leftover time is dominated by the latency between the AGW and the SubscriberDB/Brokerd.

Looking first at the overall latency, we see that in all cases, the attachment time with Magma-CellBricks is comparable to Magma-unmodified. In fact, the attachment latency with Magma-CellBricks is 14.0% smaller than the unmodified Magma when we run the SubscriberDB and Brokerd in the us-west-1 region (31.68ms vs. 36.85ms) and 40.8% smaller when in us-east-1 (98.62ms vs. 166.48ms). This is because the standard S6A attachment procedure [3] in our baseline involves two round-trips between the AGW and the SubscriberDB (the standard involves two requests, an Authentication Information Request and an Update Location Request (ULR) made to the Subscriber DB) whereas in Magma-CellBricks a bTelco does not send the second (ULR) request (see §4).

Looking now at the breakdown in latency, we see that in the local setup, the attachment request processing at the AGW and Brokerd accounts for about 70% of the total request latency ($\approx 20ms$), before and after modifying Magma alike. This confirms that our changes to Magma such as adding brokerd and crypto operations introduce negligible performance overhead ($\approx 2ms$) across the modules. When the SubscriberDB and Brokered are in the cloud, the total request latency is dominated by the network latency between the AGW and cloud as the "Other" bars indicate. Our latencies for us-west-1 are lower than us-east-1 simply because the former is geographically closer to our local testbed.

In summary, CellBricks adds little overhead to the attachment process and – by eliminating one round-trip between the AGW and cloud – can even improve attachment latencies compared to existing cellular implementations. However, as mentioned earlier, CellBricks undergoes attachments more frequently than in current cellular networks; we evaluate the impact of these more frequent attachments next.

6.2 Emulation over the Internet

With CellBricks's host-driven approach to mobility, switching towers can involve re-authenticating and initiating new transport-layer "subflows," each of which could impact end-to-end performance. This impact depends on multiple factors such as the frequency of handovers, packet loss, *etc.* To capture these in a realistic manner, we *emulate* CellBricks over the T-Mobile network in our urban region. This allows us to capture real-world conditions such as the density of tower deployment, devices on the move, real-time background traffic, handover patterns, and application behavior. Prior work has shown that MPTCP performs well in the face of soft handovers between Wi-Fi and cellular providers [25, 32, 33, 70, 80]. In this paper, we instead evaluate MPTCP when migrating across cellular access providers (bTelcos) and do so under the extreme scenario in which each provider operates only a single tower. To our knowledge, this is the first comprehensive evaluation of host-driven mobility in wide-area cellular networks.

Overview. In today's cellular infrastructure, a UE typically retains its IP address when it switches towers. Hence, the crux of our approach is that we will *emulate* an IP address change each time a handover occurs. At a high level, our emulation works as follows. First, we use a real UE (running Linux, MPTCP, and applications) that connects to a real cellular network (T-Mobile) and we exploit low-level APIs on the UE's Qualcomm chipset to detect when a handover occurs. Whenever a handover is detected, we emulate an IP address change to the container running our test applications. This involves invalidating the old address and creating a new one. We introduce a delay d between invalidation and when the new address is available - d is a parameter that we can tune to model the overhead of authentication/attachment (as measured in the previous section). This change in IP address will in turn trigger MPTCP to take appropriate action (*e.g.,* creating a new subflow). Finally, we use GRE tunneling to carry packets with the emulated IP address between the client and server. Tunneling is used only for emulating IP changes in today's infrastructure, and will not be needed in a real CellBricks deployment. Throughout the above process, we run an application and measure its performance. We leave the evaluation of CellBricks on iOS/Andoird apps, on protocols other than MPTCP, and on soft (make-before-break) handovers to future work.

Methodology. We now describe this emulation process.

(i) *Equipment.* Our UE consists of a ZTE MF820B LTE USB Modem (with Qualcomm chipset) [107] and a laptop (Ubuntu 18.04 with 4.19 MPTCP-enabled kernel) that runs application client code. The modem uses an unlimited prepaid SIM card and connects to the laptop via a USB port. We run the server side of the applications on Amazon EC2 (region: `us-west-1` with `c5.xlarge` instances). We use two UEs and two EC2 server VMs for each run: one UE-VM pair runs MPTCP while the other runs regular TCP. The TCP pair is our baseline that represents current infrastructure (see iv), while the MPTCP pair is subject to our emulation of CellBricks.

(ii) *Detecting handover.* Qualcomm chipsets expose a diagnostics interface via which we read baseband messages, *e.g.,* using the tool QCSuper [78]. We pass the RRC messages to Wireshark's in-memory capture to extract handover events. On detecting the start of a handover, we emulate an UE IP change as described below.

(iii) *Emulating IP change.* We first note that we only emulate IP changes as below for the UE-VM pair running MPTCP; our baseline UE-VM pair run unmodified TCP and we do not change their IP address across handovers. Applications run inside docker containers on the UE and VM. To emulate IP changes, on detecting a handover at the UE, a proxy program invokes `ifconfig` to set the container's (virtual) network interface to `0.0.0.0` – this emulates address invalidation when switching bTelcos (§4.2). The proxy then waits for a time d before re-assigning the interface a new IP address. The interval d represents the overhead of attachments in CellBricks. Unless noted otherwise, we set $d = 31.68ms$, based on our prototype benchmarks in §6.1 (us-west-1 test).

Interestingly, our early experiments revealed that the mainline MPTCP implementation limits how fast the MPTCP stack can react to an IP address change. It does so by introducing a wait period between when it first detects an address change and when it takes any corrective action (*e.g.,* starting a new subflow). This wait period is

Application		MTTHO		Ping: p50		iPerf: Avg. Throughput		VoIP: MOS		Video: Avg. Quality Level		Web: Avg. Load Time	
Unit		second		millisecond		mbps		1-5/excellent		level (0-5)		second	
Route / Time of Run		D	N	D	N	D	N	D	N	D	N	D	N
Suburb	MNO	73.50	65.60	45.95	46.71	1.25	17.27	4.38	4.38	1.96	4.91	4.78	1.81
	CellBricks					1.20	16.85	4.35	4.33	1.98	4.91	4.96	1.76
Downtown	MNO	68.16	50.60	49.60	48.53	1.14	16.54	4.30	4.33	2.03	4.94	5.12	1.89
	CellBricks					1.11	15.41	4.25	4.32	1.97	4.94	5.22	1.89
Highway	MNO	44.72	25.50	49.48	48.38	1.10	11.38	4.34	4.34	1.95	4.89	5.05	1.86
	CellBricks					1.11	12.42	4.27	4.30	1.97	4.90	5.18	1.80
Overall Perf. Slowdown		-	-	-	-	2.06%	3.06%	1.15%	0.92%	0.51%	-0.20%	2.60%	-1.61%

Table 1: Comparisons of application performance in CellBricks vs. today's cellular networks (MNO).

hard-coded to 500 ms (see `address_worker` [93]) which effectively masks the overhead d that CellBricks introduces. This wait period is an optimization to avoid flapping and can be adjusted. For our default test setup, we choose to *retain* this 500 ms wait period so as to reflect the performance one can expect with MPTCP as deployed today and hence our default results represent the pessimistic case for CellBricks. Later, we modify MPTCP to remove this default value and repeat key tests to show the effect of varying d.

Finally, to carry packets between our MPTCP-based UE and VM, we set up a software switch (OVS [73]) on each side of the UE and VM. The client-side OVS switch tunnels the packet to the OVS switch at the server, which strips off the packet's outer header such that the server sees packets with the UE's new IP address. For parity, we run the same OVS setup in our baseline TCP scenario but in this case, OVS simply pass packets through without any tunneling.

(iv) *Applications and their metrics* We run four classes of applications: standard network benchmarks (ping, iperf [56]), voice calls (pjsua [75]), video streaming (HLS [68, 97]), and web browsing (page downloading). Since VoIP does not use TCP (and hence MPTCP), we need a different approach to handle IP address changes in CellBricks. For this, we modify the pjsua client to use SIP's re-invite mechanism where a host sends a SIP re-Invite message to its peer upon IP changes allowing both endpoints to set up new RTP sessions [85],

Table 1 lists the performance metrics we track for each application. To measure the quality of voice calls, we used an industry standard quantitative call quality metric, the Mean Opinion Score (MOS), which can be numerically derived from the packet loss, latency, and jitter measured during the call [82]. The MOS varies from 1 to 5.0 where a score of 2 indicates poor quality and 4 indicates good quality. For video streaming, we measure the average quality level of the video playout, a key metric that is used to estimate the quality of experience for HLS/DASH-based video streaming [63]. Each quality level maps to a pre-defined video quality (*e.g.,* bitrate settings) and the higher the level the better the video quality but also the greater the network bandwidth consumption. Our HLS server streams 6 different quality levels (0-5) varying from 144p to 720p, transcoded using ffmpeg [39] from the same video file. We play the video stream with the hls.js [97] player at the UE and add instrumentation to collect metrics.

(v) *Mobility trajectory.* We pick three representative routes in the downtown, suburban, and highway areas of our geographic region. We repeatedly drive along each route with two UEs, each independently running the same application. We run tests during the day and the midnight-to-dawn period because, as we discovered during our drives, T-Mobile enforces different rate limiting policies at different times; see Appendix A for measurements of this behavior.

Main results: Table 1 summarizes our key results. We show the performance for each of our four applications, separated by whether the tests were run in the day (D) *vs.* night (N). We compare the performance of CellBricks to our baseline which is the current MNO architecture running TCP, for each route (suburb, highway, *etc.*) and in summary (the last two rows). The 2nd and 3rd column report some basic statistics for calibration: the mean-time-to-handover (MTTHO) measures the average time between handovers and the ping results measure the network latency from our UE to our server VM in EC2 (us-west). As expected, we see lower MTTHO when driving faster (*e.g.,* at night *vs.* day).

Overall, our main finding is that CellBricks with MPTCP achieves performance comparable to the TCP baseline for all four applications, with a slowdown of at most 3.06% (last row). Surprisingly, we even observe cases where CellBricks outperforms the MNO baseline, *e.g.,* web downloads at night are 1.61% faster with CellBricks (we explore why shortly).

In secondary observations, we see that: (i) video streaming is least sensitive to the choice of handover schemes due to its use of segment buffering that helps tolerate throughput fluctuations during handovers, (ii) most applications perform better at night when T-Mobile relaxes its rate limiting, allowing applications to achieve higher throughput (an average of 15.46 Mbps at night compared to 1.16 Mbps during the day). VoIP is less bandwidth intensive (requiring $\approx$ 30 kbps) and hence is less sensitive to this effect.

Understanding CellBricks's performance. We dig deeper to understand CellBricks's competitive performance.

One factor is simply the frequency of handovers – even in our worst-case (driving along the highway at night), we observe a handover only every 25.5 seconds and hence any overheads of re-attachment are averaged out. (Recall from earlier in this section, that MPTCP by default introduces a wait time of 500ms before initiating a new subflow.) However, as we'll show shortly, even when we zoom into the periods around handovers, CellBricks performs well.

We find that the reason for this has to do with the dynamics of slow-start. On a handover, CellBricks initiates a new MPTCP subflow, which enters slow-start and quickly catches up with its TCP counterpart in the MNO baseline (recall that the TCP connection undergoes no change during handovers beyond reacting to any loss that might occur). In fact, for short periods, the MPTCP subflow achieves *higher* throughput than the TCP flow. *E.g.,* Fig.8 zooms in

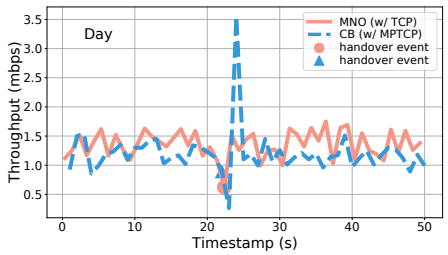

Figure 8: Comparison of the network throughput (iperf) achieved by MNO and emulated CellBricks over time.

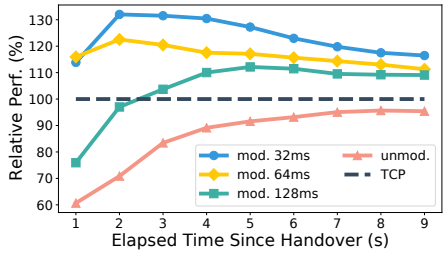

Figure 9: Impact of varying attachment latency on the iperf throughput. We report the relative performance using the TCP results from the same run as the baseline.

on the iPerf throughput at 1-second intervals for a 50s period in four of our traces. A handover event occurs at around second 23 and we see that MPTCP's performance drops close to zero (reflecting MPTCP's 500ms wait period) but then quickly ramps back up and temporarily even overshoots the TCP flow (we see this in the "spike" that appears in the few seconds right after the handover). We were also able to reproduce this phenomenon in controlled experiments.

Factor analysis: varying attachment latency. Next, we evaluate the impact of attachment latency on the performance of CellBricks. For this, we reconfigure MPTCP to remove the 500ms default wait period and rerun the iperf experiments for different attachment latencies: d=32, 64, 128 ms. (Recall from our prototype benchmarks that we expect attachment latencies in the 30-80ms range depending on the location of our broker.) We measure performance at night so that performance is less constrained by T-Mobile's rate limits. Fig.9 shows our results. The Y-axis shows the average throughput that CellBricks achieves normalized by that of our baseline's TCP throughput. To show the performance impact at different timescales, we measure normalized throughput in the n seconds after a handover and plot performance for different n on the X-axis. I.e., a data point corresponding to 2s on the X-axis shows the average throughput MPTCP achieved in the 2s window after handover, normalized by the average throughput that TCP did in the same period.

As expected, we see that performance degrades with higher attachment latencies. *e.g.*,, at second 2, CellBricks with an attachment latency of 32ms has 7.7% higher performance than with an attachment latency of 64 ms. We also see that removing the 500ms wait time in MPTCP improves our performance and, although MPTCP and TCP eventually converge to equal throughput, CellBricks now routinely outperforms our MNO baseline during handovers! In general we find that, without MPTCP's 500ms wait time, CellBricks achieves 10%-30% higher throughput than TCP in the first few seconds after handovers due to the impact of slow-start.

In summary, CellBricks's approach to mobility does not degrade application performance.

7 RELATED WORK AND CONCLUSION

Various efforts seek to improve implementations of the *existing* cellular architecture; *e.g.*, via disaggregation [44], improved modularity [17, 77, 90], or leveraging software [58, 106]. We instead propose a new architecture that redesigns and reorganizes functions across different players so as to improve competition amongst providers and reduce architectural complexity. We see these goals as complementary.

CellBricks shares similarities to network infrastructure sharing approaches like MORAN [51] and open-access networks [28] which enable MNOs to operate on the same RAN infrastructure. These require trust and tight coupling between the shared RAN and the MNOs that use it, incurring high transaction costs and limiting scale. CellBricks alieviates these constraints by supporting lightweight, many-to-many relationships between bTelcos and brokers. In the Wi-Fi domain, there are proposals like eduroam [100] and OpenRoaming [99] that allow users to receive access from Wi-Fi hotspots operated by a diverse range of organizations. Compared with these proposals, CellBricks allows untrusted access providers with its secure authentication and billing protocols, and supports seamless mobility across bTelcos with its host-driven design.

CellBricks shares many high level goals with the personal router project [27]: *e.g.*, enabling an open, competitive market in which small access providers can offer services. However, the personal router project relies on Mobile IP [72] for mobility and implicitly assumes that access providers are trusted, both of which are different from CellBricks, as mentioned above.

Finally, there is a long history [41] of researchers and industry proposing novel architectures aimed at simplifying the deployment of mobile networks, including truly micro-scale networks [46, 54, 81, 86], particularly in rural areas. *E.g.*, Magma [38] and CCM [53] do so via an orchestrator service that acts as an intermediary in establishing trust relationships with both bTelcos and larger providers. More generally, these efforts aim to enable small-scale networks where MNOs provide no service, while ours is to potentially enable the *replacement* of these MNOs by bTelcos.

We believe the time is right to explore new cellular designs such as CellBricks since we are seeing a proliferation of low-cost software defined radio base stations (which makes it easier to explore new designs) at the same time 5G has brought a pressing need for denser deployments (which makes finding new and efficient designs important). CellBricks can be incrementally deployed, initially complementing existing networks just as current cellular networks do generational upgrades. Moreover, all of the changes required by CellBricks are quite feasible: no change to the radio/RAN, straightforward changes to the software functions in the cellular core, minor changes to UE firmware, and only configuration changes (to enable MPTCP) to the network software stack on clients and servers. An incremental deployment of these relatively minor changes would enable a transformation in how cellular service is delivered.

ACKNOWLEDGEMENTS

We thank our shepherd Olivier Bonaventure, the anonymous reviewers, Christian Maciocco, and Gergely Pongracz for their useful feedback. We gratefully acknowledge the financial support from Intel, VMware, Facebook and Ericsson, as well as support for authors at ICSI and UC Berkeley from NSF grants 1817115, 1553747.

REFERENCES

[1] 3GPP. 2016. *LTE;Telecommunication management; Performance Management (PM); Performance measurements Evolved Universal Terrestrial Radio Access Network (E-UTRAN).* Technical Specification (TS) 32.425. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_ts/132400_132499/132425/13.05.00_60/ts_132425v130500p.pdf Version 13.5.0.

[2] 3GPP. 2017. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification.* Technical Specification (TS) 36.331. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_ts/136300_136399/136322/14.01.00_60/ts_136322v140100p.pdf Version 14.1.0.

[3] 3GPP. 2017. *Universal Mobile Telecommunications System (UMTS); LTE; Evolved Packet System (EPS); Mobility Management Entity (MME) and Serving GPRS Support Node (SGSN) related interfaces based on Diameter protocol.* Technical Specification (TS) 29.272. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_ts/129200_129299/129272/14.03.00_60/ts_129272v140300p.pdf Version 14.3.0.

[4] 3GPP. 2018. *Lawful Interception (LI); Handover interface for the lawful interception of telecommunications traffic.* Technical Specification (TS) 201.671. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_es/201600_201699/201671/03.02.01_50/es_201671v030201m.pdf Version 3.2.1.

[5] 3GPP. 2019. *Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Policy and charging control architecture.* Technical Specification (TS) 23.203. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_ts/123200_123299/123203/15.05.00_60/ts_123203v150500p.pdf Version 15.5.0.

[6] 3GPP. 2020. Non-Access Stratum. https://www.3gpp.org/technologies/keywords-acronyms/96-nas. (2020).

[7] 3GPP. 2020. *Non-Access Stratum (NAS) protocpol for 5G System (5GS);.* Technical Specification (TS) 24.501. 3rd Generation Partnership Project (3GPP). https://www.etsi.org/deliver/etsi_ts/136300_136399/136322/14.01.00_60/ts_136322v140100p.pdf Version 15.6.0.

[8] 3GPP. 2020. *Universal Mobile Telecommunications System (UMTS); LTE; Digital cellular telecommunications system (Phase 2+) (GSM); 3G security; Lawful interception architecture and functions.* Technical Specification (TS) 33.107. 3rd Generation Partnership Project (3GPP). https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2266 Version 16.0.0.

[9] 3GPP. 2021. 3GPP RAN Sharing. https://www.3gpp.org/news-events/1592-gush. (2021).

[10] 3GPP. 2021. 3GPP Specification series. https://www.3gpp.org/DynaReport/36-series.htm. (2021).

[11] Africa Mobile Networks. [n. d.]. http://www.africamobilenetworks.com. ([n. d.]). Retrieved 6/2020.

[12] Agentschap Telecom. 2013. Regeling gebruik van frequentieruimte zonder vergunning 2008. http://wetten.overheid.nl/BWBR0023553/volledig/geldigheidsdatum_23-04-2013. (April 2013).

[13] CBRS Alliance. [n. d.]. https://www.cbrsalliance.org. ([n. d.]).

[14] Apple. 2021. Wi-Fi network roaming with 802.11k, 802.11r, and 802.11v on iOS. https://support.apple.com/en-us/HT202628. (2021).

[15] AT&T. 2020. AT&T Private Cellular Networks. https://www.business.att.com/products/att-private-cellular-networks.html. (2020).

[16] AWS. 2021. AWS EC2 Regions. https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RegionsAndAvailabilityZones.html. (2021).

[17] Arijit Banerjee, Rajesh Mahindra, Karthik Sundaresan, Sneha Kasera, Kobus Van der Merwe, and Sampath Rangarajan. 2015. Scaling the LTE Control-Plane for Future Mobile Access. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 19.

[18] Oyster Bay. 2013. Top 10 Mobile Carriers Generate US 202B in Gross Profit. https://www.abiresearch.com/press/top-10-mobile-carriers-generate-us-202-billion-in-/. (2013). Accessed: 2020-12-26.

[19] Naga Bhushan, Junyi Li, Durga Malladi, Rob Gilmore, Dean Brenner, Aleksandar Damnjanovic, Ravi Teja Sukhavasi, Chirag Patel, and Stefan Geirhofer. 2014. Network densification: the dominant theme for wireless evolution into 5G. *IEEE Communications Magazine* 52, 2 (2014), 82–89.

[20] Olivier Bonaventure. 2019. MPTCP Deployment. http://blog.multipath-tcp.org/blog/html/index.html. (2019).

[21] BroadbandSearch. 2020. Mobile vs. Desktop Internet Usage. https://bit.ly/2MPNi5V. (2020).

[22] CableLabs. 2019. EPS AKA. https://www.cablelabs.com/insights/a-comparative-introduction-to-4g-and-5g-authentication. (2019). Accessed: 2020-04-29.

[23] CellBricks. 2021. CellBricks's Artifacts. https://cellbricks.github.io/artifact-sigcomm21/. (2021).

[24] CellBricks. 2021. CellBricks's Technical Report. https://cellbricks.github.io/cellular_sigcomm_extended.pdf. (2021).

[25] Yung-Chih Chen, Yeon-sup Lim, Richard J Gibbens, Erich M Nahum, Ramin Khalili, and Don Towsley. 2013. A measurement-based study of multipath tcp performance over wireless networks. In *Proceedings of the 2013 conference on Internet measurement conference*. 455–468.

[26] Cisco. 2020. Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. (2020).

[27] David D Clark and John T Wroclawski. 2000. *The personal router whitepaper.* Technical Report. MIT Technical Report.

[28] Red Compartida. [n. d.]. http://www.sct.gob.mx/red-compartida/index-eng.html. ([n. d.]).

[29] Congreso de la Republica del Peru. 2013. Ley No. 30083. http://www.leyes.congreso.gob.pe/Documentos/Leyes/30083.pdf. (2013).

[30] Vertical Consultants. 2020. Cell Tower Industry Facts & Figures 2020. https://www.celltowerleaseexperts.com/cell-tower-lease-news/cell-tower-industry-facts-figures-2016/. (2020).

[31] Wikipedia contributors. 2020. International mobile subscriber identity. https://en.wikipedia.org/wiki/International_mobile_subscriber_identity. (2020).

[32] Andrei Croitoru, Dragos Niculescu, and Costin Raiciu. 2015. Towards wifi mobility without fast handover. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 219–234.

[33] Quentin De Coninck and Olivier Bonaventure. 2019. MultipathTester: Comparing MPTCP and MPQUIC in mobile environments. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 221–226.

[34] Omkar Dharmadhikari. 2019. 5G Link Aggregation with Multipath TCP (MPTCP). https://www.cablelabs.com/5g-link-aggregation-mptcp. (2019).

[35] Ericsson. 2020. Evolving Cellular IoT for industry digitalization. https://www.ericsson.com/en/internet-of-things/iot-connectivity/cellular-iot. (2020).

[36] ETSI. 2020. Lawful Intercept ETSI. https://www.etsi.org/technologies/lawful-interception. (2020).

[37] EventHelix. 2019. Attachment Call Flow. https://www.eventhelix.com/lte/attach/lte-attach.pdf. (2019). Accessed: 2020-04-29.

[38] Facebook. [n. d.]. Magma. https://www.magmacore.org/. ([n. d.]).

[39] FFmpeg. 2020. A complete, cross-platform solution to record, convert and stream audio and video. https://ffmpeg.org/. (2020).

[40] R. Fielding, Y. Lafon, and J. Reschke. 2014. RFC 7233: Hypertext Transfer Protocol (HTTP/1.1): Range Requests. (2014).

[41] Claude Fischer. 1992. *America Calling: A Social History of the Telephone to 1940.* University of California Press, Berkeley, CA.

[42] Alan Ford, Costin Raiciu, Mark Handley, Olivier Bonaventure, and C Paasch. 2013. RFC 6824: TCP extensions for multipath operation with multiple addresses. *Internet Engineering Task Force* (2013).

[43] R.D. Foreman. 1999. Scale Economies In Cellular Telephony: Size Matters. Journal of Regulatory Economics 16, 297–306 (1999), https://doi.org/10.1023/A:1008131223498. (1999).

[44] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K Marina. 2017. Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine* 55, 5 (2017), 94–100.

[45] Lorenzo Franceschi-Bicchierai. 2012. Why Monopolistic Telecoms Threaten Internet Equality. https://mashable.com/2012/09/17/telecom-monopoly-internet-equality/. (2012).

[46] Hernan Galperin and François Bar. 2006. The Microtelco Opportunity: Evidence from Latin America. In *Information Technologies and International Development*, Vol. 3.

[47] Torsten J Gerpott, Wolfgang Rams, and Andreas Schindler. 2001. Customer retention, loyalty, and satisfaction in the German mobile cellular telecommunications market. *Telecommunications Policy* 25, 4 (2001), 249 – 269. https://doi.org/10.1016/S0308-5961(00)00097-5

[48] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. srsLTE: An open-source platform for LTE evolution and experimentation. In *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*. 25–32.

[49] Google-Fi. 2021. https://fi.google.com/about/. (2021).

[50] GSAcom. 2019. LTE in Unlicensed and Shared Spectrum: Trials, Deployments and Devices. https://gsacom.com/paper/lte-unlicensed-shared-spectrum-2/. (2019).

[51] GSMA. 2016. Unlocking Rural Coverage: Enablers for commercially sustainable mobile network expansion. (7 2016). https://www.gsma.com/mobilefordevelopment/resources/unlocking-rural-coverage-enablers-commercially-sustainable-mobile-network-expansion/

[52] GSMA. 2018. *Enabling Neutral Host.* Case Study. GSM Association (GSMA). https://www.gsma.com/futurenetworks/wp-content/uploads/2018/09/180920-CCS_GSMA_Case_Study-FINAL_NE-Modelling-removed.pdf

[53] Shaddi Hasan, Mary Claire Barela, Matthew Johnson, Eric Brewer, and Kurtis Heimerl. 2019. Scaling Community Cellular Networks with CommunityCellularManager. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI'19)*. 735–750.

[54] Kurtis Heimerl, Shaddi Hasan, Kashif Ali, Eric Brewer, and Tapan Parikh. 2013. Local, Sustainable, Small-Scale Cellular Networks. In *Proceedings of the Sixth*

International Conference on Information and Communication Technologies and Development (ICTD '13). ACM, Cape Town, South Africa, 2–12.
[55] Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. 2019. Privacy Attacks to the 4G and 5G Cellular Paging Protocols Using Side Channel Information.. In *NDSS*.
[56] iPerf. 2021. iPerf Network Benchmarks. https://iperf.fr/. (2021).
[57] Jana Iyengar and Martin Thomson. 2021. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. (May 2021). https://doi.org/10.17487/RFC9000
[58] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. 2013. SoftCell: Scalable and Flexible Cellular Core Network Architecture. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*. ACM, 163–174.
[59] Meghan Keneally. 2018. How the T-Mobile and Sprint merger could impact consumers. https://abcnews.go.com/Business/mobile-sprint-merger-impact-consumers/story?id=54826385. (2018).
[60] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. 2017. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 183–196.
[61] Yuanjie Li, Kyu-Han Kim, Christina Vlachou, and Junqing Xie. 2019. Bridging the data charging gap in the cellular edge. In *Proceedings of the ACM Special Interest Group on Data Communication*. 15–28.
[62] Yuanjie Li, Chunyi Peng, Zengwen Yuan, Jiayao Li, Haotian Deng, and Tao Wang. 2016. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 202–215.
[63] Yao Liu, Sujit Dey, Don Gillies, Faith Ulupinar, and Michael Luby. 2013. User experience modeling for DASH video. In *2013 20th International Packet Video Workshop*. IEEE.
[64] Christopher Mitchell. 2018. Profiles of Monopoly: Big Cable & Telecom. https://ilsr.org/monopoly-networks/. (2018).
[65] Christopher Mitchell. 2020. A Major Telecom Monopoly Fails America. (Apr 2020). https://ilsr.org/a-major-telecom-monopoly-fails-america/
[66] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas Henderson. 2008. *Host identity protocol*. Technical Report. RFC 5201, April.
[67] Netmanias. 2013. LTE Security II: NAS and AS Security. https://www.netmanias.com/en/?m=view&id=techdocs&no=5903. (2013).
[68] NGINX. 2021. Nginx Hls module. http://nginx.org/en/docs/http/ngx_http_hls_module.html. (2021).
[69] Binh Nguyen, Tian Zhang, Bozidar Radunovic, Ryan Stutsman, Thomas Karagiannis, Jakub Kocur, and Jacobus Van der Merwe. 2018. ECHO: A Reliable Distributed Cellular Core Network for Hyper-Scale Public Clouds. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. ACM, 163–178.
[70] Christoph Paasch, Gregory Detal, Fabien Duchene, Costin Raiciu, and Olivier Bonaventure. 2012. Exploring mobile/WiFi handover with multipath TCP. In *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. 31–36.
[71] Steven Pearlstein. 2019. Washington Post Article. https://www.washingtonpost.com/business/2019/06/19/looming-t-mobile-sprint-merger-is-wakeup-call-free-markets-failures/. (06 2019).
[72] Charles E Perkins. 1997. Mobile ip. *IEEE communications Magazine* 35, 5 (1997), 84–99.
[73] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. 2015. The design and implementation of open vswitch. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 117–130.
[74] Aaron Pressman. 2020. Fortune. T-Mobile claims largest 5G network. https://fortune.com/2020/11/05/t-mobile-5g-coverage-verizon-best/. (2020). Accessed: 2020-12-26.
[75] PJSIP project. 2021. https://github.com/pjsip/pjproject. (2021).
[76] Telecom Infra Project. 2021. https://telecominfraproject.com/. (2021).
[77] Zafar Ayyub Qazi, Melvin Walls, Aurojit Panda, Vyas Sekar, Sylvia Ratnasamy, and Scott Shenker. 2017. A High Performance Packet Core for Next Generation Cellular Networks. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 348–361.
[78] QCSuper. 2020. https://github.com/P1sec/QCSuper. (2020).
[79] QualComm. 2020. Qualcomm QXDM. https://bit.ly/2YrOVw1. (2020).
[80] Costin Raiciu, Dragos Niculescu, Marcelo Bagnulo, and Mark James Handley. 2011. Opportunistic mobility with multipath TCP. In *Proceedings of the sixth international workshop on MobiArch*. 7–12.
[81] Rhizomatica. 2013. http://rhizomatica.org/. (2013). Retrieved 4/2013.
[82] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. 2001. Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, Vol. 2. IEEE, 749–752.
[83] Jonathan Rosenberg, Henning Schulzrinne, Gonzalo Camarillo, Alan Johnston, Jon Peterson, Robert Sparks, Mark Handley, and Eve Schooler. 2002. RFC3261: SIP: session initiation protocol. (2002).
[84] M Isabel Sanchez and Azzedine Boukerche. 2016. On IEEE 802.11 K/R/V amendments: Do they have a real impact? *IEEE Wireless Communications* 23, 1 (2016), 48–55.
[85] Henning Schulzrinne, Steven Casner, R Frederick, and Van Jacobson. 2003. RFC3550: RTP: A transport protocol for real-time applications. (2003).
[86] Spencer Sevilla, Matthew Johnson, Pat Kosakanchit, Jenny Liang, and Kurtis Heimerl. 2019. Experiences: Design, Implementation, and Deployment of CoLTE, a Community LTE Solution. In *The 25th Annual International Conference on Mobile Computing and Networking*. 1–16.
[87] Statista. 2020. US Wireless Subscriptions Market Share. https://bit.ly/2Yxsbuz. (2020).
[88] Randall Stewart and Christopher Metz. 2001. SCTP: new transport protocol for TCP/IP. *IEEE Internet Computing* 5, 6 (2001), 64–69.
[89] Daehyun Strobel. 2007. IMSI catcher. *Chair for Communication Security, Ruhr-Universität Bochum* 14 (2007).
[90] Oğuz Sunay, Shad Ansari, Woojoong Kim, Pingping Lin, Hyunsun Moon, Badhrinath Padmanabhan, Guru Parulkar, Larry Peterson, and Ajay Thakur. 2020. *Aether: Enterprise-5G/LTE-Edge-Cloud-as-a-Service*. Technical Report. Open Networking Foundation.
[91] Mist Systems. 2021. 802.11k, 802.11r, and 802.11v. https://www.mist.com/documentation/802-11k-802-11r-802-11v/. (2021).
[92] Hughes Systique. 2014. LTE Wifi Data Offload - A Brief Survey. https://hsc.com/DesktopModules/DigArticle/Print.aspx?PortalId=0&ModuleId=1215&Article=224. (2014).
[93] Multipath TCP. 2020. MPTCP/mptcp_fullmesh.c. https://github.com/multipath-tcp/mptcp/blob/5b127fba5f34e8acbb5067d5940bd13678c7b7dc/net/mptcp/mptcp_fullmesh.c#L1070. (2020).
[94] Telecoms.com. 2020. Neutral host networks and how to support them. https://telecoms.com/opinion/neutral-host-networks-and-how-to-support-them/. (2020).
[95] USRP. 2020. USRP B205mini. https://www.ettus.com/all-products/usrp-b205mini-i/. (2020).
[96] Tommaso M. Valletti. [n. d.]. Is Mobile Telephony a Natural Oligopoly? Review of Industrial Organization 22, 47–65 (2003), https://doi.org/10.1023/A:1022191701357. ([n. d.]).
[97] video dev. 2021. JavaScript HLS client using Media Source Extension. https://github.com/video-dev/hls.js/. (2021).
[98] W3Techs. 2020. Usage statistics of QUIC for websites. https://w3techs.com/technologies/details/ce-quic. (2020).
[99] WBA. 2021. OpenRoaming. https://wballiance.com/openroaming/. (2021).
[100] Klaas Wierenga and Licia Florio. 2005. Eduroam: past, present and future. *Computational methods in science and technology* 11, 2 (2005), 169–173.
[101] Cricket Wireless. 2021. https://www.cricketwireless.com/,. (2021).
[102] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. 2011. Design, Implementation and Evaluation of Congestion Control for Multipath TCP.. In *NSDI*, Vol. 11. 8–8.
[103] RF Wireless World. 2020. LTE security | Security in LTE networks. https://www.rfwireless-world.com/Tutorials/LTE-security.html. (2020).
[104] RF Wireless World. 2021. LTE to WLAN(wifi) Handover. https://www.rfwireless-world.com/Terminology/LTE-WLAN-handover.html. (2021).
[105] Christos Xenakis and Christoforos Ntantogian. 2015. Attacking the baseband modem of mobile phones to breach the users' privacy and network security. In *2015 7th International Conference on Cyber Conflict: Architectures in Cyberspace*. IEEE, 231–244.
[106] Mao Yang, Yong Li, Depeng Jin, Li Su, Shaowu Ma, and Lieguang Zeng. 2013. OpenRAN: a software-defined ran architecture via virtualization. *ACM SIGCOMM computer communication review* 43, 4 (2013), 549–550.
[107] ZTE. 2012. MF820B LTE USB Modem Quick Guide. https://usermanual.wiki/ZTE/MF820B/pdf. (2012).

APPENDIX

Appendices are supporting material that have not been peer-reviewed.

A T-MOBILE: NIGHT *VS.* DAY

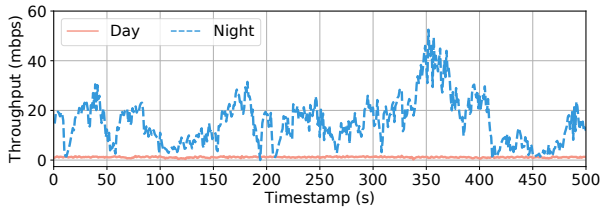

Figure 10: Comparison of iperf throughput over time in the day and midnight (the downtown route) mobility experiment.

Fig.10 depicts the iperf throughput measured during the day and the night traversing the same downtown route. The performance exhibit clear bi-modal pattern – the average throughput in the night (14.95 mbps) is 14.5x higher than what measured in the day (1.03 mbps) with the peaks differ even more dramatically (52.5 mbps vs. 1.75 mbps). We also notice that the throughput has higher variance in the night, with a standard deviation of 8.94, compared to 0.32 in the day. As we repeat the same experiments, we found that the throughput enters the high-mode consistently at around 12:30am. We conjecture this is due to the different rate limiting policies the MNO (T-Mobile) enforces during these two time windows, where the MNO "switches off" the aggressive rate limiting used in the day around midnight.