SoftCell: Scalable and Flexible Cellular Core Network Architecture

Xin Jin[†], Li Erran Li[⋆], Laurent Vanbever[†], and Jennifer Rexford[†]
Princeton University[†], Bell Labs[⋆]

ABSTRACT

Cellular core networks suffer from inflexible and expensive equipment, as well as from complex control-plane protocols. To address these challenges, we present SoftCell, a scalable architecture that supports fine-grained policies for mobile devices in cellular core networks, using commodity switches and servers. SoftCell enables operators to realize high-level service policies that direct traffic through sequences of middleboxes based on subscriber attributes and applications. To minimize the size of the forwarding tables, SoftCell aggregates traffic along *multiple* dimensions—the service policy, the base station, and the mobile device—at different switches in the network. Since most traffic originates from mobile devices, SoftCell performs fine-grained packet classification at the *access* switches, next to the base stations, where software switches can easily handle the state and bandwidth requirements. SoftCell guarantees that packets belonging to the same connection traverse the same sequence of middleboxes in both directions, even in the presence of mobility. We demonstrate that SoftCell improves the scalability and flexibility of cellular core networks by analyzing real LTE workloads, performing micro-benchmarks on our prototype controller as well as large-scale simulations.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Packet-switching networks*; C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network management*

General Terms

Algorithms, Design, Management

Keywords

Cellular core networks, software-defined networking, architecture design

1. INTRODUCTION

Cellular data traffic has exploded in recent years, in large part due to the rapid proliferation of cellular devices such as smart phones, tablets, smart meters and other Machine-to-Machine (M2M) devices [1]. This trend is likely to continue. As an illustration, Cisco predicts a 13 times increase of cellular data traffic between 2012 and 2017 [2]. New cellular technologies, like Long Term Evolution (LTE), have helped cellular providers to keep up with the traffic growth by increasing their radio access capacity. However, they now face the challenge of keeping up with the increasing demand in their core networks, which carry the User Equipment (UE) traffic between the Base Station (BS) and the Internet, as shown in Figure 1.

Unlike traditional IP networks, cellular providers rely extensively on customized policies based on a wide variety of *subscriber attributes* and *application types.* Typical subscriber attributes include the cell-phone model or the M2M device type, the operating-system version, the billing plan, the options for parental controls, whether the total traffic exceeds a usage cap, or whether a user is roaming. Typical application types include video traffic (for transcoding), web traffic (for caching), or specific applications for which the developers pay the carrier on the user's behalf [3] (for exempting that traffic from the user's cap). For example, the carrier may direct traffic for older phones through an echo-cancellation gateway, video traffic through a transcoder during times of congestion, M2M fleet-tracking traffic through a low latency path, and all traffic through a firewall.

To route traffic and perform fine-grained packet processing, cellular providers rely on specialized and proprietary devices, namely: Serving Gateways (S-GWs) and Packet data network Gateways (P-GWs). S-GWs are mainly used as mobility anchors to provide seamless mobility. P-GWs centralize most network functions like content filtering, traffic optimization, firewalls, and lawful intercept [4]. P-GWs sit at the boundary of the cellular network and the Internet. The base stations, S-GWs, and P-GWs communicate using GPRS Tunneling Protocol (GTP).

Centralizing nearly all data-plane functionalities in the P-GWs makes cellular core networks remarkably inefficient, complex, and inflexible [5, 3] for at least three reasons. First, cellular core networks must forward all traffic through the P-GWs—including device-to-device traffic and local content distribution network services. This increases the network delay and congestion. Second, since P-GWs are not modular, carriers often end up paying for functionalities they do not need; when a function is not available, carriers have no

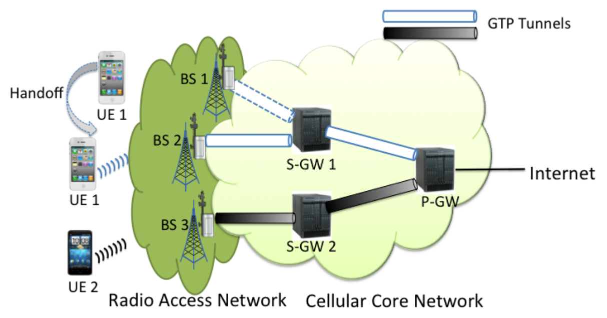


Figure 1: LTE network architecture

choice but to replace the P-GWs—even if they are sufficient for most purposes. Finally, carriers cannot "mix and match" capabilities from different vendors (e.g., use a firewall from one vendor, and a transcoder from another), or "scale up" the resources devoted to a specific function [3, 6].

Instead of performing all these functions at the Internet boundary, we argue that cellular providers should adopt a network design more akin to modern data centers. The network should consist of a fabric of simple core switches, with most functionality moved to low-bandwidth access switches (close to the base stations) and a distributed set of middleboxes that the carrier can expand as needed to meet the demands. These middleboxes could be dedicated appliances, virtual machines running on commodity devices [6], or packet-processing rules installed directly in the switches [7, 8]. A logically-centralized controller can then route traffic through the appropriate middleboxes, via efficient network paths, to realize a high-level service policy (e.g., directing a UE's video traffic through a transcoder and a firewall).

However, implementing such a design in cellular networks introduce unique scalability challenges. cellular networks are defined along many dimensions, leading to large number of packet classifiers. Yet, commodity switches can only store a few thousand to tens of thousands of rules [9]. Second, nearly all traffic in cellular networks goes to and from the Internet, whereas data centers [10, 11, 12] are dominated by traffic between servers. This places heavier bandwidth and state requirements on the cellular gateway switches. Third, device mobility is frequent and unplanned, requiring additional state to ensure seamless connectivity and direct all packets belonging to a connection through the same sequence of middleboxes.

To address these challenges, we present SoftCell, a scalable architecture for supporting fine-grained policies for mobile devices in cellular core networks. SoftCell employs the following two novel techniques:

Muti-dimensional aggregation: SoftCell significantly reduces the size of switch tables by aggregating entries along *multiple dimensions*, combining the benefits of traditional location-based routing and tag-based routing. identifier (identifying middlebox paths), (ii) the base station identifier, and (iii) a local UE identifier. SoftCell also exploits the wildcard matching capabilities of the TCAMs in modern switches to *selectively* match on these dimensions. The SoftCell controller aggregates the forwarding entries dynamically, using an online algorithm.

Smart access edge, dumb gateway edge: SoftCell obviates the need to perform packet classification at the Internet gateway edge. Instead, SoftCell performs all packet classification at the access edge, using software switches along with local software controllers. leverages the fact that traffic begins at the access edge, where the number of To avoid reclassifying the return traffic at the Internet edge, SoftCell has the ability to *embed* policy identifiers directly in the IP packet headers, enabling the Internet edge to only perform basic packet forwarding.

We built a SoftCell controller on top of Floodlight [13]. We then evaluated it using: i) real traces from a large LTE deployment; ii) micro-benchmarks on our prototype; and iii) large-scale simulation experiments. Our experiments and analysis show that SoftCell can easily handle the workload of a large LTE network and support thousands of service-policy clauses with just a few thousand TCAM entries in the core switches.

The rest of the paper is organized as follows. Section 2 presents the architecture of SoftCell. It describes the components of a SoftCell network and the flexible, high-level policies SoftCell supports. It then gives an overview of the technical challenges and solutions. Section 3 describes the multi-dimensional aggregation technique to scale service routing. Section 4 explains the asymmetric edge design to scale packet classification. Section 5 discusses how SoftCell handles various network dynamics. We give the performance evaluation in Section 6. We provide some discussions on SoftCell in Section 7, followed by an examination of related work in Section 8. The paper concludes in Section 9.

2. SOFTCELL ARCHITECTURE

SoftCell's goal is to support numerous fine-grained policies in a scalable manner for cellular core networks. A cellular core network is the network that connects base stations and the Internet as shown in Figure 2. In this section, we introduce the components in a SoftCell network, the specification of service policies, and the key design decisions.

2.1 SoftCell Core Network Components

SoftCell interconnects unmodified UEs (via base stations) and the Internet (via gateway switches), as shown in Figure 2. SoftCell does *not* require specialized network elements (e.g., S-GWs and P-GWs) or point-to-point tunneling (e.g., user-level GTP tunnels) used in today's LTE networks.

Controller: The controller implements high-level service policies by installing switch-level rules that direct traffic through middleboxes. Service policies are specified on subscriber attributes and application types. To compute these paths, the controller has access to the (mostly static) attributes of each UE (e.g., billing plan and phone model).

To support unmodified UEs, SoftCell controller implements the LTE signalling protocols used between UEs and their Mobility Management Entities (the main control plane entity in LTE). These protocols handle connection establishment and disconnection, tracking area update, paging, etc.

Access switches: Each base station has an *access* switch that performs fine-grained packet classification on traffic from UEs. Access switches can be software switches that

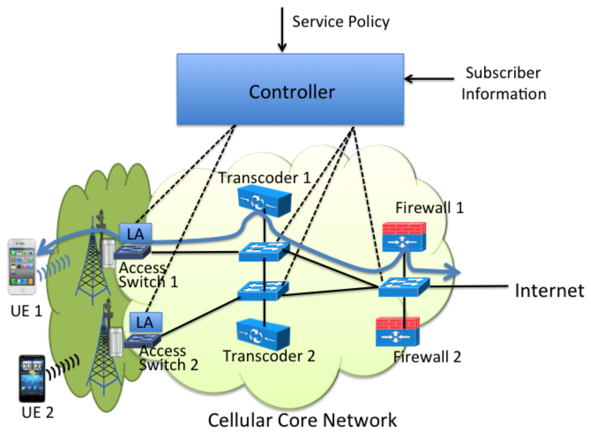


Figure 2: SoftCell network architecture

run on commodity servers. Today's software switches like Open vSwitch [14] can easily store 100K microflows in a hash table, and perform packet forwarding at several gigabits per second [15]. The server also runs a local agent (LA in Figure 2) that caches packet classifiers for attached UEs, to minimize interaction with the central controller.

Core switches: The rest of the network consists of *core* switches, including a few *gateway* switches connected to the Internet. These core switches are commodity hardware switches. They forward traffic through appropriate middleboxes. We assume that they can perform arbitrary wildcard matching on IP addresses and TCP/UDP port numbers (as in today's merchant silicon [16]), or can cache flat rules after processing wildcard rules locally in software (as in DevoFlow [17]). SoftCell gateway switches are *much* cheaper than P-GWs; they just perform packet forwarding, and relegate sophisticated packet processing to middleboxes.

Middleboxes: SoftCell supports commodity middleboxes such as dedicated appliances, virtual machines, or packet-processing rules on switches. Each middlebox function (e.g., firewall) may be available at multiple locations. SoftCell supports stateful middleboxes that require all packets in both directions of a connection to traverse the same instance.

The radio access networks consist of base stations that connect to unmodified UEs using existing protocols for managing mobility, sessions, and authentication. Just as today, a UE retains a single IP address as it moves between base stations in the same cellular core network; any changes the cellular core network makes to the IP addresses of packets are not visible to the UEs. SoftCell uses a different, location-dependent IP address for routing within the core network and the Internet (§ 3). Access switches perform address translation, transparent to UEs (§ 4). SoftCell does not require any change to the radio hardware at the base station, or common functions such as scheduling, radio resource management, and paging. SoftCell only changes how the base stations communicate with the core network, by having the base stations coordinate the controller to enforce service policies. Similarly, SoftCell does not require changes to commodity middleboxes, or any support from the rest of the Internet.

2.2 Flexible, High-Level Service Policies

Prio	Predicates	Service Actions
1	provider = B	Firewall
2	provider != A	Drop
3	app = video $\wedge$ plan = Silver	[Firewall, Transcoder]
4	app = VoIP	[Firewall, Echo-Cancel]
5	device type=M2M fleet	[HighPriority, Firewall]

Table 1: Example service policy for carrier A

The SoftCell controller directs traffic over network and middlebox paths, based on the service policy. Service policies are defined at a high level of abstraction, based on subscriber attributes and applications. The controller handles low-level details like ephemeral network identifiers, the locations of middleboxes and switches, and application identification. A service policy is composed of multiple clauses that specify *which* traffic (specified by a predicate) should be handled in *what* way (specified by an action):

Predicates: A predicate is a boolean expression on subscriber attributes and application types. Subscriber attributes consist of device type, billing plan, device capabilities, provider, etc. Application types include web browsing, real-time streaming video, VoIP, etc.

Service action: An action consists of a sequence of middleboxes, along with quality-of-service (QoS) and access-control specifications. Specifying a sequence of middleboxes allows the carrier to impose ordering constraints (e.g., firewall before transcoder). The action does not indicate a specific instance of each middlebox, allowing the controller to automatically select middlebox instances and network paths that minimize latency and load.

Priority: The priority is used to disambiguate overlapping predicates. The network forwards traffic using the highest-priority clause with a matching predicate.

In this paper, we focus on middlebox service policies, since they require more sophisticated *traffic steering* rather than simple local processing to drop packets or mark the type-of-service bits.

Table 1 shows an example of service policy for carrier A. In this example, carrier A has a roaming agreement with carrier B which enables B's subscribers to use A's network as fallback. To avoid abuse, the first clause directs all traffic of B's subscribers through a firewall. The second clause disallows traffic of subscribers from all other carriers. The remaining clauses specify the handling of A's own subscribers, with all traffic going through a firewall. The third clause indicates that the video traffic to subscribers on the "silver" billing plan must go through a transcoder after the firewall. The fourth clause specifies that VoIP traffic must go through an echo cancellation box (improving voice quality) after a firewall. The fifth clause requires M2M fleet-tracking traffic to be forwarded with high priority to ensure low latency.

2.3 SoftCell Design Challenges and Solutions

Before going into the details of SoftCell, we first give an overview of the design challenges and solutions.

Challenge 1: support fine-grained service policies with small switch tables. Fine-grained service policies in large networks can easily lead to an explosion in the data-plane state needed to direct traffic through the right middleboxes. For instance, suppose a service policy has 1000 clauses (very reasonable for a future network that consists of billions of M2M devices used in all kinds of settings such as tele-health, asset tracking, and building security) in a network with 1000 base stations. Suppose for each policy clause, we must instantiate a policy path (that traverses the right middlebox instances) between each base station and the gateway switch. This results in 1 million paths. Existing service routing techniques (e.g., [18]) are not able to install such a huge amount of paths with limited switch flow tables.

Solution: multi-dimensional aggregation. SoftCell tackles this challenge by leveraging aggregation. As simply aggregating on destination IP addresses does not provide enough flexibility and scalability to implement fine-grained service policies, SoftCell introduces *multi-dimensional* aggregation, which combines the benefits of traditional location-based routing and tag-based routing to scale to large networks with large service policies.

Challenge 2: support fine-grained packet classification in asymmetric topology. To enforce service policies, we must first classify packets at the network edge to decide which policy clause to apply. This is relatively easy in data centers as most traffic is "east west" (between servers *in* the data center). Packets are typically classified by software switches in hypervisors or first-hop hardware switches [10, 11, 12], and the load is *equally* distributed over all classification switches at the edge. However, in cellular core networks most traffic is "north south" (between the Internet and UEs). The *access* edge connecting to UEs, consists of thousands of base stations. An access switch at a base station handles traffic from up to 1000 UEs attached to the base station. The total traffic volume is around 20 Mbps to 1 Gbps [19, 20]. The *gateway* edge facing the Internet consists of just a few gateway switches that direct traffic to those thousands of base stations, and the traffic volume can be several Tbps. Thus, classifying packets at the gateway edge at line rate is very difficult.

Solution: smart access edge, dumb gateway edge. SoftCell exploits the unique property of cellular networks that traffic is initiated by UEs, and classifies packets *only* at the access edge. Using a novel state-embedding technique, SoftCell "piggybacks" the classification results in the source IP address and port number, so the gateway switches can easily forward return traffic by examining the destination address and port.

Challenge 3: scalable handling of network dynamics. Cellular networks operate under considerable churn due to UE mobility. Whereas data-center operators can plan VM migration in advance, cellular carriers have little control over UE mobility (beyond limited load balancing between nearby base stations). Existing research on mobility management shows how to minimize packet loss during handoff, but does not address service policies. Stateful middleboxes rely on *policy consistency*, where all packets of a connection must traverse the same middlebox instance. Besides policy consistency, centralized control introduces a new failure mode—controller failure—that must be addressed.

Solution: smart local agent at base stations. Each access switch acts as a mobility anchor for attached UEs. Upon a UE handoff, ongoing flows continue to reach the old access switch via the old path. This ensures policy consistency, while leveraging the vast number of access switches to achieve scalability. New flows traverse the new access switch, and new policy paths, to minimize path stretch. To ensure fast recovery from controller failures, SoftCell runs multiple controller replicas and maintains a consistent, distributed store of the control state. Most state changes slowly, lowering the overhead of maintaining strong consistency. The most dynamic state is UE location. A backup controller can fetch up-to-date UE location data directly from the local agent at each base station.

We describe the three challenges and solutions in § 3, § 4, and § 5 respectively. SoftCell does not restrict where carriers place the middleboxes. Carriers can either distribute middleboxes in the whole cellular core or centralize middleboxes in a computing cluster near the gateway. No matter which approach carriers adopt, they would always face the three challenges mentioned above. Furthermore, some middleboxes may be better to put inside the network like caching proxies and CDN nodes so that the routing to UEs are more efficient, while others may be better to put near the gateway like firewalls that block malicious Internet traffic. Actually, vendors like Ericsson are very interested in SoftCell-like solutions [21]. Here we identify the essential technical challenges to employ SDN in cellular core networks and provide SoftCell to solve them.

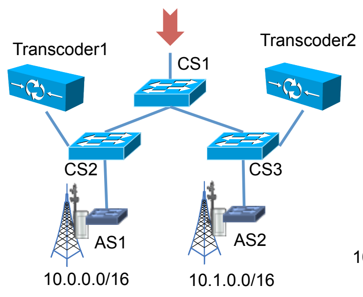
3. SCALABLE SERVICE ROUTING WITH MULTI-DIMENSIONAL AGGREGATION

To implement service policies, SoftCell routes traffic over sequences of middleboxes that are distributed throughout the network. While more flexible than a centralized P-GW, SoftCell service routing requires fine-grained forwarding rules. SoftCell overcomes this scalability challenge by aggregating forwarding rules along *multiple dimensions*. Moreover, SoftCell directly exploits the wildcard matching capability enabled by TCAM in modern switches to *selectively* match on multiple dimensions. In this section, we present SoftCell's aggregation strategy, and an online algorithm for computing the forwarding entries.

3.1 Multi-Dimensional Aggregation

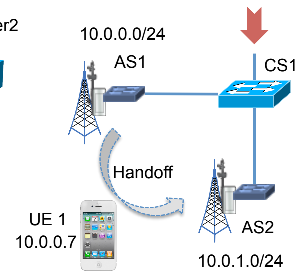
In traditional IP networks, routers forward traffic based solely on the destination prefix, and operators align IP prefixes with the topology to enable aggregation of contiguous prefixes. However, destination-based forwarding is not flexible enough for cellular networks, where forwarding can depend on subscriber attributes or application types. To enable more flexible forwarding, a classical solution is to forward traffic using VLAN tags or MPLS labels. However tag-based forwarding scales poorly as it enforces "flat-routing" and removes the ability to aggregate contiguous entries, even if the destination is the same. While label stacking and label swapping can reduce the number of tags, carrying multiple MPLS labels in the packet header incurs large overhead, and existing middleboxes do not necessarily understand, or even preserve, MPLS labels.

SoftCell combines the benefits of location-based routing and tag-based routing, by leveraging the ability of commodity switches to selectively match on different packet fields.



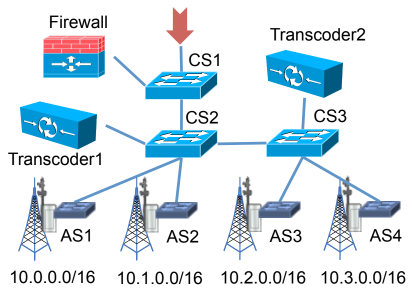
Switch	Match	Action
CS1	10.0.0.0/16	Forward to CS2
	10.1.0.0/16	Forward to CS3

(a) Location-based routing



Switch	Match	Action
CS1	10.0.0.7	Forward to AS2
	10.0.0.0/16	Forward to AS1
	10.1.0.0/16	Forward to AS2

(b) Mobility



Switch	Match	Action
CS1	tag1	Forward to Firewall
CS2	tag1, 10.0.0.0/15	Forward to Transcoder1
	tag1, 10.2.0.0/15	Forward to CS3
CS3	tag1, 10.2.0.0/15	Forward to Transcoder2

(c) Flexible policy

Figure 3: Examples of multi-dimensional aggregation rules for traffic arriving from the Internet

In particular, SoftCell forwarding rules aggregate on three dimensions: (i) *policy*, (ii) *location*, and (iii) *UE*.

Aggregation by policy (policy tag): Service policies defined on high-level attributes seem very compact. However, Subscriber attributes are not easily translated or aggregated with network addresses. Consider for instance the third clause in Table 1. As UEs with a "silver plan" can have a variety of IP addresses, this policy requires a rule for each flow in the worst case. We could conceivably assign "silver plan" UEs IP addresses under the same subnet, allowing us to assign one rule that matches on the IP prefix. However, we cannot do this for *every* attribute, not to mention that many service policies are defined on combinations of attributes. To minimize the rules in core switches, we use a *policy tag* to aggregate flows on the same policy path. This tag is associated at the access switch, allowing core switches to forward packets based on policy tags.

Aggregation by location (hierarchical IP address): In many core switches, traffic destined to the same base station traverses the same outgoing link, even if the packets go through different middleboxes. By including location information in the UE addresses, we can aggregate traffic by IP prefix. Furthermore, cellular core networks have a natural hierarchical structure. Therefore, we assign each base station an IP prefix, called *base station ID*, and IDs of nearby base stations can be further aggregated into larger blocks. We can aggregate even more by combining policy tags and IP addresses. Suppose two policy paths going to two base stations share a long path segment before branching. If assigned the same policy tag, a single rule matching on the tag can forward packets along the shared segment until the branching point, where traffic is divided based on the base station prefix.

Aggregation by UE (UE ID): Some middleboxes (like intrusion detection systems) need a way to identify groups of flows associated with the same UE. Clearly, this is impossible if all flows for the same base station share the same address. Packets therefore need an UE identifier (*UE ID*) that differs from other UEs at the same base station. Having an UE ID in each packet also enables optimizations for handling mobility, by installing switch rules that forward in-progress flows to the UE at its new location. Together, the base station prefix and the local UE ID form a hierarchical location-dependent address (LocIP) for the UE. This LocIP is transparent to the UE and is used for routing in the core network and the Internet, but *not* the radio access network. The UE is allocated a permanent IP address via DHCP when it first attaches to the network. This permanent IP address does not change, while LocIP changes when the UE moves between base stations. Access switches perform the translation between the permanent IP address and LocIP.

We can furthermore maximize the aggregation of the data-plane state by *selectively* matching on multiple fields. We describe three examples.

Location-based routing: In Figure 3(a), core switch CS1 matches on the base-station prefix to forward traffic to CS2 and CS3. CS2 and CS3 then decide whether to direct traffic to a transcoder based on the policy tag. Notice that CS1 does not need to base its forwarding decision on the tag.

UE mobility: In Figure 3(b), CS1 forwards traffic to base stations according to the destination IP prefix. When UE1 moves from access switch AS1 to AS2, we install a high-priority rule at CS1 to match on both the base station prefix and the UE ID. This ensures that new flows reach UE1 at AS2 over a direct path. More advanced mobility handling that ensures policy consistency is discussed later in § 5.1.

Flexible policy: Figure 3(c) illustrates how to implement the third clause in Table 1 with "tag1." CS1 forward "tag1" packets to the Firewall[1]. Suppose we assign AS1 and AS2 traffic to Transcoder1, and AS3 and AS4 traffic to Transcoder2. CS2 matches on both the tag and the prefix (more precisely, the aggregated prefix of two base stations) to forward AS1 and AS2 traffic to Transcoder1, and AS3 and AS4 traffic to CS3. CS3 finally forwards AS3 and AS4 traffic to Transcoder2.

3.2 Rule Minimization in Core Switches

We now present a greedy online algorithm that performs multi-dimensional aggregation given a stream of policy paths

[1]Traffic *from* middleboxes is identified based on the inport.

Algorithm 1 Install a new policy path

```
Input:
  – path: the policy path to install
  – prefix: the IP prefix of the origin base station
  – candTag: the set of candidate tags for the base station
Output: switch rules and a tag for this policy path
    Step 1: Choose a tag to minimize new rules
 1: for $t$ in $candTag$ do
 2:     $newRule[t] = 0$    ▷ new rules needed if tag $t$ is used
 3:     for $(sw_i, sw_{i+1})$ in $path$ do
 4:         if $sw_i.getNextHop(t, prefix)! = sw_{i+1}$ then
 5:             if $!sw_i.canAggregate(t, prefix, sw_{i+1})$ then
 6:                 $newRule[t] += extraRules(path, sw_i)$
 7: if $candTag \,! = \emptyset$ then
 8:     $tag^* = \arg\min_t \{newRule[t]\}$
 9: else
10:     $tag^* = new\ tag$
    Step 2: Install the path with the prefix and tag
11: for $(sw_i, sw_{i+1})$ in $path$ do
12:     if $sw_i.getNextHop(tag^*, prefix)! = sw_{i+1}$ then
13:         if $sw_i.canAggregate(tag^*, prefix, sw_{i+1})$ then
14:             $sw_i.aggregateRule(tag^*, prefix, sw_{i+1})$
15:         else
16:             $sw_i.installRule(tag^*, prefix, sw_{i+1})$
```

as input (see Algorithm 1). The algorithm operates in an online fashion because policy paths can be dynamically installed or removed due to policy changes or middlebox load balancing. We only briefly explain the algorithm here and defer the reader to [22] for a more detailed explanation.

Basic multi-dimensional aggregation algorithm: Intuitively, the algorithm performs multi-dimensional aggregation of a policy path p in two stages: aggregating first by policy, then by location. A policy path p is a sequence $(sw_0, \ldots, sw_k)$ of adjacent switches such that sw_0 represents an access switch connected to a base station and sw_k a gateway switch connected to the Internet. Given p, the algorithm first computes *candTag*, the set of tags used on any switch belonging to p, except sw_0[2]. For each tag $t \in$ *candTag*, the algorithm computes the total number of forwarding rules that must be created if that tag is reused to set-up p (line 1–6).

To do so, the algorithm iterates over each consecutive pairs of switches (sw_i, sw_{i+1}) in p and checks whether the forwarding rule associated with t on sw_i can be reused as such to forward traffic using p, i.e. whether its next-hop is sw_{i+1} (line 4). If the next-hop differs, the algorithm (function $extraRules(path, sw_i)$) accounts for extra forwarding rules matching both t and the location address *unless* that extra rule can be aggregated with another existing rule (line 5). If the next hop is a middlebox, we need rules for traffic *to* and *from* the middlebox. To ensure correctness, the algorithm aggregates two rules if and only if their location prefixes are contiguous. At the end of the loop, the algorithm returns the tag that minimizes the number of new rules or returns a new tag if needed (lines 7–10). Finally, the algorithm installs the forwarding rules to switches using consistent updates techniques [23] and aggregating the entries where it can (line 11–16).

[2]Otherwise, it would be impossible to distinguish among different policy paths originated from the same sw_0.

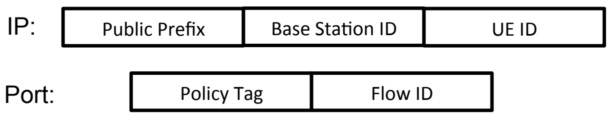


Figure 4: Embedding location and policy information in source IP address and source port number. Thus the information can be implicitly piggybacked in return traffic.

Dealing with loops: Ideally, we should only compute and install loop-free paths. However, due to the flexibility of service policies and placements of middleboxes, in some cases policy paths can contain one or more loops. As an illustration, consider a policy path enforcing outbound video traffic to go through a firewall before a video transcoder in Figure 2. A loop that enters the same switch twice but through different links can easily be differentiated based on the input ports. However, a loop that enters the same switch twice from the *same* link is more difficult to handle. In such a case, the algorithm uses additional tags to disambiguate the forwarding decisions (omitted in the algorithm for space constraints). More specifically, the algorithm breaks the loop into two segments and uses different tags for each segment. At the switch that connects these two segments, the algorithm installs a rule to "swap" these two tags. This approach can be generalized to support nested loops.

Discussion of offline algorithm: Our online algorithm is optimal if each policy path is processed one at a time. For extremely constrained environments, we can couple the online algorithm with an offline algorithm that would regularly recompute the optimal forwarding entries. We leave a description of the offline algorithm to separate work as Algorithm 1 already supports orders of magnitude more policy paths on commodity switches than what is required today by operators.

4. SCALABLE PACKET CLASSIFICATION WITH ASYMMETRIC EDGE DESIGN

Each packet entering the network must be associated with the appropriate policy tag and location-dependent IP address. This imposes overhead in both the data plane (to apply packet-classification rules) and the control plane (to fetch the rules). SoftCell places key functionality at the low-bandwidth *access* edge, to limit the data-plane overhead on the gateway switches and the control-plane overhead on the controller.

4.1 Packet Classification at the Access Edge

Each time a packet arrives at a base station, the access switch needs to translate the permanent IP address to the location-dependent IP address assigned to the UE, and attach the appropriate policy tag. To realize this, SoftCell installs one rule for each microflow at the access switch. A base station has at most 1000 UEs with (say) ten flows each, resulting in 10,000 microflows—easily supported in a software switch [15]. To avoid classifying packets again at the gateway switches (which aggregates traffic for thousands of base stations), SoftCell embeds the packet-classification result in the packet.

Embedding state in packet headers: Rather than *encapsulating* packets, as is commonly done in data-center networks, we *embed* the policy tag, base station ID, and UE ID in the packet header. This ensures that the return traffic from the Internet carries these fields. For example, we could encode the state as part of the UE's IP address (e.g., in IPv6), or a combination of the UE's IP address and TCP/UDP port number (e.g., in IPv4) as shown in Figure 4. The access switch rewrites the source IP address to the location-dependent IP address (i.e., the carrier's public prefix, as well as the base station and UE IDs), and embeds the policy tag as part of the source port. UEs do not have many active flows, leaving plenty of room for carrying the policy tag in the port-number field. With this embedding mechanism, our three identifiers are implicitly "piggybacked" in return traffic arriving from the Internet. The gateway switch can simply forward incoming packets based on the destination IP address and port number.

Dealing with security and privacy issues: Directly applying this approach may raise some security and privacy challenges. Malicious Internet hosts may spoof policy tags and congest network links or middleboxes, though these attacks can be blocked using conventional firewalls. In addition, changing a UE's local IP address each time it moves to a new base station would make it easier for Internet servers to infer the user's location. To address these concerns, SoftCell can perform network address translation (NAT) as packets arrive from the Internet. Specifically, we require the NAT function to pick a different IP address and/or port number for every flow, whether or not the UE moves. In addition, these public IP address and port pairs cannot be correlated with the UE's location (or with the decision to change locations). In practice, NATs are already extensively deployed today, as cellular providers are short of public IP addresses for each UE [24]. As such, SoftCell gives the same level of security and privacy protection as today's cellular networks.

4.2 Local Control Agent at the Access Edge

Sending the first packet of every microflow from the access switch to the controller would introduce high overhead. Instead, SoftCell introduces a local software agent running at each base station to scale the control plane. Note that the gateway switches don't perform fine-grained packet classification and thus do *not* need local agents, as the policy tags are piggybacked in the packet headers.

The local agent caches a list of *packet classifiers* for each UE at the behest of the controller. The packet classifiers are a *UE-specific* instantiation of the service policy that matches on header fields and identifies the appropriate policy tag, if a policy path already exists. When the UE arrives at the base station, the controller computes the packet classifiers based on the service policy, the UE's subscriber attributes, and the current policy tags. When the UE starts a new flow, the local agent consults these classifiers to determine the right policy tag for these packets, and installs a microflow rule in the access switch, similar to the DevoFlow "clone" function [17]. The local agent only contacts the controller if no policy tag exists for this flow—that is, if a packet is the first one at this base station, across all UEs, that needs a particular policy path.

For example, suppose UE7 arrives at base station 1 with prefix 10.0.0.0/16. The local agent first assigns a UE ID *10* to the UE. Now UE7 is associated with the location-dependent address 10.0.0.10. The local agent contacts the controller to fetch a list of packet classifiers for this UE. Suppose the list includes two packet classifiers:

```
1. match:dst_port=80, action:tag=2
2. match:dst_port=22, action:send-to-controller
```

When a packet with destination port 80 from UE7 arrives, the access switch does not find any existing microflow rule, and directs the packet to the local agent. The local agent determines that the traffic matches the first packet classifier. Since the policy path already exists, the local agent simply installs a microflow rule in the access switch which (i) rewrites the UE IP address to 10.0.0.10 and (ii) pushes "tag=2" to the source port number, without contacting the central controller. Suppose another packet arrives from UE7 with destination port 22. This flow matches the second packet classifier and the action is "send to controller", meaning that the policy path has *not* been installed yet. The local agent sends a request to the central controller to install a new policy path and return the policy tag. Then, the local agent can update the packet classifier and install a microflow rule for the packets of this flow.

In this way, local agents cache UE-specific packet classifiers and process most flows locally, significantly reducing the control-plane load on the controller.

5. HANDLING NETWORK DYNAMICS

In this section, we present how SoftCell handles network dynamics, mainly UE mobility and controller failure. SoftCell handles churn in a scalable fashion, through the clean division of labor between core and edge.

5.1 Policy Consistency Under Mobility

Seamless handling of device mobility is a basic requirement for cellular networks. UEs move frequently from one base station to another, and carriers have no control over when and where a UE moves. In addition to minimizing packet loss and delay, carriers must ensure that ongoing flows continue traversing the original sequence of middleboxes (though not necessarily the same switches), while reaching the UE at its new location. Such *policy consistency* is crucial for traffic going through stateful middleboxes, like firewalls and intrusion prevention systems. However, *new* flows should traverse middlebox instances closer to the UE's new location, for better performance. As such, SoftCell must differentiate between old and new flows, and direct flows on the appropriate paths through the network.

Differentiate between old and new flows: Incoming packets for old flows have a destination IP address corresponding to the UE's old location, so these packets naturally traverse the old sequence of middleboxes to the old base station. SoftCell merely needs to direct these packets to the new base station, which then remaps the old location-dependent address to the UE's permanent address. During the transition, the controller does not assign the old location-dependent address to any new UEs. For the traffic sent *from* the UE, the old access switch has a complete list of microflow rules for the active flows. Copying these rules to the new access switch ensures that packets in these flows continue

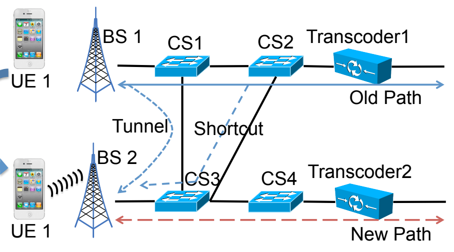


Figure 5: Tunnels and shortcuts for old flows

to use the old IP address, to avoid a disruption in service. Each UE has a relatively small number of active connections (say, 10), limiting the overhead of copying the rules. To minimize hand-off latency, the SoftCell controller could copy these rules in advance, as soon as a UE moves *near* a new base station.

Efficiently reroute the old flows: To handle ongoing connections during mobility events, SoftCell maintains long-lived *tunnels* between nearby base stations, as shown in Figure 5. These tunnels can carry traffic for *any* UEs that have moved from one base station to another. This "triangle routing" ensures policy consistency and minimizes packet loss, at the expense of higher latency and bandwidth consumption. The many short-lived connections would not experience any significant performance penalty. To handle long-lived connections more efficiently, the controller can establish temporary *shortcut* paths for directing traffic between the new base station and the old policy path, as shown in Figure 5. The controller can learn the list of active microflows from the access switch at the old base station, and install rules in the core switches to direct incoming packets over the shortcut paths. A single UE may need multiple shortcuts, since different traffic may go through different middleboxes. As such, these shortcut paths are created when a UE moves, and removed when a soft timeout expires—indicating that the old flow has ended.

5.2 Handling Control Plane Failures

We now describe how SoftCell handles control-plane failures. We focus on control-plane failure because the controller can easily handle topology changes (e.g., switch failures) by recomputing paths and modifying rules in the affected switches.

Handling controller failure: Controller failure is handled by maintaining a distributed, consistent copy of the control-plane state. SoftCell carefully divide the labor between the controller and local agents. The state of the central controller mainly includes: the service policy, the subscriber attributes, the policy paths, and the UE locations and local IP addresses. SoftCell enables fast failure recovery by simply replicating the controller. Indeed, the first three parts of the controller state change very slowly, making it affordable to maintain strong consistency. Also, although the UE locations change relatively often, a UE only associates with one base station at a time. Upon a controller failure, a replica can correctly rebuild the UE location state by querying local agents.

Handling local agent failure: The state of a local agent mainly includes: (i) the packet classifiers (generated from the service policy and subscriber attributes by the central controller), and (ii) the UEs' location-dependent IP addresses. Nonetheless, that state is never updated by the local agent (read-only to the local agent); only the central controller can update the state. Upon a local-agent failure, SoftCell restarts the local agent, which fetches the related state from the controller again. Observe that the impact of a failure is purely local and does not affect the agents at other base stations.

6. PERFORMANCE EVALUATION

In this section, we demonstrate the scalability and performance of our SoftCell architecture. We start by quantifying the control-plane load of a large LTE network using actual LTE traces (§ 6.1). We then perform micro-benchmarks of our SoftCell controller prototype and show that it can easily accommodate such a load (§ 6.2). Finally, we perform large-scale simulations and show that SoftCell can handle thousands of service policy clauses on commodity switches (§ 6.3).

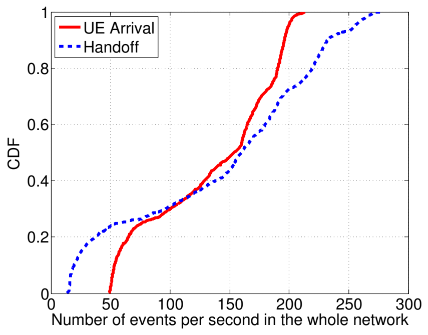
6.1 LTE Workload Characteristics

As a first step towards SoftCell deployment, we measured the workload of a real cellular network to understand the performance requirements of the controller. In contrast to other LTE measurement works, we study the aggregate arrival rates of UEs and flows and show the implications on control-plane load.

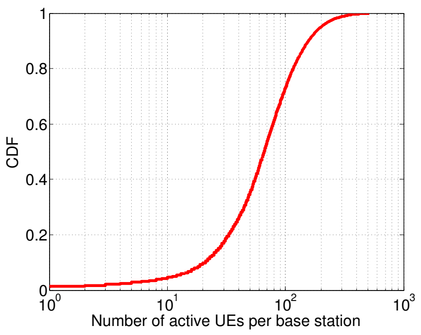
Dataset description: We collected about 1TB traces from a large ISP's LTE network during one week in January 2013. The dataset covers a large metropolitan area with roughly 1500 base stations and 1 million mobile devices (including mobile phones and tablets). The trace is *bearer*-level. A radio bearer is a communication channel between a UE and its associated base station with a defined Quality of Service (QoS) class. The trace includes various events such as radio bearer creation, UE arrival to the network, UE handoff between base stations, etc. When a flow arrives and there is an existing radio bearer with the same QoS class, the flow will use the existing radio bearer. Since radio bearers time out in a few seconds, a long flow may trigger several radio bearer creation and deletion events. Since we do not have flow-level information, we use radio bearers to estimate flow activities. We present measurement results for a typical weekday.

Network-wide characteristics: Figure 6(a) shows the CDF of UE arrivals and handoffs in the whole network. A UE arrival means a UE first attaches to the network (e.g., after a UE is powered on). A UE handoff event means a UE moves from one base station to another. From the figure, we can see that the 99.999 percentile of UE arrivals and handoffs per second are 214 and 280, respectively. As each of these events require the central controller to contact local agents (send packet classifiers) or update core switches (install short-cuts for long flows), it implies that the controller should be able to handle hundreds of such events per second.

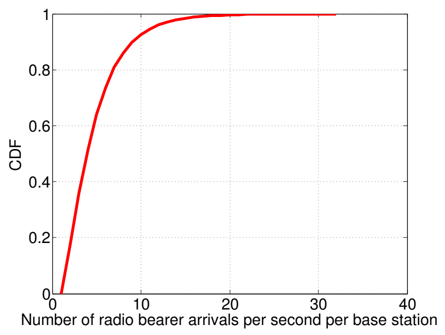
Load on each base station: Figure 6(b) shows the CDF of active UEs per base station. We see that a typical base station handles hundreds of active UEs with a 99.999 percentile of 514. Figure 6(c) depicts the radio bearer arrival rate at each base station. The number is relatively small, only 34 for the 99.999 percentile. As one radio bearer typically carries a handful of concurrent flows [25, 26], we expect



(a) Events in the whole network



(b) Active UEs per base station



(c) Radio bearer arrivals per base station

Figure 6: Measurement Results of a LTE network

the actual flow arrival rate to be around several hundred per second. These results imply that the local agent has to keep state for several hundred UEs and process a maximum of tens of thousands new flows per second. As most policy paths would have already been installed in the network, new flow requests only require the local agent to install packet classification rules at the access switch.

6.2 Controller Micro Benchmark

We have implemented a SoftCell control-plane prototype on top of the popular Floodlight [13] OpenFlow controller. The prototype implements both the SoftCell central controller and the SoftCell local agent. In the following, we perform micro-benchmarks on the prototype with Cbench [27], and compare the results with the measurement results in § 6.1. Cbench emulates a number of switches, generates packet-in events to the tested controller, and counts how many events the controller processes per second (throughput). Each test server has an Intel XEON W5580 processor with 8 cores and 6GB of RAM.

Central controller performance: First, we evaluate the throughput of the controller. Recall that the controller must send packet classifiers to local agents when a UE attaches or moves to a base station. We use Cbench to emulates 1000 switches and let these switches keep sending packet-in events to the controller. From the controller viewpoint, these packet-in events correspond to packet-classifier requests coming from 1000 of local agents. The controller then replies to these requests with packet classifiers as fast as it can. The result is that the controller can process 2.2 million of requests per second with 15 threads. This is more than enough to handle the hundreds of UE arrivals or handoffs per second for the LTE network in § 6.1.

Local agent performance: Second, we evaluate the local agent throughput. Recall that the local agent handles new flows based on packet classifiers fetched from the central controller. Thus its throughput depends on how frequently it needs to contact the central controller. Table 2 shows the evolution of the local agent throughput in function of the cache hit ratio. A cache hit ratio of 80% means that the local agent can handle 80% of the flows locally and needs to contact the central controller for the remaining 20%. From the table, it is easy to see that local agent throughput is sufficient to handle the number of new flows at a base station (a small tens of thousands per second). Indeed, even in the worst case where the local agent has to contact the controller for every flow, it is still able to handle 1.8K events per second. Further performance gains are possible by prefetching packet classifiers from the controller when a UE moves in range of the base station.

Cache Hit Ratio	0%	20%	40%	60%	80%	100%
Throughput	1.8K	2.3K	3.0K	4.5K	8.6K	505.8K

Table 2: Effect of cache hit ratio on local agent throughput

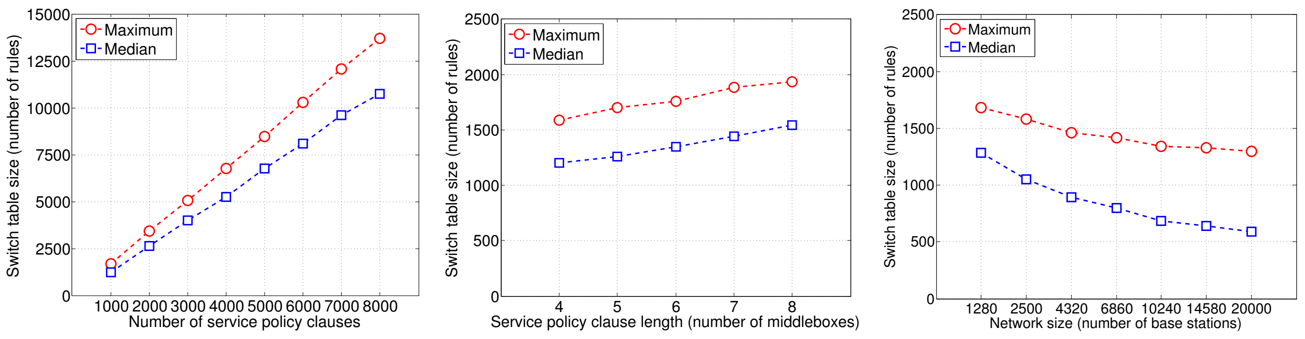
The result also validates the need to employ a hierarchical control plane. If there were no local agents, requests from those thousands of base stations would all go to the central controller. Such load (tens or hundreds of millions of requests per second) is difficult to handle by a single controller.

6.3 Large-Scale Simulations

We now demonstrate the scalability of the SoftCell data plane through large-scale simulations.

Methodology: We use synthetic topologies inspired by best current practices to design cellular networks (see e.g. [19, 28]). The topology has three layers: *access*, *aggregation* and *core*. The access layer consists of clusters of 10 base stations interconnected in a ring [28]. The aggregation layer consists of k *pods*, each of which has k switches connected in full-mesh. In each pod, $k/2$ switches are connected to $k/2$ base station clusters; the remaining $k/2$ switches are connected to $k/2$ switches in the core layer. The core layer has k^2 switches connected in full-mesh. They finally connect to a gateway switch. The whole topology with parameter k has $10k^3/4$ base stations. For example, $k = 8$ (resp. $k = 20$) gives a network with 1280 (resp. 20000) base stations. For each topology, we assume that there are k different types of middleboxes. We randomly connect one instance of each type in each pod in the aggregation layer and two instances of each type in the core layer. On top of this topology, we generate n policy paths for *each* base station to the gateway switch. A policy path traverses m randomly chosen middlebox instances. We use shortest-path algorithm to compute routes. Finally, we measure the number of rules in each switch flow table. In the base case, we consider $n = 1000$, $m = 5$ and $k = 8$. We vary k, n and m to show how the switch state is affected by the number of service policy clauses, the policy length and the network size, respectively.

Effect of number of service policy clauses: Figure 7(a) shows the maximum and median size of the switch forwarding table with respect to the number of service policy clauses. We can see that switch table size increases linearly with the



(a) Effect of the number of policy clauses (b) Effect of service policy clause length (c) Effect of network size

Figure 7: Large-scale simulation result. Thanks to multi-dimensional aggregation, SoftCell data plane is able to support thousands of service policy clauses on commodity switches.

number of service policy clauses with a small slope (less than 2). In particular, to support 1000 service policy clauses (1.28 million policy paths!), switches store a median of 1214 rules and a maximum of 1697 rules. This good performance is a direct consequence of the multi-dimensional aggregation (see § 3). It is true that one service policy clause may instantiate one policy path to every base station (thus thousands of policy paths for just *one* policy clause). But the corresponding entries can be aggregated like CS1 in Figure 3(c) if only one middlebox instance is used for this clause or like CS2 and CS3 if multiple instances are used.

Effect of service policy clause length: Figure 7(b) shows the switch table size with respect to the policy length. When the policy length is 8, the maximum table size is 1934. As before, we see that table size increases linearly with the policy length with a small slope. The reason is also similar. When a policy clause is longer, more middleboxes are traversed. But most affected switches on the path only require one additional rule that matches on the tag as CS1 in Figure 3(c); only a few switches require multiple rules to dispatch traffic to multiple middlebox instances as CS2 and CS3.

Effect of network size: Figure 7(c) shows the switch table size with respect to the network size. We see the table size decreases as the network grows. It is true that with more base stations, we have to install more policy paths for the same service policy clause, thus need more rules. But remember that we can do aggregation on policy tags and base station prefixes. The increase of rules is small due to aggregation, but all rules are now distributed over more switches as the network is larger. This leads to the result that when the network grows, switches maintain smaller tables for the same number of service policy clauses.

In summary, SoftCell can support thousands of service policy clauses in a network of thousands of base stations with a few thousand TCAM entries. The gain essentially comes from the ability to selectively match on multiple dimensions.

7. DISCUSSION

In this section, we give some discussions on relevant topics of SoftCell as follows.

Leveraging multi-table capabilities: As described in Section 3, the SoftCell data plane maintains mainly three types of flow entries: 1) matching both tags and IP prefixes, 2) matching tags only, and 3) matching IP prefixes only. SoftCell could leverage the multi-table capabilities offered by modern switches [9]. This is interesting as only Type 1 entries require the use of TCAMs (which are expensive and power hungry). In contrast, Type 2 and Type 3 entries can be stored in tables using exact match and IP-prefix match, respectively. In our design, Type 1 rules have the highest priority, followed up Type 2, and Type 3 rules have the lowest priority. This matches nicely with the priority of the multiple tables (TCAM higher than exact match which in turn is higher than longest prefix match) in some switch chipset [29].

Traffic initiated from the Internet: Although most traffic in cellular networks today are initiated from UEs, some carriers [30] also offer public IP address options. When a gateway switch receives packets destined to these public IP addresses, the gateway will act like an access switch. It will install packet classifiers that translate the public IP addresses and port numbers to LocIPs and policy tags. Note that these packet classifiers are not microflow rules and do not require communication with the central controller for every microflow. They are coarse-grained (match on the UE public IPs and port numbers) and can be installed once.

Mobile-to-mobile traffic: Mobile-to-mobile traffic is handled in a similar way as mobile-to-Internet traffic. There are two cases to distinguish. First, when the two UEs X and Y are not in the same cellular core network, then to X, Y is just another host on the Internet. Second, when X and Y are in the same cellular core network, SoftCell establishes a direct path between them without detouring via a gateway switch. Compared to today's cellular networks where all traffic has to go via the P-GW, SoftCell's routing scheme is more efficient.

Asymmetric Internet routing: For ease of description, we have assumed that flows leaving a gateway switch return to the same gateway switch. However, Internet routing is not guaranteed to be symmetric. If gateway switches are not the border routers peering with other autonomous systems, then the actual border routers can be configured to route return traffic to the same gateway switch. Otherwise, the controller can install corresponding switch rules for return traffic in all possible gateway switches (mostly a small fraction of all the gateway switches).

On-path middleboxes: One problem of on-path middleboxes is that it is unavoidable to traverse them in some cases. Therefore, if a service policy specifies that certain flows cannot traverse certain middleboxes (which we have not considered in our service policy), then the path computation has to avoid them. In case no feasible path exists, the policy path request will be denied.

Incremental deployment: SoftCell does not require any modifications to UEs or to the Internet. To incrementally deploy SoftCell in existing cellular networks, carriers can put a proxy at each base station. These proxies would serve as GTP tunnel end-points, allowing the core switches to carry normal IP packets. By doing so, the network between base stations and the Internet is an IP core that can be managed directly by the SoftCell controller. It is also possible to terminate GTP tunnels on modified OpenFlow switches [31]. Another solution is to use network virtualization like FlowVisor [32]. Carriers could then divide the network into two slices: one slice is handled by legacy protocols, and the other is handled by SoftCell. Initially, carriers can let SoftCell only handle a small fraction of the traffic. Then, they can gradually migrate the traffic to the slice handled by SoftCell. Finally, they can have SoftCell control all traffic and remove the first slice.

Inter-operation with LTE networks: To inter-operate with a LTE network for inter-system handoff, the SoftCell controller needs to communicate with the LTE network's eNodeBs and Mobility Management Entities (MMEs) using LTE protocols. In other words, SoftCell controller needs to implement S1-MME and S10 interfaces. To inter-operate with LTE Home Subscriber Servers (HSSs), SoftCell controller needs to implement the S6a interface.

8. RELATED WORK

Our quest is to build a scalable architecture to support fine-grained policies for mobile devices in cellular core networks. SoftCell differs from prior work as summarized on cellular network architecture, scalable data center, software defined networks, and middleboxes.

Cellular network architecture: Several recent works have exposed the complexity and inflexibility of cellular networks [5, 3] and several research efforts [33, 31, 34, 35, 36] have aimed at fixing the problems. CellSDN [33] presents the first high-level design of software-defined cellular networks that SoftCell fully develops. [31] proposes to integrate the support of GTP tunnels within OpenFlow. SoftCell could leverage such a support during partial deployment (see § 7). [34] introduces vertical forwarding which makes easier to forward with different protocols across different network layers with SDN. OpenFlow Wireless [35] focuses on virtualizing the data path and configuration. Unlike SoftCell, none of them present a scalable network architecture for fine-grained policies. SoftRAN [36] uses SDN principles to redesign the radio access network. It is therefore complementary to SoftCell which focuses at redesigning the core network instead.

Scalable data centers: Our addressing scheme shares some similarity to prior work on scalable data centers [10, 11, 12]. However, they mainly deal with "east west" traffic. For instance, [12] requires intelligent Internet gateways to deal with Internet traffic. In contrast, our gateways are much simpler because we "embed" policy and location information in the packet header, rather than relying on the controller to install fine-grained packet-classification rules at gateway switches. Also, these works do not present techniques for enforcing service policies.

Software defined networks: Recent work like DevoFlow [17] and DIFANE [37] improves upon Ethane [7] by moving some processing from the control plane to the data plane. However, their techniques cannot address specific cellular network requirements like fine-grained policies, or policy consistency under mobility. Fabric [38] and SDIA [39] describe the idea of core/edge separation, which argues to put most intelligence at the edge and keep the core simple. Although similar in spirit, SoftCell introduces the concept of *asymmetric edge* and gives a novel solution of "smart access edge, dumb gateway edge" to scale the system. SoftCell also gives a specific solution in the core to support fine-grained service routing, which is not addressed in Fabric and SDIA.

Middleboxes: There have been many works on middleboxes recently (e.g., see [18, 40, 41, 42, 43, 44]). The closest works to SoftCell are PLayer [18] and SIMPLE [43]. PLayer is a pioneering work on how to enforce flexible middlebox traversals. SIMPLE takes a further step to enable better load balancing and support middleboxes that modify packets. SoftCell differs from both them by identifying the specific challenges to enable fine-grained policies for large cellular networks and providing novel techniques to make the system scalable.

9. CONCLUSION

Today's cellular core networks are expensive and inflexible. In this paper, we propose SoftCell, a scalable architecture for supporting fine-grained policies in cellular core networks. SoftCell achieves scalability in the data plane by (i) pushing packet classification to low-bandwidth access switches and (ii) minimizing the state in core network through effective, multi-dimensional aggregation of forwarding rules. SoftCell achieves scalability in the control plane by caching packet classifiers and policy tags at local agents that update the rules in the access switches. Our prototype and evaluation demonstrate that SoftCell significantly improves the scalability and flexibility of cellular core networks.

10. ACKNOWLEDGMENTS

We would like to thank the CoNEXT reviewers, our shepherd Steve Uhlig, Harish Viswanathan, Olivier Bonaventure, Srinivas Narayana, and Naga Praveen Katta for their feedback on earlier versions of this paper. This work was supported by the NSF under grants 1040705 and 1247764.

11. REFERENCES

[1] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "A first look at cellular machine-to-machine traffic: Large scale measurement and characterization," in *ACM SIGMETRICS*, June 2012.

[2] Cisco, "Cisco visual networking index forecast projects 13-fold growth in global mobile internet data traffic from 2012 to 2017." `http://newsroom.cisco.com/release/1135354`.

[3] S. Elby, "Carrier vision of SDN and future applications to achieve a more agile mobile business," October 2012. Keynote at the SDN & OpenFlow World Congress.

[4] "Cisco PGW packet data network gateway." http://www.cisco.com/en/US/products/ps11079/index.html.

[5] B.-J. Kim and P. Henry, "Directions for future cellular mobile network architecture," *First Monday*, vol. 17, December 2012.

[6] "Network functions virtualization: Introductory white paper," October 2012. http://www.tid.es/es/Documents/NFV_White_PaperV2.pdf.

[7] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, "Rethinking enterprise network control," *IEEE/ACM Transactions on Networking*, vol. 17, August 2009.

[8] R. Wang, D. Butnariu, and J. Rexford, "OpenFlow-based server load balancing gone wild," in *Hot-ICE Workshop*, March 2011.

[9] B. Stephens, A. Cox, W. Felter, C. Dixon, and J. Carter, "PAST: Scalable Ethernet for data centers," in *ACM SIGCOMM CoNext Conference*, December 2012.

[10] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *ACM SIGCOMM*, August 2009.

[11] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: A scalable fault-tolerant layer 2 data center network fabric," in *ACM SIGCOMM*, August 2009.

[12] VMware NSX, "The platform for network virtualization." https://www.vmware.com/files/pdf/products/nsx/VMware-NSX-Datasheet.pdf.

[13] "Floodlight OpenFlow Controller." http://floodlight.openflowhub.org/.

[14] "Open vSwitch." http://openvswitch.org/, 2013.

[15] "The rise of soft switching, part II: Soft switching is awesome," June 2012. http://networkheresy.com/2011/06/25/the-rise-of-soft-switching-part-ii-soft-switching-is-awesome-tm/.

[16] "Broadcom Trident chipset." http://www.broadcom.com/products/Switching/Data-Center/BCM56850-Series.

[17] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "DevoFlow: Scaling flow management for high-performance networks," in *ACM SIGCOMM*, August 2011.

[18] D. Joseph, A. Tavakoli, and I. Stoica, "A policy-aware switching layer for data centers," in *ACM SIGCOMM*, August 2008.

[19] M. Howard, "Using carrier Ethernet to backhaul LTE," *Infonetics Research White Paper*, 2011.

[20] L. Whitney, "Ericsson demos faster LTE speeds of almost 1Gbps." http://news.cnet.com/8301-1035_3-20075328-94/ericsson-demos-faster-lte-speeds-of-almost-1gbps/.

[21] A. Takacs, E. Bellagamba, and J. Wilke, "Software-defined networking: The service provider perspective," in *Ericsson Review*, February 2013.

[22] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "SoftCell: Taking control of cellular core networks," Tech. Rep. TR-95-13, Princeton University CS, May 2013.

[23] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *ACM SIGCOMM*, August 2012.

[24] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *ACM SIGCOMM*, August 2011.

[25] A. Rahmati, C. Shepard, C. Tossell, A. Nicoara, L. Zhong, P. Kortum, and J. Singh, "Seamless flow migration on smartphones without network support," *IEEE Transactions on Mobile Computing*, 2013. To appear.

[26] Y. Zhang and A. Arvidsson, "Understanding the characteristics of cellular data traffic," in *ACM SIGCOMM CellNet Workshop*, August 2012.

[27] "Cbench OpenFlow Controller Benchmark." http://www.openflow.org/wk/index.php/Oflops.

[28] R. Nadiv and T. Naveh, "Wireless backhaul topologies: Analyzing backhaul topology strategies," *Ceragon White Paper*, 2010.

[29] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, and Y. Zhang, "Serverswitch: a programmable and high performance platform for data center networks," in *USENIX NSDI*, 2011.

[30] AT&T, "Wireless IP options for mobile deployments." https://www.wireless.att.com/businesscenter/solutions/connectivity/ip-addressing.jsp.

[31] J. Kempf, B. Johansson, S. Pettersson, H. Luning, and T. Nilsson, "Moving the mobile evolved packet core to the cloud," in *IEEE WiMob*, October 2012.

[32] R. Sherwood, G. Gibb, K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed," in *Operating Systems Design and Implementation*, USENIX, 2010.

[33] L. Li, Z. Mao, and J. Rexford, "Toward software-defined cellular networks," in *EWSDN*, October 2012.

[34] G. Hampel, M. Steiner, and T. Bu, "Applying software-defined networking to the telecom domain," in *IEEE Global Internet Symposium*, April 2013.

[35] K.-K. Yap, R. Sherwood, M. Kobayashi, T.-Y. Huang, M. Chan, N. Handigol, N. McKeown, and G. Parulkar, "Blueprint for introducing innovation into wireless mobile networks," in *ACM VISA Workshop*, August 2010.

[36] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *ACM SIGCOMM HotSDN Workshop*, August 2013.

[37] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with DIFANE," in *ACM SIGCOMM*, August 2010.

[38] M. Casado, T. Koponen, S. Shenker, and A. Tootoonchian, "Fabric: A retrospective on evolving SDN," in *ACM SIGCOMM HotSDN Workshop*, August 2012.

[39] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker, "Software-defined Internet architecture: Decoupling architecture from infrastructure," in *ACM SIGCOMM HotNets Workshop*, October 2012.

[40] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi, “The middlebox manifesto: Enabling innovation in middlebox deployment,” in *ACM SIGCOMM HotNets Workshop*, 2011.

[41] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, “Design and implementation of a consolidated middlebox architecture,” in *Networked Systems Design and Implementation*, April 2012.

[42] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, “Toward software-defined middlebox networking,” in *ACM SIGCOMM HotNets Workshop*, 2012.

[43] Z. Qazi, C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, “SIMPLE-fying middlebox policy enforcement using SDN,” in *ACM SIGCOMM*, August 2013.

[44] S. Fayazbakhsh, V. Sekar, M. Yu, and J. Mogul, “FlowTags: enforcing network-wide policies in the presence of dynamic middlebox actions,” in *ACM SIGCOMM HotSDN Workshop*, August 2013.