# KLEIN: A Minimally Disruptive Design for an Elastic Cellular Core

Zafar Ayyub Qazi
Stony Brook University

Phani Krishna Penumarthi
Stony Brook University

Vyas Sekar
Carnegie Mellon University

Vijay Gopalakrishnan
AT&T Labs Research

Kaustubh Joshi
AT&T Labs Research

Samir R. Das
Stony Brook University

## ABSTRACT

Today's cellular core, which connects the radio access network to the Internet, relies on fixed hardware appliances placed at a few dedicated locations and uses relatively static routing policies. As such, today's core design has key limitations—it induces inefficient provisioning tradeoffs and is poorly equipped to handle overload, failure scenarios, and diverse application requirements. To address these limitations, ongoing efforts envision "clean slate" solutions that depart from cellular standards and routing protocols; e.g., via programmable switches at base stations and per-flow SDN-like orchestration. The driving question of this work is to ask if a clean-slate redesign is necessary and if not, how can we design a flexible cellular core that is minimally disruptive. We propose KLEIN, a design that stays within the confines of current cellular standards and addresses the above limitations by combining network functions virtualization with smart resource management. We address key challenges w.r.t. scalability and responsiveness in realizing KLEIN via backwards-compatible orchestration mechanisms. Our evaluations through data-driven simulations and real prototype experiments using `OpenAirInterface` show that KLEIN can scale to billions of devices and is close to optimal for wide variety of traffic and deployment parameters.

## Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]; C.2.1 [**Network Architecture and Design**]: Centralized Networks; C.2.3 [**Network Operations**]: Network Management

## Keywords

Middlebox, Network Function Virtualization, Network Management, Software-Defined Networking

## 1 Introduction

Over the last few years, we have observed explosive growth in mobile Internet-connected devices, spurred by the commoditization of smartphones, tablets, and other devices [25]. Reports suggest that mobile traffic volumes are poised to surpass traditional fixed-line Internet usage for many applications [17, 23, 21]. Furthermore, with the onset of Internet-of-Things deployments, analysts predict orders of magnitude more devices connected via cellular networks with diverse application demands.

This dramatic growth in volume and application diversity creates significant stresses on the *cellular core*—the operator's network between the radio access and the egress to the global Internet. The cellular core is a critical piece of the infrastructure which provides key cellular-specific data plane functions such as the Serving Gateway (S-GW) and Packet Data Network Gateway (P-GW) and various IP- and application-layer middlebox services (e.g., firewalls, proxies, and transcoders). Today, such functions are deployed using expensive and fixed function "big-iron" appliances [46]. These appliances are typically concentrated in a small number of datacenter sites in the operator's backbone and user traffic is routed to the nearest datacenter using standard cellular procedures; e.g., using 3GPP standards [48].

Unfortunately, this current architecture results in fundamental sources of *inelasticity*, which in turn hurts costs, application performance, and evolvability [44]. (We elaborate in §2). For instance, the fixed capacity of the hardware forces operators to make provisioning decisions that lead to both significant underutilization, and an inability to handle unanticipated changes in the workload such as flash crowds, failures, and signaling storms (e.g., [26, 15]). This architecture also creates inefficient tradeoffs between provisioning cost and latency considerations; i.e., consolidation lowers cost via statistical multiplexing but inflates paths vs. disaggregation to reduce path lengths escalates costs as each site needs to be provisioned for peak loads.

Now, it is possible to address these sources of inelasticity using a clean-slate approach that fundamentally refactors how the cellular core is designed, provisioned, and managed. Indeed, several recent efforts (e.g., [34, 39]) have demonstrated the promise of such clean-slate architectures that argue for ubiquitous deployment of core functions and suggest that we need per-flow SDN-like mechanisms, using new "smart" switches at every base station.

The driving question behind our work is to ask if a clean-slate redesign is fundamentally necessary or if we can address these aforementioned limitations of cellular core networks in a *minimally disruptive* manner.

To address this question, we use data from a large cellular carrier to quantitatively evaluate three candidate cellular core designs: (1) TODAY's fixed hardware approach using 3GPP compliant routing; (2) A hypothetical INTERMEDIATE design that uses network functions virtualization (NFV), requires no changes to existing cellular signaling and core routing, and performs dynamic load distribution; and (3) A CLEANSLATE solution that uses NFV but is not constrained to be 3GPP compatible and can use fine-grained per-flow routing. Our analysis shows (perhaps surprisingly) that



*SOSR '16, March 14-15, 2016, Santa Clara, CA, USA*



DOI: http://dx.doi.org/10.1145/2890955.2890971

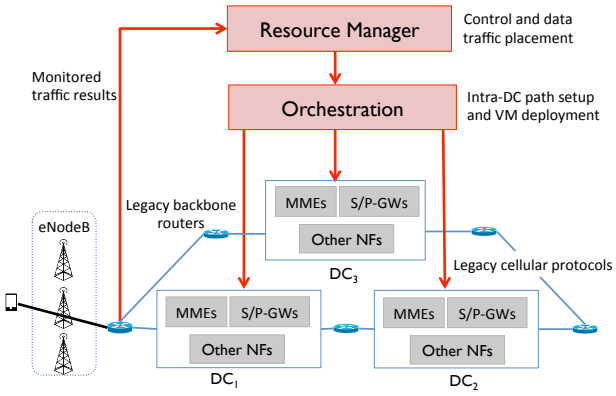


**Figure 1: KLEIN overview.**

the INTERMEDIATE design achieves close-to-optimal provisioning tradeoffs and load balancing objectives relative to the CLEANSLATE approach.

We thus argue that this INTERMEDIATE design can serve as the basis for a *minimally disruptive* design for future cellular core architectures that can address today's cellular core limitations. In particular, NFV is already a reality for carriers [7, 24, 6, 44], and there are many open-source and commercial efforts to virtualize cellular core functions [13, 19, 18, 5, 2, 3].

Building on these insights, we design KLEIN (Figure 1),[1] which provides a practical realization of this above INTERMEDIATE design. Specifically, KLEIN extends existing cellular core in two minimally disruptive ways: (1) use of virtualized EPC functions together with (standard) SDN mechanisms for service chaining inside the datacenters and (2) a global resource management scheme for mapping devices' traffic to different datacenter locations. KLEIN is 3GPP-compliant and requires no changes to existing cellular signaling and core routing.

This paper addresses two key challenges to translate the hypothetical INTERMEDIATE design into reality. First, we design a responsive resource management layer that can handle billions of devices and thousands of data centers. Second, we engineer backwards-compatible network orchestration mechanisms to realize these dynamic resource management decisions.

We prototype KLEIN using the open source `OpenAirInterface` [18] platform. We use `Floodlight` [10] and custom controllers for the KLEIN control plane to manage the core network. We validate KLEIN using a range of trace-driven and real testbed experiments. We find that: (a) KLEIN is scalable: it takes less then 20 s to reconfigure the load with 2000 data centers and 5 billion devices; (b) KLEIN is close to optimal — within 10% of an ideal CLEANSLATE for different traffic mix and latency budgets; (c) KLEIN can improve end-application performance by a factor of 5; and (d) KLEIN can handle data center failures both rapidly and efficiently, taking less than 2.3 s and reducing the maximum data center load by a factor of 2.

**Contributions and Roadmap:** In summary, this paper makes the following contributions:

- An empirical demonstration of key limitations of today's cellular core with respect to cost-performance tradeoffs (§2).
- A data-driven design space exploration (§3) that shows that it is indeed possible to address these limitations with a minimally disruptive design.
- A practical architecture (§4), with a responsive and scalable resource management algorithm that can handle billion of devices and thousand of sites (§5) and backwards compatible orchestration mechanisms (§6).
- A proof-of-concept implementation (§7), to show the benefits of elastically scaling and balancing load on virtualized EPC functions.

We discuss related work (§9) before concluding in §10.

[1]The name is inspired by Yves Klein, a pioneering artist in the Minimal art movement, https://en.wikipedia.org/wiki/Yves_Klein. The name also means "small" or little in German which is indicative of the change we mandate.

## 2 Background and Motivation

In this section, we use data from a nationwide cellular provider to highlight the limitations of today's cellular core networks. Before doing so, we provide a brief introduction to the cellular network architecture and how the cellular core is implemented today.

### 2.1 Cellular Core Background

The 3GPP LTE cellular network (See Figure 1) consists of two main components: the LTE Radio Access Network (RAN), and the Evolved Packet Core (EPC). The RAN consists of the eNodeB (i.e., base station), which communicates with the User Equipment (UE) via the radio link and then forwards packets to the eventual destination via the EPC.

The main network elements in the EPC include the MME (mobility management unit), S-GW, and P-GW (serving and packet gateways). The MME is responsible for all *control plane* messaging including user authentication, session establishment and release, and mobility management. The S-GW and P-GW provide the *data plane* functions, specifically packet routing/forwarding, traffic management and accounting, and policy enforcement. The EPC also includes middleboxes (e.g., NATs, firewalls and proxies) that are traversed before a packet reaches the Internet [48, 27]. Today, these network functions are realized via dedicated hardware appliances deployed in a small number of central data centers (DCs). Traffic from a UE is statically mapped to one of these data centers; e.g., the nearest data center. Between the eNodeBs and the data centers, there are a large number of small data centers (e.g., central offices) [12], which act as aggregation points for traffic from eNodeBs. There is a natural hierarchy in the deployment, where a geographical region usually consists of a large data center, and many small data centers.

### 2.2 Limitations of Today's Cellular Core

We highlight three key limitations of today's cellular core using data collected at the core network of a major provider. We start by describing the data set.

**Dataset description:** We use load traces collected for several months during 2014-2015 at tens of thousands of base stations of a large cellular provider in the US. The dataset gives a time series of data traffic volumes at 5 minute intervals at each base station, for each 'APN', and 'GW device'. APN refers to one or more collection of services (e.g., Voice, Data, M2M). GW device refers to the actual hardware appliance in the datacenter that runs an EPC datapath element (S-GW or P-GW) that processes the corresponding data. For each device, we have the information about the corresponding data center the GW is located in. The data set covers tens of data centers and hundreds of APNs. No user data or any personal information that identifies individual users are collected.

Next, we focus on specific set of analyses on this data set to highlight specific limitations.

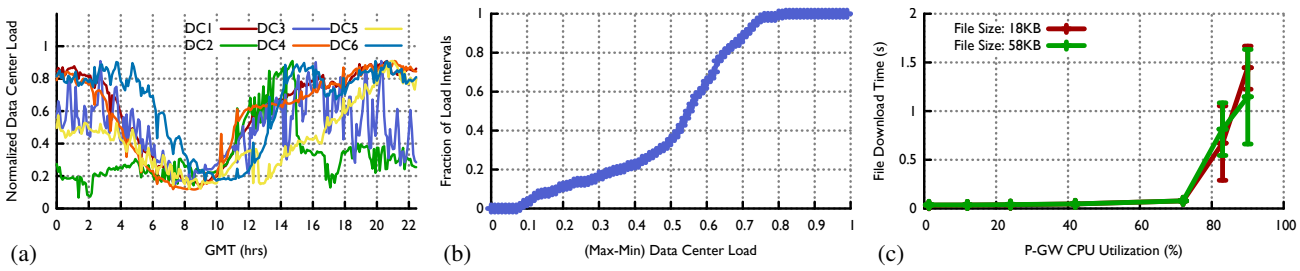


**Figure 2: Load imbalance and potential impact on applications: (a) one sample snapshot of load distribution across DCs; (b) CDF of (max-min) DC loads over all snapshots; (c) impact of EPC load on file download time**

**(1) Impact on applications:** The fixed nature of provisioning and static routing in the cellular core causes load imbalance, which in turn could impact user-perceived application performance. Figure 2(a) shows the imbalance in S/P-GW loads across data centers for one entire day. Since the data centers capacities vary, the load of each data center is normalized by the peak load seen in that data center in the entire data set. This quantity serves as proxy for the capacity. We observe that the difference between the minimum and maximum data center utilization could be as high as 80%.

Such load imbalance is not an anomaly. Figure 2(b) shows the distribution of the difference between maximum and minimum normalized data center loads over all snapshots collected over a 2 month period. We observe that the difference is more than 50% for more than 60% of the times. Also, for about 15% of the time, load imbalance is more than 70%. This means that one data center is maximally loaded while another is almost free.

Such load imbalance can potentially hurt application performance. To demonstrate this, we benchmarked the impact of EPC load on file download times on a software EPC testbed (Phantomnet [28]). Here, we increase the P-GW load by generating background traffic, and observe the impact on downloading files of two different sizes (16 KB and 58 KB) over a TCP connection. We observe in Figure 2(c) that when P-GW utilization is $\geq$ 80%, the file download time is more than an order of magnitude higher as compared to the case when P-GW utilization is <70%. A high EPC load clearly hurts user-perceived performance. Extreme load imbalance demonstrated in Figure 2(b) means that such performance metrics would improve if flexible provisioning could be made available.

**(2) Resource provisioning:** At the same time, today's networks are significantly over-provisioned. We compare *sum of peak* loads on individual S/P-GW devices with *peak of sum* loads on these devices. The ratio of sum of peak vs. peak of sum is a good indicator of resource over-provisioning. We observe that the sum of peak GW/DC load is about 1.7 times the corresponding peak aggregate load respectively. This means at most only 60% of provisioned resources are utilized at any time.

**(3) Provisioning cost vs. wider deployment:** Today's networks suffer from path inflation as all UE data traffic has to be routed to one of the few large centralized data centers [48]. Wider deployment of processing sites can improve latency,[2] but due to the fixed nature of mapping traffic to processing sites this can happen only at the expense of higher provisioning cost. Figure 3 demonstrates this. The plot here assumes various number of data centers. The locations of such data centers are assumed close to the eNodeBs they meant to serve. This is done by a nearest-neighbor clustering of eNodeBs in the 2D space and locating the data centers at the centroid of such clusters. We map the traffic to the nearest data center and compute the sum of peak loads on the data centers as the provisioning cost. The plot shows that in order to get an order of magnitude improvement in latency, for example, the cost will increase 5 fold (while from 10 to 500 data centers).

[2]This can be done by expanding the use of small on path data centers e.g., central offices. (See Section 2.1).

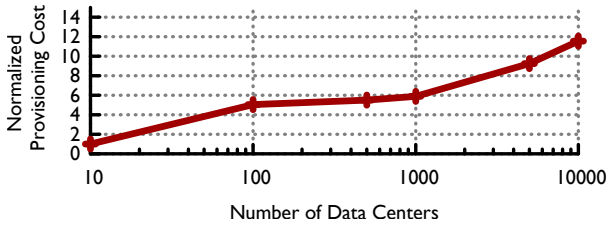


**Figure 3: Provisioning cost vs. number of data centers with static provisioning and routing.**

$$Minimize \sum_{d} Prov_d, subject\ to \quad (1)$$

$$\forall d, e : Load_{d,e} < Prov_d \quad (2)$$

$$\forall d, e : Load_{d,e} = \sum_{a,d,e,i} f_{a,d,i,e} \times T_{a,e,i} \times F_a \quad (3)$$

$$\forall a, d, e, i : f_{a,d,e,i} \in [0, 1] \quad (4)$$

$$\forall d : Prov_d < Cap_d \quad (5)$$

$$\forall a, e, i : T_{a,e,i} = \sum_{d,e} f_{a,d,e,i} \times T_{a,e,i} \quad (6)$$

$$\forall a, d, e, i : Latency_{d,i} \times f_{a,d,e,i} < Budget_a \quad (7)$$

**Figure 4: Linear Program (LP) formulation for CLEANSLATE.**

## 2.3 Summary of Key Observations

In summary, our data-driven analysis shows that:

- Load imbalance in cellular core network can be as high as 80%. This could impact application performance by as much as 7 fold.
- At most only 60% of the core network compute resources are utilized at any time.
- For wider cellular core deployments e.g., 500 data centers, provisioning cost is 5× higher than in the case of 10 data centers.

# 3 A Case for a Minimalistic Roadmap

The previous analysis shows the limitations in today's cellular core. These can potentially be addressed by a clean slate redesign - a gen-

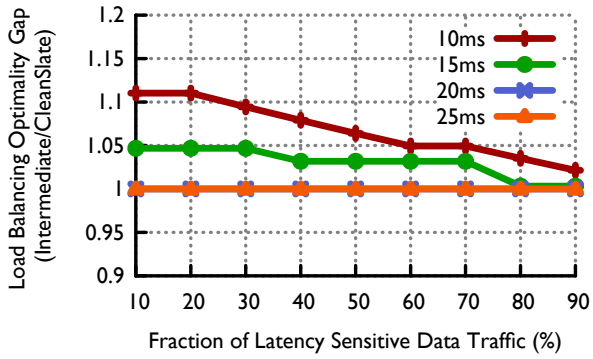


**Figure 5: The load balancing optimality gap between INTERMEDIATE and CLEANSLATE.**

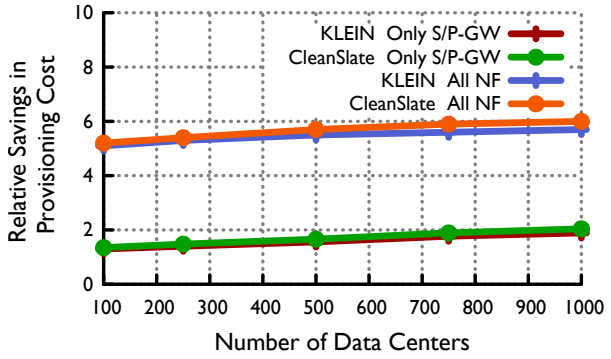


**Figure 6: Reduction in provisioning cost with INTERMEDIATE and CLEANSLATE.**

eralization of recent work in [34, 39] - that distributes the processing resources widely and performs a fine grained (such as per-flow in the ideal case) dynamic mapping of traffic to such resources. While an approach like this could be optimal, this also requires a fundamental redesign of the cellular core. In this section we demonstrate – using a similar data-driven analysis as before – that such a disruptive redesign is unnecessary and an intermediate 3GPP compatible design can be used to address these limitations.

## 3.1 Design space

We consider a broad design space characterized by four dimensions: (1) *Implementation Platforms:* We could have each network function running on a fixed, hardware-based appliance or as a virtualized/software appliance; (2) *Routing granularity:* packets can be routed across the cellular core network e.g., per-flow (e.g., SDN-like) routing vs per-UE tunnels (conforming 3GPP); and (3) *Resource management:* How the traffic from different UEs and APNs are allocated to the available compute and network resources.

Given the dimensions of this design space, we consider three concrete instances.

- TODAY's network deployment, where (1) fixed, hardware based appliances are used to implement the EPC functions, (2) routing of flows is done at a per-UE granularity and (3) the resource management is static and each UE is routed to the nearest data center.
- At the opposite extreme, we consider a CLEANSLATE approach, where (1) network functions are virtualized, (2) fine-grained per-flow routing (which is in conflict with 3GPP constraints), and (3) dynamic resource management. This a logical extension of recent clean slate designs [34, 39]; the key addition is that we assume some form of optimal resource management scheme which these prior efforts largely ignore.
- Finally, we consider an intermediate approach that we call INTERMEDIATE. INTERMEDIATE attempts to preserve the benefits of CLEANSLATE while also preserving full 3GPP compatibility. Here, (1) the network functions are still virtualized; but (2) routing decisions are at a per-UE granularity rather than the per-flow approach in CLEANSLATE; (3) dynamic resource management is used to re-balance the load as necessary.

## 3.2 Methodology

In order to compare the three design points, we conduct a data-driven study, using same data as in (Section 2.2). We devise linear programming based optimizations for modeling CLEANSLATE and INTERMEDIATE designs to evaluate the provisioning and load balancing benefits. Below we describe the simulation setup and optimizations.

**Simulation Setup:** We classify APNs into 3 different groups: Data, M2M, and Voice. For the first class, we assume that the traffic can be further divided into latency-sensitive and latency-tolerant applications. We vary the mix of latency sensitive data traffic and the delay budget for latency sensitive applications. The input to the simulations is the data traffic volume corresponding to different traffic classes from the data set.

**Optimizations:** In the CLEANSLATE design, we assume that the traffic can be split from each base station and application class, and routed to different data centers. We formulate CLEANSLATE as a linear program, with the objective of minimizing the total resources provisioned in the core network (Eq (1)). $Prov_d$ corresponds to the capacity to be provisioned in a data center $d$ . The key decision variable in CLEANSLATE is a fractional variable $f_{a,d,e,i}$ which gives the fraction of traffic from base station $i$ and application $a$, that should be routed to the data center $d$, in a time epoch, $e$. Figure 4 shows the LP. It includes the constraints that all the traffic should be served Eq (6), and that the latency budget for each traffic class should satisfied Eq (7). $Load_{d,e}$ is the load on a data center $d$ in an epoch $e$, $T_{a,e,i}$ is the traffic volume from eNodeb $i$ for traffic class $a$ in epoch $e$, $Cap_d$ is the maximum capacity that can be allocated to data center $d$, $F_a$ is the total resource footprint of an application class $a$, $Latency_{d,i}$ is the latency from the eNodeb $i$ to data center $d$, and $Budget_a$ is the latency budget for application class $a$.

For INTERMEDIATE, we map each base-station's load to a single data center. This is because of the 3GPP constraint that at any time we can only have a single SGW attached to a UE. We model INTERMEDIATE as an ILP with the objective of minimizing the total resources provisioned in the core network – the sum of resources provisioned in each data center. However, the key decision variable in this case is a binary variable $b_{a,d,e,i}$ which gives mapping of traffic from base station $i$, application $a$, that should be routed to the data center $d$, with the additional constraint that all the traffic from a base station $i$ is routed to the same data center.

We also consider a load balancing exercise, where we minimize the maximum data center utilization for CLEANSLATE and INTERMEDIATE. The formulations are similar, except (1) the objective is minimizing $MaxDCUtil$, where $MaxDCUtil$ is the utilization of the most utilized data center in the core network, (2) provisioned capacity, $Prov_d$ at each data center $d$ is fixed, and (3) the key decision variables are not per-epoch: $f_{a,d,i}$ and $b_{a,d,i}$. We use CPLEX to solve these optimization problems.

## 3.3 Results

**Load balancing:** We first consider a load balancing objective, where we try to minimize the maximum utilization of any data center for CLEANSLATE and INTERMEDIATE, and evaluate the optimality gap: $\frac{Intermediate}{CleanSlate}$. We vary the mix of traffic (delay sensitive and delay tolerant) and the latency budgets of the applications. We consider 4 hours of peak load on a week day in November and reconfigure the traffic routing every 5mins. We take the maximum load observed for each data center over the 4 hour period for both CLEANSLATE and INTERMEDIATE. Figure 5 shows the optimality gap with different delay budgets and traffic mix. The key takeaway is that for all latency budgets, INTERMEDIATE performs close to the CLEANSLATE design. More concretely, we observe an optimality gap of about 5-10%, if the latency budget is 10ms and fraction of latency-sensitive traffic is less than 70%. For all other cases, the optimality gap is less than 5%, and it converges to close to 0 if the fraction of latency sensitive traffic is 90% or more. The reason the optimality gap is reducing as the fraction of latency-sensitive traffic increases is because then even clean slate has very few opportunities for balancing load across sites. Since most traffic is latency constrained, there are only a few sites where the traffic can be sent to.

**Reduction in provisioning cost:** Next, we consider a provisioning exercise to minimize the resources needed to handle the time varying traffic patterns generated across a week. The metric of interest here is the relative savings that INTERMEDIATE and CLEANSLATE provides vs. today's deployment model where all traffic is usually routed to the nearest data center: $\frac{Cost_{Intermediate/CleanSlate}}{Cost_{Today}}$. We observe that with increasing number of data centers, we can achieve similar benefits in resource savings in Figure 6. We can save as much as 2 times more S/P-GW resources with INTERMEDIATE and CLEANSLATE as compared to Today's static architecture, and as 6 times more total resources, if we also consider other network functions.[3]. CLEANSLATE and INTERMEDIATE achieves these benefits while satisfying the latency budgets for data traffic.

## 3.4 Summary

The data-driven simulations show that:

- INTERMEDIATE is within 10% of CLEANSLATE in terms of load balancing for a variety of traffic mix and latency budgets.
- Both CLEANSLATE and INTERMEDIATE need 6× less resources as compared to Today for a cellular core with 1000 data centers.

We argue that this INTERMEDIATE approach can form the basis of a minimally disruptive cellular core. We show in the following sections how such an approach can be implemented using only minimal changes in cellular infrastructure while assuring complete legacy compliance. This makes it a very attractive option for the carriers.

# 4 System Overview and Challenges

The previous section showed that we can potentially address most of the limitations of the cellular core today through a hypothetical INTERMEDIATE design, which is 3GPP compliant. In this section, we give an overview of KLEIN, a practical instantiation of the INTERMEDIATE design. We discuss our envisioned core network deployment, the challenges in realizing KLEIN and outline our key ideas to address these challenges.

[3]In the case of "ALL NF" in Figure 6, we also took into account placement of middleboxes other than S/P-GW (e.g., Firewalls, Transcoder etc.) by assuming hypothetical service chains for different traffic classes.

| Number of UEs | Number of data centers | Computation Time |
|---|---|---|
| ∼50,000 | 6 | ∼20s |
| ∼50,000 | 100 | ∼500s |
| ∼50,000 | 1000 | >1 day |

**Table 1: Scalability with a 2-level resource management decomposition.**

## 4.1 Overview

We envision that the cellular carrier has deployed many data centers. We assume that the cellular carrier can have both large data centers and small data centers. Large nation wide carriers already have a few thousand central offices [12]. We assume data centers are connected from the base station via traditional backbone routers and inter-connecting links. Each data center has commodity hardware servers and runs virtualized EPC functions (e.g., MME, S/P-GW) and other network functions (e.g., Firewall, NAT).

Our objective is to dynamically distribute the network load and provision virtualized network functions over the provider's available compute/network resources. This requires a resource manager which deals with load distribution and placement of virtualized network functions. For achieving these we want to construct a backwards compatible network orchestration layer, which is compatible with existing 3GPP mechanisms and requires minimal changes to the existing core network.

As seen in Figure 1, KLEIN extends the existing cellular core in three simple ways: (1) virtualized network functions instead of fixed hardware appliances, (2) resource manager, which performs dynamic resource management, and (3) a backwards-compatible network orchestration layer for implementing the output of the resource manager. We argue that each of these changes is minimally disruptive. First, for (1), we observe that the use virtual functions is already on several carrier roadmaps [7, 24, 6, 44]. Second, (2) is a "bolt-on" control component that does not require any additional network infrastructure. Finally, for (3), we observe that in contrast with CLEANSLATE, KLEIN does not require changes to the existing 3GPP mechanisms or core network routing.

## 4.2 Challenges

**(1) Responsive resource management:** The two key challenges in designing a responsive and scalable resource manager are:

- First, the problem size makes it difficult to develop a responsive and scalable resource manager, for instance, a nationwide cellular carrier in the nearby future may have billions of devices and few thousand data centers. Specifically, this entails solving a large optimization problem, which cannot scale to such input size even with state-of-the-art solvers. Even a 2-level decomposition does not scale, as shown in Table 1. It takes >1 day for it to reconfigure the load, even for a small input size of 50,000 UEs and 1000 data centers.
- Second, the resource manager has to instantiate both the cellular control and data plane functions, which have inter-dependencies. As we described in §2.1, MME is the cellular control function, and S/P-GW are data plane functions in EPC. The MME interacts with S/P-GW during different events, e.g., a UE's attachment to the network and eNodeB handover. 3GPP provides guidelines for delay budgets for MME and S/P-GW [14]. The S/P-GW delay budget depends primarily on the requirements of the data traffic, while the delay budget for MME depends on the device characteristics (e.g., mobile smartphone device vs stationary M2M device). MME delay budgets are more stringent

because the MME deals with all the signaling traffic from the UE. If we assume that MME and S/P-GW can be placed in different data centers, we have three different types of delay budgets to consider, $Budget_{t,a}^{data}$, $Budget_t^{UE-MME}$ and $Budget_t^{MME-SGW}$, for device type $t$ and application $a$, Modeling these three constraints, yields a quadratic constraint, as we show later in §5.

**(2) Network orchestration:** The second key challenge is to implement the output of the resource manager, i.e., map a UE's data and cellular control traffic to VMs. This has broadly three challenges: 1) Backwards compatible wide-area orchestration to get the UE to the selected data center, and 2) Intra-DC orchestration: to get the traffic through selected VMs, 3) Handling load reconfiguration, because unlike today's static core network, KLEIN derives its benefits from dynamically reconfiguring the network load, which may require moving a UE's traffic to a different data center to re-balance the load on the network. This raises the question whether KLEIN can use existing orchestration and 3GPP mechanisms.

# 5 Resource Manager

The KLEIN resource management module distributes the network load across the datacenters, while ensuring that the latency budgets for different applications are satisfied. The key challenge here is achieving responsiveness at scale. In order to realize the benefits of the vision we outlined in §3, we need this module to rebalance the load and assign UEs to compute resources on fine-grained timescales (tens of seconds). However, this is challenging because the underlying distribution problem entails solving a large-scale and non-linear optimization. To address this, we use a combination of three key ideas: (1) solving the problem at an aggregate rather than UE granularity; (2) decoupling the problem of placing control and data-plane functions; and (3) decomposing the global optimization into a three-level hierarchy. As we will show this enables near optimal performance at scale.

We begin by setting up the key requirements of the optimization problem and then present our key ideas.

## 5.1 Problem Formulation

**Provider Setup:** We assume that the cellular core has been provisioned with a set of data centers $\{D_d\}_d$ and high-capacity backbone switches and links. Traffic from the base stations will be forwarded to one or more data centers. Each data center $D_d$ has pre-provisioned capacity with fixed number of servers and each server $s$ has fixed number of VM slots. We assume that the network is partitioned into *regions* – a collection of data centers in geographical proximity. This is similar to the way the core network is partitioned today [30].

The cellular operator has historical traffic patterns and has rough estimates of traffic volumes associated with end-user applications and cellular control traffic. Let $Data_{u,a,e}$ represent the data traffic associated with a UE $u$, an application $a$, in epoch $e$ and let $Ctrl_{u,e}$ represent the control signaling traffic associated with the UE $u$in epoch $e$. This information may be collected using network monitoring data (e.g., NetFlow).

**Processing requirements:** Different applications and device types may require different processing requirements. For instance an M2M device may be required to go through a specific chain of service or NFs. Similarly video traffic may go through additional transcoder middleboxes. For each traffic class $c$, consisting of a combination of device-type $t$ and application $a$, there is an associated policy service chain or a sequence of NFs. Each device type $t$ is also associated with a set of cellular control functions it needs.

**Objective:** The goal is to decide the assignment of data-plane and cellular control-plane traffic to a data center, and provisioning of required EPC functions and middleboxes. There are a few considerations in this assignment. First, we want traffic load across servers in the core network to be balanced. Specifically, the utilization of data centers to be balanced. Second, we need to ensure that each UE $u$ meets it processing (e.g., service chains) and latency bounds on data and control plane functions.

**Formulation:** We introduce three key decision variables: (1) $n_i^{d,s}$ denoting how many VMs of type $v_i$ (can be MME, S/P-GW and other network functions) of NF $i$ to run on server $s$ of data center $d$; (2) $DP_{u,s,d}$ which denotes whether data-traffic for UE $u$ is processed in server $s$ in data center $d$ and (3) $CP_{u,s,d}$ which denotes whether cellular control traffic for UE $u$ is processed at server $s$ in datacenter $D$.

Unfortunately, solving this problem is challenging on two key fronts. First, this is a large discrete optimization problem where the problem size (million of UEs and potentially thousands of data centers) makes it computationally intractable. Second, attempting to model constraints on the latency budgets between data-plane and control-plane functions inevitably yields quadratic constraints as shown below which make the problem even harder to solve with off-the-shelf solvers.[4] Specifically, Eq 8 highlights how modeling the latency between the SGW and MME introduces a non-linear interaction between the $CP$ and $DP$ variables. $L_{d,d'}$ refers to the latency between data center $d$ and $d'$.

$$\forall u \in t : \sum CP_{u,s,d} \times DP_{u,s',d'} \times L_{d,d'} \leq Budget_t^{MME-SGW} \tag{8}$$

## 5.2 Key Ideas

As we saw above, solving this problem is challenging because of the *scale* and the *non-linear* dependencies across the decision variables. To address these issues, we introduce three key heuristics:

- *Aggregation:* The first insight is that we do not need to solve the problem at a UE granularity. We may be able to achieve near-optimal results by aggregating UEs into groups of UEs, and make decisions at coarser aggregate granularities.
- *Hierarchical decomposition:* Finally, we observe that in practice we do not always need to solve the global optimization problem every few seconds. We can decompose the optimization along the natural hierarchy of global, regional, and intra-datacenter local controllers. For instance, the global controller need not assign the precise server inside the datacenter or the specific datacenter and can instead delegate these to the regional and local controllers respectively. Thus, higher levels which need a more global view solve simpler problems at coarser timescales, while the lower levels which need to be more responsive to avoid performance issues can run more rapid reconfigurations.
- *Decoupling control and data placement:* The key reason for the quadratic constraint in Eq 8 is that we were trying to solve the joint optimization of placing control and data functions. We can break this nonlinearity in one of two ways: (1) We can choose the control function placement and then solve the data-plane problem, with additional constraint of $Budget_t^{MME-SGW}$ or (2) constrain the control and data plane functions to be in the same server (i.e., collapsing $CP$ and $DP$). As we will see below, we find that approach (2) works well in the global controller and approach (1) in the regional controllers.

[4]Note that in section §3, we only modeled the data-plane latencies, hence there were no quadratic constraints.

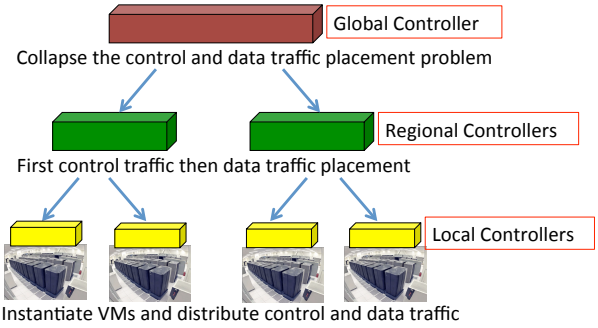


**Figure 7: An overview of KLEIN's resource manager.**

## 5.3 Our Approach

Next, we describe how we synthesize the above three heuristics into a practical and scalable resource management solution.

**Three level hierarchy:** We decompose the optimization problem into three logical subproblems following the natural structure of large cellular providers. Figure 7 shows this decomposition.

1. The global controller runs a *Region Selection Problem (RSP)* that assigns (aggregate) UE groups to specific regions;
2. The regional controller then runs the *Data center Selection Problem (DSP)* and further subdivides set of the (aggregate) UE groups assigned to it across the datacenters in its region;
3. The local or intra-datacenter controller runs a *Server Selection Problem (SSP)* which assigns specific servers inside each selected data center (as given by DSP) to run the required VMs.

This decomposition enables us to scale as the individual RSP, DSP, SSP problems can be solved respectively by the global, regional and local controllers. We also evaluated other degrees of decomposition and found that for the workload characteristics, this 3-level decomposition was a sweet spot between scalability and complexity; e.g., we tried a 2-level decomposition strategy and earlier showed in §4 that it does not scale (Table 1). We describe the specific optimization subproblem each tackles and the approach we use next.

**Region Selection Problem: (RSP):** In the first step, KLEIN global controller distributes traffic across region such that the load is balanced at a region-level granularity. The global controller takes as input the $Ctrl_{g,e}$ and $Data_{g,a,e}$ values, as well the aggregate capacities of individual regions and assigns each aggregate UE group, $g$ to a specific region based. To break the non-linear/quadratic dependency between the control and data-plane functions, the RSP simply assigns both to be in the same region, collapsing the CP and DP. We can formulate the RSP as an ILP, and rerun this ILP periodically (every 60 mins). We discuss the choice of this reconfiguration period in §8.1. In case any region is overloaded, they can make an `upcall` to KLEIN's global controller to reconfigure the load. The RSP formulation in essence is similar to the ILP formulation described in §3 for INTERMEDIATE, except that reconfiguration decisions are made for UE groups and load is distributed across regions.

**Data Center Selection Problem (DSP):** Each regional controller then has the responsibility for selecting the data center for every UE group $g$ that has been assigned to it by the RSP. Specifically, it has to choose a data-center for the control functions and another (possibly different) datacenter for the data-functions. At this stage, the regional controllers seek to distribute load at an aggregate datacenter granularity within the region. Now, the DSP runs a two-step procedure to break the quadratic dependency. First, it places the control functions and then runs a separate ILP for data plane functions. The ILP is similar to the DSP problem, except that the load is distributed across data centers, instead of regions. This procedure runs roughly every 5 minutes. We discuss the choice of this reconfiguration period in §8.1. In case a data center is overloaded, the local controller can make a `upcall` to KLEIN's regional controller.

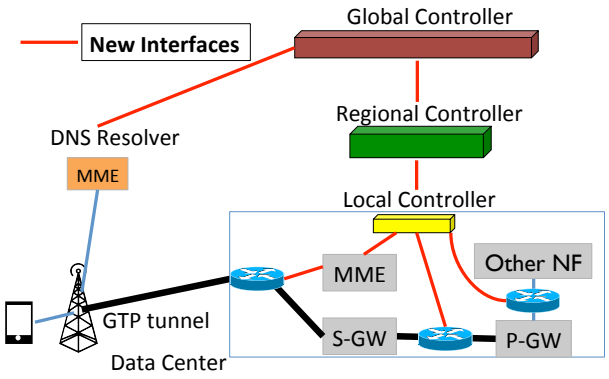


**Figure 8: An overview of network orchestration in a KLEIN based cellular core.**

**Server Selection Problem (SSP):** Finally, each local controller within each datacenter has to distribute load across servers and instantiate VMs. We use a simple greedy heuristic here, choosing nodes with higher capacities and ensuring that NFs in the same chain are assigned to the same server or the same rack similar to prior work [31].

# 6 Network Orchestration

Given the output of KLEIN's resource manager, UE's data and cellular control traffic need to be assigned to VMs. We discuss how this is done in two phases: (1) wide-area orchestration to get the UE to the correct DC, (2) intra-DC orchestration to get the traffic through correct VMs. Figure 8 provides an overview of these orchestration mechanisms. Below we describe these mechanisms, and how KLEIN handles the reassignment of UEs.

## 6.1 Wide-area Orchestration

In KLEIN, the cellular carrier's wide area network– the backbone network connecting the base stations and the data centers remains unmodified. We assume legacy routing, with carriers using existing tunneling mechanisms such as GTP to connect base stations to data centers. Today, when a UE attaches to an eNodeB, the eNodeB performs a DNS lookup to identify a MME to forward the attach request to. In response to the attach request, the MME acts as a DNS resolver to help select the S/P-GWs and set up the GTP tunnels.

In KLEIN, we enhance the attachment process slightly. In the initial MME selection DNS query, the eNodeB now includes the device and subscriber identifier of the UE in addition to the location information it sends today. This DNS query is serviced by the nearest MME using its DNS resolver capabilities. It uses the device and subscriber identifier to map the UE into a KLEIN UE group, and subsequently uses the mapping provided by the KLEIN controller to select an MME for servicing the UE's attach request and as its future *home* for control plane traffic. An attach request is

now sent by the eNodeB to the chosen home MME, which similarly identifies the UE's group from the device and subscriber information in request and selects a S/P-GW based on the UE group to GW mapping provided by the KLEIN controllers.

This enhanced registration procedure requires the MME to maintain a connection with the KLEIN's controllers to obtain an up-to-date UE group to MME and S/P-GW mapping. Fortunately, since MMEs use a standard DNS interface for server selection, such a connection requires minimal integration on the KLEIN controller's part.

## 6.2 Intra-datacenter Orchestration

While the wide-area orchestration is responsible for choosing an MME, S-GW, and P-GW to route the UE's traffic. intra-DC orchestration is needed to implement the NF (middlebox) service chains that traffic passes through after it leaves the P-GW. There are two main considerations here:

1. After it leaves the P-GW, the next VM a packet needs to be sent to depends on the service chain associated with the traffic. We need to implement service chains corresponding to different traffic classes (UE devices and applications). These NFs may include elements such as NAT, firewalls, intrusion detection systems, TCP-termination proxies for improving latency and throughput, content-caches, or media transcoders.

2. With elastic scaling, each service chain NF may be implemented as a collection of load-balanced VMs and the number of such VMs may grow or shrink based on demand. This requires a load balancing mechanism at the level of each server.

We borrow from prior work [31, 40, 38, 8] to solve (1) and (2). We use SDN inside data centers to enforce service-chain policies by dynamically routing traffic to the desired sequence of VMs. We apply service chain policies based on APN values. We use a tag-based approach similar to [31, 40] to ensure that service chains can be correctly implemented. Each VM has a tag value, and forwarding is done based on these tags. A service chain we assume is based on the UE device-type and application.

To balance load across multiple instances of a network function, we cannot use a dedicated load balancer, because it would itself become a bottleneck, since its on the path of every VM. We use a distributed load balancer at the level of each server, similar to [31, 38] to balance load among multiple instances of the same network functions in a server while handling scale-in and scale-out of the NF VMs.

## 6.3 KLEIN's Reconfigurations

As KLEIN's resource manager reconfigures traffic load, a UE's processing may need to be migrated to new EPC instances. 3GPP protocol's mandate that MME selects the S/P-GW for a device. To ensure backwards-compatibility KLEIN maintains an interface with MME instances and dynamically updates in these MME instances mappings from UE to S/P-GW instances. We use standard 3GPP handover mechanisms to migrate a UE to a different MME/SGW instance [1, 11] and existing (but non-standardized) mechanisms for P-GW handover [4, 16]. To ensure that migrating the P-GW and NAT associated with a UE to a new site does not change the IP address associated with the UE's sessions we assume, as is common practice in carriers today, that the data center sites are connected over a layer 2 MPLS-based VPN. As has been demonstrated in previous work [47, 33], IP session migration using L2VPNs can accommodate even demanding applications like gaming.

Below we consider all the three possible types of reconfigurations by KLEIN and how they will be handled by the network orchestration layer:

- **Intra-DC:** In KLEIN, a local controller may reconfigure the load to a different VM, inside the same data center. If a UE's traffic is placed to a different MME instance, the local controller triggers a MME handover from the old MME instance using standard 3GPP handover procedures [1]. If the UE's data traffic needs to be moved to another S/P-GW instance within the same data center, the local controller initiates a S/P-GW handover that moves the UE session context from the old S/P-GW instance to the new S-GW instance. These handovers are standard operating procedure for supporting mobility in existing 3GPP networks, and are well supported by existing implementations. For an intra-DC handover, no changes to the service chain are needed, because the distributed load balancer ensures session affinity such that the new P-GW will continue to send a UE's packets to the same next hop VM in the chain. Typically, session affinity is implemented using a common flow-table across the SDN switches implementing the load balancer [22].
- **Intra-Region:** In case the UE's data or signaling traffic needs to be moved to another data center, the same standard mechanisms for MME, S/P-GW handover as in the intra-DC case are used. However, there are several important differences. First, in case of intra-DC handovers, the new mapping is provided by the regional controller to the local controllers. Second, when a P-GW or external facing NAT moves from one DC to another, we need to ensure that the UE's IP address does not change. This is achieved through the L2 MPLS-VPNs as described above. And finally, any middlebox state also needs to be migrated from the old DC to the new DC. This can be achieved by ensuring that the middleboxes support a state-migration protocol such as OpenNF [32].
- **Inter-Region:** In case a UE's traffic is moved to a data center in a different region, the global controller passes UE remappings to the regional controller (old-new data center), which then in turn passes it to the local controllers. The local controllers then trigger MME handover or S-GW handover. In case of P-GW, it moves the instance to the new data center. Similar considerations of middlebox state migration as in the intra-region case are involved in inter-region transfers.

# 7 Implementation

In this section, we describe an implementation of KLEIN that we will use in the following section for performance evaluation. The implementation consists of EPC, resource management and orchestration layers and uses emulated UEs and eNodeBs.

**EPC implementation:** We use `OpenAirInterface` (OAI) version 0.1, an open source software implementation of EPC functions (MME, S-GW, P-GW, HSS), and eNodeB and UEs [18]. The UE and eNodeB behaviors are emulated; they are virtualized and run inside VMs. The EPC functions are also run inside a VM. In OAI, the EPC functions, viz., MME, S-GW, and P-GW, are tightly integrated and run inside the same VM. A key limitation of OAI is that the binding of UE to EPC is static. We extended OAI to enable dynamic remapping of the UE to a different EPC instance. We made some simplifying assumptions to do this, such as copying all UE contexts to all MMEs at the beginning. It is possible to do this for a small testbed and does not impact the broad performance measures we are interested in here.

**Resource management and orchestration layers:** Each EPC instance is realized as a VM. Emulated UEs and eNodeBs are also run inside VMs. In our testbed, we have a single UE per eNodeB

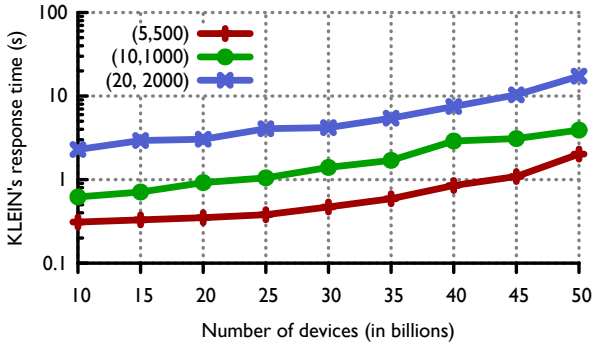


**Figure 9:** KLEIN**'s responsiveness**

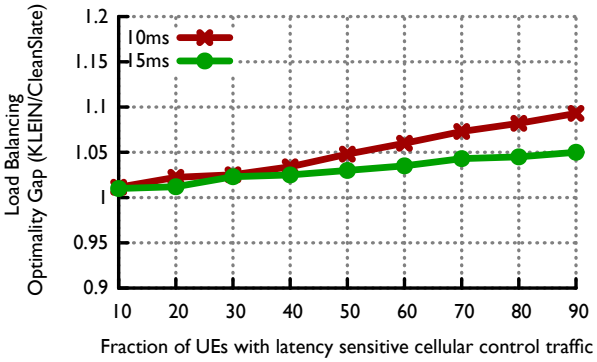


**Figure 10:** KLEIN**'s optimality**

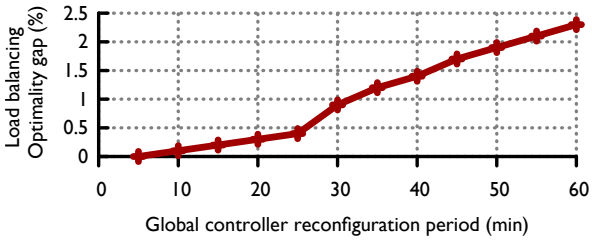


**Figure 11: Varying reconfiguration period**

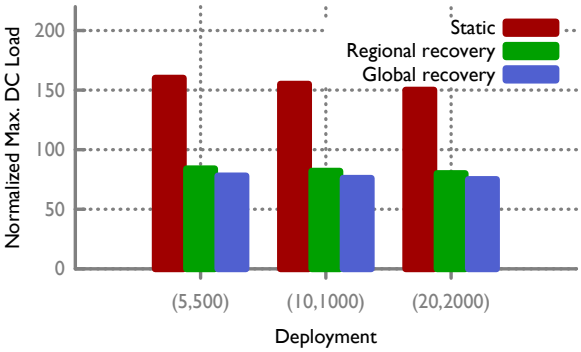


**Figure 12:** KLEIN**'s failure handling**

because of OAI constraints. We use OpenvSwitch [20] to emulate switches inside the DCs. We developed custom implementations of the global and regional controllers using `CPLEX` to run these algorithms. We use the `Floodlight` [10] SDN controller which installs rules inside SDN switches to dynamically route traffic among the VMs.

# 8 Evaluation

In this section, we use a combination of real testbed and trace-driven simulations to demonstrate the following benefits of KLEIN:

1. KLEIN is scalable and within 10% of the optimal. Our system takes less than 20 s to reconfigure a network with 2000 data centers and 5 billion devices and is within 10% of an ideal CLEANSLATE for a range of workloads. (§ 8.1)
2. KLEIN can improve end-application performance by upto a factor of 5. (§ 8.2)
3. The KLEIN end-to-end prototype implementation delivers the promised benefits and accurately mirrors the intended load distribution. (§ 8.3)
4. KLEIN offers new dimensions of elasticity and fault tolerance. It can handle data center failures both rapidly and efficiently, taking less than 2.3 s, and reducing the maximum load by a factor of 2. (§ 8.4)

**Setup and methodology:** Before going into the results, we describe the testbed and simulation setups used for the experiments.

- **Testbed:** The testbed runs on Emulab [9]. The testbed uses upto 24 machine with 2.4 GHz 64-bit Quad Core and 12 GB of RAM. On each machine, we assigned equal amount of resources to each VM: 1 vCPU (virtual CPU) and 3 GB of memory. Each VM runs Ubuntu 12.04 (Linux kernel version 3.13.0-32-generic).
- **Simulation setup:** The large scale simulations using data set in §2 are run on a machine with 80 CPU cores and 500 GB of RAM, with each core being a Xeon E74850 running at 2 GHz.
- **Deployments:** We consider different core network sizes, from 500 data centers to 2000 data centers. The locations of such data centers are assumed close to the eNodeBs they meant to serve. This is done by a nearest-neighbor clustering of eNodeBs in the 2D space and locating the data centers at the centroid of such clusters.
- **Traffic demands:** We use the data set in §2 to get the traffic demands. In addition, we vary the mix of traffic (latency sensitive and latency tolerant) and the delay budgets from different traffic classes to consider a variety of traffic scenarios.

## 8.1 Scalability and Optimality

**Scalability and responsiveness:** Figure 9 shows the run time of KLEIN for different cellular core network sizes and number of devices. A configuration (A,B) corresponds to a deployment of *B* data centers, where we assumed *A* regions. The y-axis shows the total response time when both the global and regional controllers have to reconfigure load. We observe that run time is less than 20 s for a network with 2000 sites and 50 billion devices.[5] In contrast with a 2-level decomposition (as shown in Table 1) even with aggregation, it takes more than a day to reconfigure the load.

[5]Note for 50 billion devices, we use a total 50,000 aggregates of UEs.

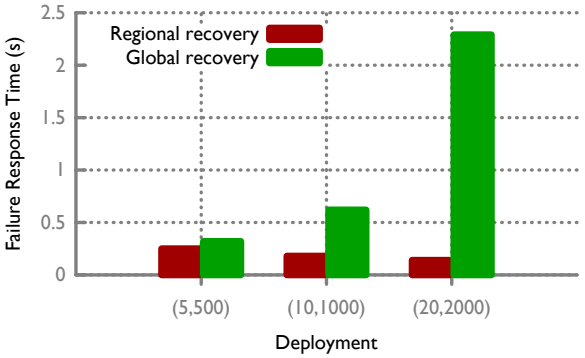


**Figure 13: KLEIN's failure response**

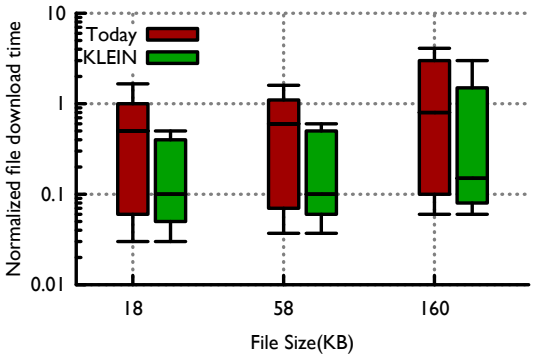


**Figure 14: Impact on end-application performance**

**Optimality:** Figure 10 shows for different traffic mix and delay budgets, how KLEIN's heuristics perform. The y-axis shows the load balancing optimality gap, (KLEIN/CleanSlate). The CleanSlate solution optimally load balances the data and control plane functions without considering any $Budget_t^{MME-SGW}$ constraints. We observe KLEIN's remains within 10% of the optimal solution, even for very stringent delay budget of 10 ms[6].

**Varying reconfiguration period:** Figure 11 shows the impact of reconfiguration period on the effectiveness of KLEIN's load balancing. The y-axis shows the load balancing gap, where we compare against a base line where every 5 mins both the regional and global controller reconfigure the load. We vary the global controller reconfiguration period, from 5 mins to 60 mins, while the regional controllers constantly reconfigure the load every 5 mins. Even with global reconfiguration at 60 mins intervals, the gap is only 2.5%. We find that one effective strategy is where global controller reconfigures the load periodically every 60 mins, while the regional controller performs reconfigurations every 5 mins.

## 8.2 End-Application Performance

Figure 14 shows the potential impact of KLEIN's load distribution on end-application performance as compared to *Today's* load distribution. We consider 3 different file sizes, and using our micro-benchmarking experiments similar to Figure 2(c), build a database of mapping between EPC utilization and file download time. Then we consider a period of 2-hours from our data-set, and consider the load distribution today and in KLEIN, specifically noting the data centers each UE data traffic was mapped to. We observe that

[6]We assumed 1000 data centers for these experiments.

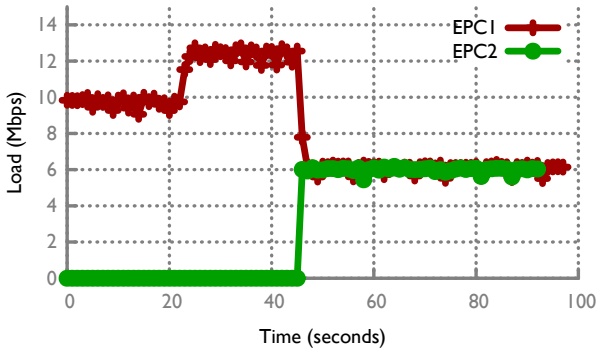


**Figure 15: Handling traffic overload on an EPC instance by instantiating a new EPC instance.**

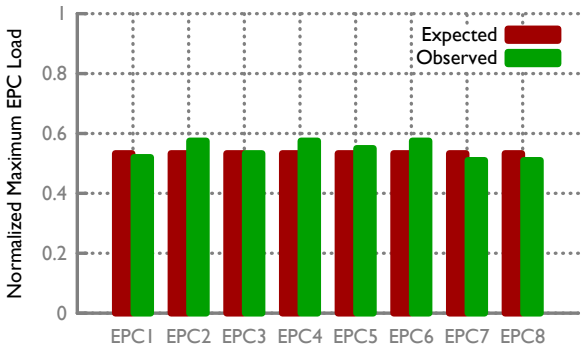


**Figure 16: Validation of KLEIN on EPC testbed**

KLEIN can reduce median file download time by upto a factor of 5, because it ensure that data center utilization is balanced.

## 8.3 End-to-End System Validation

We have validated our testbed demonstrating expected load balanced operation. Here, we consider 16 UEs attached to 16 base stations, and a total of 8 EPC instances. We use traffic data from 16 different base stations from the real data set, and scale their load to adjust to the capabilities of the testbed. We then dynamically map a UE's traffic to a specific EPC instance, based on the load on the network and using SDN Floodlight controller, to dynamically install forwarding rules the switches. We consider three different traffic distributions which represent loads hour apart. Figure 16 shows the gap between the observed load and the output from KLEIN. Figure 16 shows the gap between the observed load and the output from KLEIN optimization (Expected). For all EPC instances, the observed load is within 5% of the expected load.

## 8.4 New Opportunities

**Elastic scaling:** One of the new opportunities that KLEIN offers is the ability to dynamically scale EPC functions. This is because in KLEIN, the EPC functions are virtualized and can be instantiated based on demand. To illustrate this, we consider an elastic scaling experiment in our testbed, where we assume three UEs attached to an EPC instance which experiences a spike in load. The EPC instance is overloaded and we instantiate another EPC instance, and move some traffic to this new instance. The Figure 15 shows the time series. The load is evenly distributed and KLEIN avoids a potential overload scenario. We observe it takes in the order of a

few ms to exchange UE state between different EPC instances and be able to route data traffic to the new EPC instance.

**Failure handling:** We consider a scenario where we fail individual data centers, and use KLEIN's dynamic remapping to reconfigure the load on the network. We compare it against a *Static* failure management strategy, where in the case of a failure, the load is statically mapped to the nearest data center. Figure 12 shows KLEIN avoids potential overload scenarios in the face of failure. We consider 3 different core network deployments, consisting of 500, 1000 and 2000 data centers. KLEIN can handle data center failures at two levels. It can perform (1) *Regional recovery*, where the regional controller tries to redistribute the load within the same region and (2) *Global recovery*, where the global controller redistributes the load across the cellular core. KLEIN can reduce maximum load on any data center in the cellular core by upto 100% as compared to a *Static* strategy. As shown in Figure 13, a *Global recovery* takes KLEIN upto 2.3s and a *Regional recovery* upto 0.3s.

# 9 Related Work

**Cellular SDN and NFV:** Recent proposals on redesigning the cellular core using SDN and NFV argue for an approach akin to a clean slate design [34, 39], with new cellular signaling protocols and SDN-like wide-area routing. Other proposals [29, 42, 36, 35] present backwards compatible mechanisms for virtualizing core EPC functions like S-GW and P-GW. However, all these ignore a resource management layer for managing resources. The core contribution of this work is a scalable resource manager, realized via backwards compatible orchestration mechanisms.

**Middleboxes and NFV:** Related work has focused on "middlebox" service chaining and load balancing [40, 43, 49]. Other work has also suggested NFV-like ideas for traditional middleboxes [32, 45]. In this work we consider the problem of load distribution and network function placement over the entire core network, consisting of thousands of sites and billion of mobile devices.

**Control plane design:** There have been prior proposals for a hierarchical control plane design in different context [39, 37], our goal here is to investigate how we design a scalable and responsive resource management layer for handling thousands of sites and billion of mobile devices.

**Middlebox/EPC state management:** One of the key benefits of KLEIN comes from dynamic load balancing across sites, which in turn requires moving traffic from one data center to another. Many middleboxes maintain active per-flow state, moving traffic will require mechanisms to move the relevant state from these middleboxes to the new site. Recent work [32, 41] can be used to address these challenges.

# 10 Conclusions

Today the cellular core suffers from a range of limitations and inefficiencies. As operators are rapidly improving the access bandwidth in (4G/LTE) networks, the core network remains the bottleneck. As carriers are looking to redesign their networks, an important question is whether we need to completely redesign the cellular core. Using a data-driven analysis we find that we can near-optimally achieve the benefits of an optimal clean slate approach using KLEIN, a minimally disruptive redesign. The key observation is that by combining virtualized network functions with a smart resource management layer and a more distributed cellular core, we can achieve almost all of the promised elasticity benefits while working within the operational constraints of existing 3GPP standards.

# Acknowledgments

We would like to thank the anonymous reviewers for their feedback and suggestions on improving the paper. This research was supported in part by NSF grant CNS-1117719 and Intel Labs' University Research Office.

# 11 References

[1] 3GPP Evolved Packet System (EPS); Evolved General Packet Radio Service (GPRS) Tunneling Protocol for Control Plane (GTPv2-C). http://www.3gpp.org/DynaReport/29274.htm.

[2] ADARA. http://www.adaranet.com/.

[3] Affirmed Networks. http://www.affirmednetworks.com/.

[4] Architectural EPC Extensions for Supporting Heterogeneous Mobility Schemes. Document by MEVICO, January 2013. http://www.mevico.org/D22.pdf.

[5] ARICENT. https://www.aricent.com/.

[6] AT&T Domain 2.0 Vision White Paper. http://tinyurl.com/p4uv3s3.

[7] AT&T launches virtualized packet core in Europe. http://www.fiercewireless.com/tech/story/att-launches-virtualized-packet-core-europe/2015-10-06.

[8] Contrail Architecture. http://www.juniper.net/us/en/local/pdf/whitepapers/2000535-en.pdf.

[9] Emulab. https://www.emulab.net/.

[10] Floodlight. http://www.projectfloodlight.org/floodlight/.

[11] General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access. http://www.3gpp.org/DynaReport/23401.htm/.

[12] Introducing ONOS - a SDN network operating system for Service Providers. http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf.

[13] LTE Connectem Inc. http://www.connectem.net/.

[14] LTE Design and Deployment Strategies. http://tinyurl.com/lj2erpg.

[15] Managing the Signaling Storm. http://goo.gl/lkTyb1.

[16] Mobile Gateway Configuration Guide. Alcatel Lucent Technical Document, 2014. http://infoproducts.alcatel-lucent.com/cgi-bin/dbaccessfilename.cgi/9305240102_V1_7750%20SR-OS%20Mobile%20Gateway%20Configuration%20Guide%206.0r5.pdf.

[17] Morgan Stanley Releases The Mobile Internet Report. http://www.morganstanley.com/.

[18] OpenAirInterface. http://openairinterface.org.

[19] The OpenEPC Project. http://http://www.openepc.com/.

[20] OpenvSwitch. http://openvswitch.org/.

[21] Percentage of all global web pages served to mobile phones from 2009 to 2015. http://www.statista.com/statistics.

[22] Service Chain Load Balancing with OpenContrail. http://www.opencontrail.org/service-chain-load-balancing-with-opencontrail/.

[23] State of the News Media 2015. http://www.journalism.org/2015/04/29/state-of-the-news-media-2015/.

[24] vEPC in LTE networks: Time to move ahead. Blog, March 2015. https://techzine.alcatel-lucent.com/vepc-lte-networks-time-move-ahead?s_cid=smm15_tmc0481_bl.

[25] The Zettabyte Era: Trends and Analysis. Cisco Technology White Paper, May 2015. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-

index-vni/VNI_Hyperconnectivity_WP.pdf.

[26] O. Abdelrahman and E. Gelenbe. Signalling storms in 3G Mobile Networks. In *Proc. ICC*, 2014.

[27] M. Balakrishnan et al. Where's that phone?: Geolocating IP Addresses on 3G Networks. Proc. IMC'09.

[28] A. Banerjee et al. Phantomnet: Research infrastructure for mobile networking, cloud computing and software-defined networking. *GetMobile: Mobile Computing and Communications*, 19(2):28–33, 2015.

[29] A. Basta et al. A virtual SDN-enabled LTE EPC architecture: A case study for S-/P-Gateways functions. In *Proc. SDN4FNS*, 2013.

[30] J. Cho, B. Nguyen, A. Banerjee, R. Ricci, J. Van der Merwe, and K. Webb. SMORE: Software-defined Networking Mobile Offloading Architecture. In *Proc. AllThingsCellular'14*, pages 21–26, 2014.

[31] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey. Bohatei: Flexible and elastic ddos defense. In *Proc. SEC'15*, pages 817–832. USENIX Association, 2015.

[32] A. Gember et al. OpenNF: Enabling Innovation in Network Function Control. In *Proc. SIGCOMM*, pages 163–174, 2014.

[33] V. Jalaparti, M. Caesar, S. Lee, J. Pang, and J. Van der Merwe. SMOG: A cloud platform for seamless wide area migration of online games. In *Proc. NetGames'12*, pages 9:1–9:6, 2012.

[34] X. Jin et al. SoftCell: Scalable and Flexible Core Network Architecture. In *Proc. CoNEXT*, 2013.

[35] J. Kempf et al. Moving the Mobile Evolved Packet Core to the Cloud . In *International Workshop on Selected Topics in Mobile and Wireless Computing*, 2012.

[36] K.Pentikousis et al. Mobileflow: Toward software-defined mobile networks . In *International Workshop on Selected Topics in Mobile and Wireless Computing*, 2013.

[37] J. MacCauley et al. Extending SDN to Large-Scale Networks. *Open Network Summit*, 2013.

[38] H. Matsuba et al. Airfoil: A topology aware distributed load balancing service. In *IEEE Cloud*, 2015.

[39] M. Moradi et al. Softmow: Recursive and Reconfigurable Cellular WAN Architecture. Proc. CoNEXT '14.

[40] Z. Qazi et al. SIMPLE-fying Middlebox Policy Enforcement Using SDN. In *Proc. SIGCOMM*, pages 27–38, 2013.

[41] S. Rajagopalan et al. Split/Merge: System Support for Elastic Execution in Virtual Middleboxes. In *Proc. NSDI*, pages 227–240, 2013.

[42] M. Sama et al. Software-Defined Control of the Virtualized Mobile Packet Core. In *IEEE Communications Magazine*, 2015.

[43] V. Sekar et al. Design and Implementation of a Consolidated Middlebox Architecture. In *Proc. NSDI*, 2012.

[44] S.Elby. Carrier Vision of SDN and Future Applications to Achieve a more Agile Mobile Businesss. Keynote at the OpenFlow World Congress, 2012.

[45] J. Sherry et al. Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service. In *Proc. SIGCOMM*, 2012.

[46] Z. Wang et al. An Untold Story of Middleboxes in Cellular Networks. In *Proc. SIGCOMM*, pages 374–385, 2011.

[47] T. Wood, K. Ramakrishnan, P. Shenoy, J. Van der Merwe, J. Hwang, G. Liu, and L. Chaufournier. CloudNet: Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines. *IEEE/ACM Transactions on Networking*, 23(5):1568–1583, Oct 2015.

[48] Q. Xu et al. Cellular Data Network Infrastructure Characterization and Implication on Mobile Content Placement. In *Proc. SIGMETRICS*, pages 317–328. ACM, 2011.

[49] Y. Zhang et al. StEERING: A Software-Defined Network for Inline Service Chaining. In *Proc. ICNP*, 2013.