Optimization of Cloud Connectivity using a Smart–home Gateway

Ondrej Tomanek, Lukas Kencl
Faculty of Electrical Engineering
Czech Technical University in Prague
Email: {ondrej.tomanek, lukas.kencl} [at] fel.cvut.cz

***Abstract*—When consuming Cloud services from their homes, contemporary users face a significant degree of uncertainty with regards to Cloud–service performance and reliability. This uncertainty could be addressed by an intelligent approach deployed within a smart home gateway, which would continuously reevaluate and reassign Cloud–bound traffic to a momentarily optimal destination. We demonstrate that such optimization could bring significant benefits to individual users in terms of quality and reliability of the provided Cloud service. Furthermore, we sketch a schema of a methodology that regularly reassigns Cloud–bound traffic, based on past measurements and a constrained optimization computation; we present an experimental performance evaluation of such approach and discuss feasibility of its implementation.**

***Index Terms*—Smart home; Gateway; Cloud; Performance; Reliability; Optimization.**

I. Introduction

Nowadays, people leverage, utilize and in some cases depend upon smart devices to solve problems [1]. Many such use cases are enabled by or benefit from a service backend, increasingly hosted remotely in public Cloud datacenters (DCs). This arrangement, however, poses a non–trivial task of ensuring end–to–end Cloud connection properties, necessary for devices to operate as expected. The reality is that even the most popular Cloud services often degrade and occasionally fail for various reasons [2], [3].

We focus on the home environment [4], where smart technology has been increasingly deployed, leading to a growing network traffic and diverse applications (e.g., control, telemetry, entertainment, update). However, neither the diverse needs of applications nor the immediate Cloud–service performance status (e.g., latency, throughput, cost) are really considered.

In this paper, we present a datacenter–connectivity optimization scheme, formalized as a binary mixed–integer linear program. The scheme, to be deployed on smart–home gateways, leverages available network information and models Cloud status through statistical measures of latency, throughput and traffic costs. These then guide dynamic Cloud–bound traffic assignments. In turn, adverse network effects on packets traversing a home boundary get minimized by leveraging up–to–date information about the variously–performing candidate datacenter connections. The scheme thus:

- mitigates impact of Internet–path and Cloud outages;
- diverts traffic from degraded datacenters;
- reduces adverse network effects on existing connections;
- precalculates assignment of future Cloud connections.

Our work adds to the state of the art a solution that operates at network edge, uses readily–available input data, is preemptive, avoids vendor lock–in, allows for triggered or periodic execution and operates at a sub–request granularity while considering both upstream and downstream path performance. We evaluate the scheme merits by a discrete–time simulation – using two common scenarios over 70 days, using a typical application mix, hosted in five representative smart–home locations on four continents and serviced by eight globally–deployed datacenters of two major public Cloud service providers – Amazon AWS [5] and Microsoft Azure [6] (from now on compared exclusively using randomized order and obfuscated names P1 and P2). Our small–scale evaluation confirms that such optimization reduces adverse network effects on smart home and avoids outages and poorly–performing datacenters on the Cloud end. Such improved services are beneficial for smart–home users, Cloud tenants and providers. To take full advantage of the scheme in larger deployments, practical implementation concerns have to be addressed and a good latency–prediction model has to be used. The main contributions of this paper are:

- A proposed traffic–assignment optimization scheme;
- An experimental evaluation over a large open dataset;
- A discussion of practical implementation.

The remainder of this paper is structured as follows: an overview of a related work is given in Section II. Models and an optimization task are described in Section III. Optimization merits are evaluated and validated in Section IV. Practical implementation considerations are discussed in Section V. We conclude and summarize implications in Section VI.

II. State of the art

Many popular smart appliances have strict connection–quality requirements to remain operational – *Nest*'s availability [7] or *Echo*'s latency [8] requirements to name a few. Some smart–service providers employ an uplink monitoring from appliance–end or service–end, but others do not have any such special arrangements and rely on a well–behaved network or an application–aware network, like in the case of home [9] and mobile edge networks [10]. Data from the appliances are often handled by data streaming and processing frameworks like *Kafka* [11], *Millwheel* [12] or *Kinesis* [13], which buffer and transform telemetry data inside LAN or at the edge before sending the stream to its consumers. These consumers reside

TABLE I
STATE OF THE ART

Paper	Composite metric
Sun12 [20]	server revenue, link revenue
Doyle13 [21]	carbon emissions, request time, electricity cost
Couto14 [22]	latency, survivability
Forestiero17 [23]	carbon emissions, energy utilization, electricity cost
Tripathi17 [24]	server cost, renewables cost, brown energy cost
Zhou17 [25]	request time, computation cost
Maswood17 [26]	bandwidth cost, link utilization, resource utilization
This paper	mean latency, latency deviation, timeout rate

either in Cloud datacenters or at the edge [14], like in the case of major–provider offerings *Amazon Greengrass* [15] or *Azure IoT Edge* [16]. These complex service architectures are tricky to support simultaneously, which led to the evolution of the conventional home routers into smart gateways [17]–[19] that are resource rich and have extra capabilities. Our optimization scheme supports these diverse architectures by considering application requirements and leveraging Edge and Fog computation–offload infrastructures whenever they outperform the Cloud datacenter. Also, the scheme prevents vendor and datacenter lock–in by allowing selection across multiple Cloud service providers and operating preemptively at a sub–request level.

There is a vast literature on the topic of optimal Cloud–request assignment in a presence of multiple candidate datacenters. The techniques operate at Cloud provider level and are usually implemented as mixed–integer linear programs. They score datacenters using composite metrics reflecting objectives geared towards greening, user experience or cost (summarized in Tab. I). Evaluation of the techniques is often carried out by hypothetical scenarios and varying weights of partial objectives rather than reporting and validating immediate–deployment improvement using an actual Cloud behavior ground truth. Also, effectiveness of techniques strongly depends on composite metric's weights setting, which is tricky to get right in a real environment and, unfortunately, only a little guidance is provided. Instead of the full integer programming

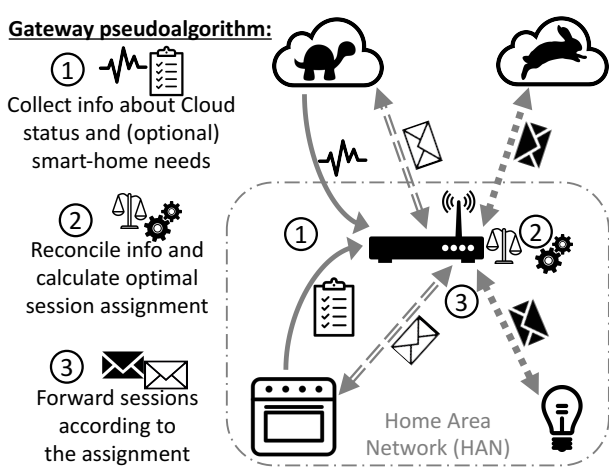

Fig. 1. *System outline.* Smart–home gateway observes status of candidate Cloud datacenters and, optionally, also the needs of smart–home devices (denoted by solid arrows). This information is then used for mitigating impact of Cloud outages and degradations, as well as for periodic or instant assignment–optimization of existing and future smart–home sessions, interacting with Cloud services (denoted by double–dashed and square–dot arrows).

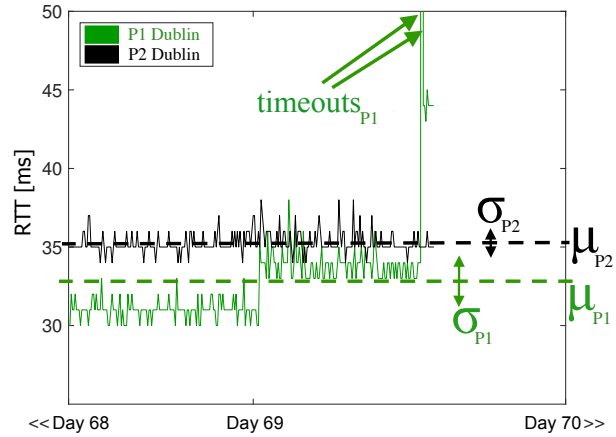

Fig. 2. *Example of adverse network effects.* Derived from j–th datacenter's RTT latency timeseries $\vec{x}_j$ are timeout rate τ_j, latency mean μ_j and standard deviation σ_j that represent the ever–present adverse network effects whose optimal combination across a set of candidate datacenters J is sought.

techniques, simpler change–point detection approaches also exist that adapt load distribution among datacenters to dynamic factors like electricity prices or availability of renewables [27].

A fair comparison among all these disparate state–of–the–art solutions is much needed in order to reveal the importance of objectives in various contexts and, consequently, maximize benefits at an acceptable complexity. Preventing comparison with our approach is a state–of–the–art's mixture of vendor lock–in and request–level nature, differing objectives, input–data availability and operation at a Cloud–provider level.

III. MODEL AND OPTIMIZATION

The system of interest, outlined in Fig. 1, consists of smart–home devices running applications, which interact with Cloud services over the network via a smart–home gateway. Smart–home applications can be classified according to the nature of relationship to adverse network conditions, i.e., applications can be latency–sensitive (realtime control), jitter–sensitive (gaming), loss–sensitive (over–the–air update) or throughput–intensive (streaming). To capture this diversity, we model the needs of a set of smart–home application sessions I that interact with Cloud services. Sessions I are described by data rates R and average packet sizes P (in both upload and download directions M), as well as sensitivity W to adverse network conditions. We also consider a monetary budget b to control Cloud traffic costs. Tab. II summarizes the notation.

We then describe a set of candidate Cloud datacenters J using their properties, as visible to smart home. Datacenters J are described by download and upload throughputs T; cost of download traffic c and $|K| = 3$ representative measures of adverse latency conditions: timeout rate τ, latency mean μ and standard deviation σ (see examples in Fig. 2). These three can be conveniently derived from j–th datacenter's round–trip–time (RTT) latency timeseries $\vec{x}_j$ of length N, obtained by probing conducted by smart–home gateway itself or by a 3rd–party service. μ and σ are calculated using timeseries normalized across all datacenters (Eq. 1). That is to unify range and to reduce bias, induced by varying distances between sites.

$$\hat{x}_{j,n} = \sqrt{1 - \frac{min\{x_{j,n}\}}{x_{j,n}}}, \;\; n = 1 \ldots N, \;\; \forall j \in J \quad (1)$$

τ is transformed using a square root operation as it tends to be highly clustered (Eq. 2). For convenience, τ, μ and σ are organized as columns of matrix A (Eq. 2, 3 and 4).

$$\tau_j = a_{j,\tau} = \sqrt{\frac{\#timeouts(\vec{x}_j)}{N}}, \;\; \forall j \in J \quad (2)$$

$$\mu_j = a_{j,\mu} = \frac{1}{N} \sum_{\forall n} \hat{x}_{j,n}, \;\; \forall j \in J \quad (3)$$

$$\sigma_j = a_{j,\sigma} = \sqrt{\frac{1}{N-1} \sum_{\forall n} |\hat{x}_{j,n} - \mu_j|^2}, \;\; \forall j \in J \quad (4)$$

Given the smart–home and Cloud models, we formulate the optimization task as a binary mixed–integer linear program (Eq. 5). The goal of the objective function f is to minimize cumulative adverse network effect (i.e., amount of latency, deviation and timeouts) on packets traversing a home network boundary. The optimized variable $s_{i,j}$ represents session i's traffic fraction, assigned to datacenter j. Constraints ensure that only datacenters capable of serving respective sessions are considered (Eq. 8), selected sessions do not undergo optimization (i.e., are not steered or split – Eq. 9 and 10), datacenter throughputs are sufficient (Eq. 11 and 12) and budget is sufficient (Eq. 13). These constraints can be relaxed based on information availability (see Fig. 3). Note that the objective function consists of only RTT–derived metrics and is unitless, since it contains one ratio and two normalized quantities. Datacenter–upstream and downstream throughputs and traffic costs are modeled as constraints – as such, they restrict the solution space and thus affect the optimal datacenter selection. We decided not to add these to the objective function due to their incomparable units ($, kbps).

$$\arg\min_{s \in \mathbb{R}} f(s) = \arg\min_{s \in \mathbb{R}} \sum_{\forall i} \sum_{\forall j} \sum_{\forall k} \sum_{\forall m} w_{i,k} a_{j,k} \frac{r_{i,m} s_{i,j}}{p_{i,m}} \quad (5)$$

subject to

$$s_{i,j} \geq 0, \;\; \forall i \in I, \;\; \forall j \in J \quad (6)$$

$$\sum_{\forall j} s_{i,j} = 1, \;\; \forall i \in I \quad (7)$$

$$s_{i,j} = 0, \;\; \forall (i,j) : j \text{ incapable of serving } i \quad (8)$$

$$s_{i,j} = s'_{i,j}, \;\; \forall i : i \text{ cannot be steered}, \;\; \forall j \in J \quad (9)$$

$$s_{i,j} \in \{0,1\}, \;\; \forall i : i \text{ cannot be split}, \;\; \forall j \in J \quad (10)$$

$$\sum_{\forall i} s_{i,j} r_{i,m} \leq t_{j,m}, \;\; \forall j \in J, \;\; m = \text{download} \quad (11)$$

$$\sum_{\forall i} s_{i,j} r_{i,m} \leq t_{j,m}, \;\; \forall j \in J, \;\; m = \text{upload} \quad (12)$$

$$\sum_{\forall i} \sum_{\forall j} s_{i,j} r_{i,m} c_j \leq b, \;\; m = \text{download} \quad (13)$$

TABLE II
SUMMARY OF NOTATION

Term	Definition
I	Set of smart–home sessions
J	Set of candidate datacenters
N	Length of RTT timeseries
K	Set of adverse network effects: τ, μ and σ
M	Set of traffic directions: upload and download
$x_{j,n}$	RTT measurement of j–th DC at n–th moment
$r_{i,m}$	Data rate of session i in direction m
$p_{i,m}$	Average packet size of session i in direction m
$w_{i,k}$	Sensitivity of i–th session to k–th adverse effect
b	Remaining financial budget for Cloud traffic
c_j	Cost of download traffic from j–th DC
$t_{j,m}$	Throughput of j–th DC in direction m
$a_{j,k}$	Size of k–th adverse network effect at j–th DC
$s_{i,j}$	Traffic fraction of i–th session, assigned to j–th DC
$s'_{i,j}$	Previous traffic fraction of i–th session to j–th DC
f	Objective function – cumulative adverse network effect

In the worst case, presence of Eq. 10–type constraints changes the problem complexity from $\mathcal{P}$ to $\mathcal{NP}$–hard. In that case, a practical program computation is only thinkable for up to our considered deployment scales, e.g., a network with a moderate number of Cloud–bound sessions I and several candidate datacenters J. Even in such case the program computation is feasible using conventional solvers. The program computation is preemptive in that it reassigns sessions capable of being reassigned throughout their duration. It is either periodic (according to a configured timer) or triggered by a significant change in Cloud status or smart–home needs. The entire optimization scheme is outlined in Fig. 3. An example of the optimization–scheme behavior around a period of datacenter degradation is depicted in Fig. 4.

IV. EXPERIMENTAL EVALUATION

We emulate the smart–home perspective using Planetlab hosts [28], deployed in five representative locations worldwide [29]. Cloud–service backend instances are emulated using the actual deployment inside four common datacenter locations of Microsoft Azure and Amazon AWS. Every smart home periodically measures TCP RTT latency to the eight candidate

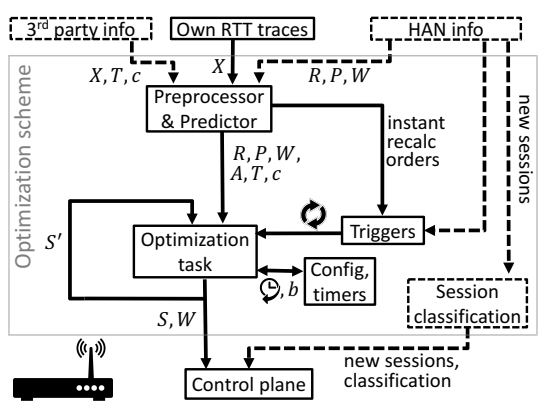

Fig. 3. *Information flow within a smart–home gateway.* Cloud status information comes from gateway's own active probing or optional 3rd–party sources. An optional information about smart–home session needs comes from monitoring the home area network (HAN) activity. Optimization–task's execution can be periodic or triggered by a significant change in HAN or Cloud status. New HAN sessions can be classified and, in turn, suitably pre–assigned to serving datacenters. The optional modules and flows, unnecessary for the general idea to work, are denoted using dashed boxes and arrows.

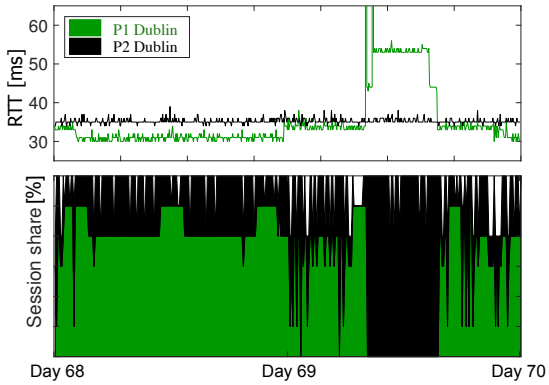

Fig. 4. *Optimization–scheme behavior.* The scheme observes preferred–DC's degrading status and, after a short delay, reacts by offloading sessions to a momentarily best–performing and suitable DC. Sessions are assigned back after the preferred–DC's status is restored. On Day 69, all six sessions were taken away of P1's DC for the period of about seven hours of excessive RTTs.

service instances. Recent parts of resulting timeseries $\vec{x}_j$ are periodically preprocessed (normalized, transformed and summarized, according to Eq. 1–4). We have experimented with various simple prediction models that consider various amounts of history, but the most recent parts of timeseries turned out to be the best predictor (likely due to irregularity of the Internet and Cloud latency distributions). Period $N = 8$ minutes minimized the prediction error given our dataset.

The optimization task then evaluates the predicted status of datacenters against smart–home session needs and periodically (here every 8 minutes) suggests optimal session assignments. The Cloud status dataset, summarized in Tab. III, was gathered using the CLAudit platform [30] that measures median RTT latency from Planetlab vantage points to Cloud datacenters every 4 minutes. This granularity does not allow for evaluation of user experience QoS measures like jitter. The CLAudit datasets are publicly available at **http://claudit.feld.cvut.cz**

The smart–home application mix (described in Tab. IV) consists of six applications: web camera, VOIP phone, HDTV, over–the–air device update, PC download and interactive home control application. Every application has a representative usage pattern, agnostic to daylight savings and weekends. This usage pattern is unknown to our optimization scheme. The application mix represents various sensitivities to timeout rate, amount of latency and deviation of latency, which are reflected in custom–sensitivities simulation scenario W'. We have also conducted a simulation scenario of default sensitivities W'', under which applications do not express any sensitivity. To discern an influence of adverse network effects, we refrain from restricting solution space – by setting negligible traffic costs $\vec{c}$, large throughputs T and large budget b. We also assume that any datacenter j can serve any session i.

TABLE III
CLOUD LATENCY DATASET SUMMARY

Attribute	Value
Timespan	10 January 2016 – 20 March 2016 (70 days)
Smart homes	Atlanta, Hiroshima, Melbourne, Belo Horizonte, Prague
Datacenters	California, Virginia, Singapore, Dublin
Cloud Providers	Microsoft Azure, Amazon AWS
RTT layer	Transmission Control Protocol (TCP)

In the following subsections, we quantify the optimization merits by contrasting performance of the optimization scheme with performance of an empirically–derived most preferred datacenter for every smart home. A preferred datacenter is defined as the datacenter receiving the largest traffic share from the smart home under the default–sensitivities W'' scenario over the 70 days (it often turns out to be one of the closest DCs to a respective smart home). The preferred datacenters are: P2's Virginia DC for the Belo Horizonte and the Atlanta smart homes; P2's Singapore DC for the Hiroshima smart home; P1's Dublin DC for the Prague smart home and P1's Singapore DC for the Melbourne smart home.

A. Merits on Datacenter end

Mitigating impact of Internet–path outages, Cloud outages and degraded datacenters reduces number of helpdesk calls, technician visits and SLA violations. The optimization scheme achieves so by observing failed or degraded network conditions and subsequently updating an assignment. Smart home thus avoids experiencing many outages and degradations (as shown using performance against ground truth – Tab. V), which could otherwise render smart devices unusable or unsafe. From the perspective of Cloud provider, gateways

TABLE IV
SMART–HOME APPLICATION MIX AND TWO APPLICATION–SENSITIVITY SIMULATION SCENARIOS

Need	Webcam	VOIP phone	HDTV	OTA device update	PC download	Home control
$r_{i,download}$	1kbps	87kbps	4.5Mbps	216kbps	5Mbps	128kbps
$r_{i,upload}$	256kbps	87kbps	1kbps	1kbps	1kbps	128kbps
$p_{i,download}$	256B	218B	1500B	1500B	1500B	128B
$p_{i,upload}$	256B	218B	1500B	1500B	1500B	128B
Can be split	Yes	No	Yes	Yes	Yes	No
Can be steered	Yes	No	Yes	Yes	Yes	No
Usage pattern	always–on	30 minutes every other daytime hour	1 hour mornings, 5 hours evenings	30 minutes once a week	6 hours evenings	10 minutes every 4 hours
Custom sensitivities						
$w'_{i,\tau}$	1	10	1	100	1	100
$w'_{i,\mu}$	10	100	10	1	1	100
$w'_{i,\sigma}$	10	100	10	1	1	1
Default sensitivities						
$w''_{i,\tau}$	1	1	1	1	1	1
$w''_{i,\mu}$	1	1	1	1	1	1
$w''_{i,\sigma}$	1	1	1	1	1	1

naturally divert traffic to a different datacenter, since degraded network conditions, once observed, are less prone to be selected by the optimization task.

We define j–th datacenter's *outage* (Eq. 14) as a period of 100% timeout rate observed by smart home. To rule out smart–home outages, the smart home simultaneously has to see at least one other candidate datacenter online.

$$\tau_u = 1 \;\wedge\; \exists v \in J : \tau_v < 1 \;\wedge\; u \neq v \tag{14}$$

We define j–th datacenter's *degradation* (Eq. 15) as a period of over 25% timeout rate, abnormally high mean latency or abnormally high standard deviation of latency.

$$(\tau_j > 0.5 \;\wedge\; \tau_j < 1) \;\vee\; \mu_j > 0.5 \;\vee\; \sigma_j > 0.5 \tag{15}$$

Due to the normalization (Eq. 1), the 0.5 abnormality threshold of μ and σ does not correspond to a single fixed value. But it guarantees a sufficiently high value, since, for every smart home, we only evaluate its respective preferred datacenter, which, under normal conditions, is expected to have $\mu \approx 0$ and $\sigma \approx 0$. The timeout rate $\tau = 0.5$ abnormality threshold directly corresponds to a 25% timeout rate.

According to the reasonable definitions of outage and degradation, the five smart homes observed a total of 35.3 hours of outage (mostly Belo Horizonte), of which 8.7 hours were observed on connections to smart–homes' preferred datacenters. Also observed on preferred connections were 65.5 hours of degradation. The optimization–scheme behavior in the presence of outage and degradation is validated in Tab. V, from which we can see that the scheme was doing a good job averting packets from all problematic datacenters.

B. Merits on Smart–home end

We use cumulative changes in f's value to demonstrate potential improvement inside a smart home – by quantifying how much can a smart home benefit from leveraging eight datacenters, compared to using just a single most–preferred datacenter. In Tab. VI, we show an upper bound of such improvement, i.e., an improvement by the optimization scheme that uses perfectly predicted Cloud latency timeseries $\vec{x}_j$ as an input. We also break down the improvement potential $\Delta(min_s f(s))$ into improvements and tradeoffs made in its respective τ, μ and σ components (e.g., Atlanta, compared to just using its preferred datacenter, has avoided all timeouts and its packets accumulated 8.7%–less latency and 3.2%–less latency deviation under custom–sensitivities scenario W'). The $\Delta(min_s f(s))$ upper bound is to be understood in the context of our scheme and our application mix. The improvement potential comes from traffic splitting and steering among momentarily well–performing datacenters, contrasted with using a single service backend of a smart–home vendor.

TABLE V

SCHEME VALIDATION – USING PERCENTAGE OF AVERTED PACKETS AND RATIO BETWEEN DECISIONS ACTUALLY MADE AND DECISIONS THAT COULD HAVE BEEN MADE TO AVERT SESSION FROM A PROBLEMATIC DATACENTER. THE UPPER BOUNDS ON PACKETS AND DECISIONS WERE DERIVED FROM THE DATASET ACCORDING TO EQ. 14 AND EQ. 15.

		Outage period		Degradation period	
Problematic datacenter	W	Averted packets	Avert decisions	Averted packets	Avert decisions
P2 Singapore	w'	99.99%	3/6	61.32%	17/42
	w''	99.99%	3/6	61.32%	16/42
P1 Singapore	w'	–	0/0	85.3%	510/564
	w''	–	0/0	85.64%	519/564
P2 Virginia	w'	83.87%	310/378	72.16%	1455/1950
	w''	84.16%	339/378	72.62%	1589/1950
P1 Dublin	w'	99.99%	5/6	74.51%	345/390
	w''	99.99%	6/6	74.92%	354/390

TABLE VI

70–DAY CUMULATIVE IMPROVEMENT OF ADVERSE NETWORK EFFECT $\Delta(min_s f(s))$, EXPRESSED AS A PERCENTAGE CHANGE. ALONG WITH PERCENTAGE CHANGES IN ITS τ, μ AND σ COMPONENTS.

Smart home	W	$\Delta(min_s f(s))$			
		$\forall k$	$k = \tau$	$k = \mu$	$k = \sigma$
Atlanta	w'	5.6%	∞	8.7%	3.2%
	w''	5.6%	∞	10.2%	2.1%
Hiroshima	w'	37.7%	∞	18.9%	69.6%
	w''	39.6%	∞	18.3%	75.7%
Melbourne	w'	42.1%	∞	81.8%	3.8%
	w''	36.3%	∞	74.4%	-0.5%
Belo Horizonte	w'	15.7%	71.1%	-6.7%	66.1%
	w''	18.7%	91.2%	-6.8%	63.5%
Prague	w'	11.9%	571.6%	-2.6%	26.3%
	w''	11.6%	689.2%	-3.8%	26.5%

Tab. VI also shows the magnitude of optimization's potential. It is high at distant and poorly–connected smart homes (Hiroshima, Melbourne), but low in the case of close proximity of smart home and datacenter (Atlanta). The lower–than–expected improvement potentials at Belo Horizonte and Prague smart homes are mainly caused by frequent tradeoffs – the optimization task sacrificed a small amount of mean latency μ (i.e., sometimes decided against a datacenter with lower latency) for significant timeout rate τ and standard deviation σ gains. These smart homes with lower improvement potential have tended to use many datacenters over the 70 days (Fig. 5).

Both the default–sensitivities W'' and the custom–sensitivities W' simulation scenario (Tab. IV) yielded similar results, suggesting low significance of sensitivities W for optimization of active sessions. However, another use of the optimization scheme is a precalculation of splits and assignments of future Cloud sessions, since, as a secondary product, the optimization task yields currently–optimal assignments of active–application categories (i.e., combinations of session needs P, R and W). Once a new session requests Cloud service, it gets classified and assigned according to the last known assignment corresponding to that category, as shown in Fig. 3. This helps to avoid suboptimal initial assignment.

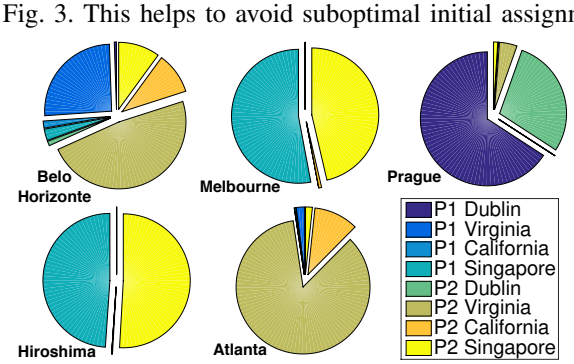

Fig. 5. *70–day optimized DC traffic shares per smart home (scenario W'')*

V. Implementation

A convenient implementation of the optimization scheme (shown in Fig. 3) is inside smart–home gateways, which are well–positioned to monitor smart–home needs together with Cloud status and, in turn, reconcile these. This section describes the naïve implementation of running the optimization scheme and related modules inside gateway's application plane. Due to the resource–intensive nature of optimal–assignment calculation and traffic forwarding, a DNS–based implementation can be used instead to relieve the header–processing burden and preserve the end–to–end principle.

The optimization scheme can be implemented as a module inside the application plane of a conventional router architecture. So, too, can optional modules for gateway's own Cloud–RTT measurements and for Cloud–status info collection from 3rd–party sources. This full–fledged architecture is shown in Fig. 6. The modules feed their Cloud–status information into the optimization scheme and the gateway itself monitors and profiles smart–home traffic for yet more information for the optimization scheme. The calculated optimal–session assignments are materialized through data–center service IP addresses and TCP/UDP ports, which flow into NAT table within gateway's data plane. Gateway operates in the NAT overload mode and continuously translates Cloud addresses and ports within inbound and outbound Cloud traffic. This implementation is applicable only to environments with sparse end–to–end traffic matrix. The optimization granularity is at the application session level, which can conveniently be represented using the flow 5–tuple (Source IP, Destination IP, Source Port, Destination Port, Protocol). To foster quality and interoperability of assignment–optimization decisions we envision future signaling to smart home, service providers and external SDN controllers.

The list of candidate datacenters comes from 3rd–party sources, provider signaling, or is preconfigured. Cloud RTTs can be obtained using gateway's own probing, as latency measurements are lightweight. Cloud throughput, traffic costs and also latency can be obtained through Cloud provider dashboards [31], [32], monitors [30], [33], benchmarks [34], [35] or SLAs. Smart–home session sensitivities and capabilities of being split or steered can be conveyed using currently unused packet fields like ToS, IP options or a novel signaling protocol. The sensitivities and capabilities are to be determined by application maker or user. Other needs such as average packet sizes or data rates are derived from traffic passing though the gateway. Some information can also be inferred from household usage patterns, possibly by using machine learning. In the case when some smart–home needs are not available, defaults are used. In the case when even Cloud status information is not available, optimization can fall back to a normal (no splitting, no steering) router forwarding. Particular sessions not to be optimized (such as CDN or cache queries) are handled by the model – using a combination of Eq. 9 and Eq. 10–type program constraints.

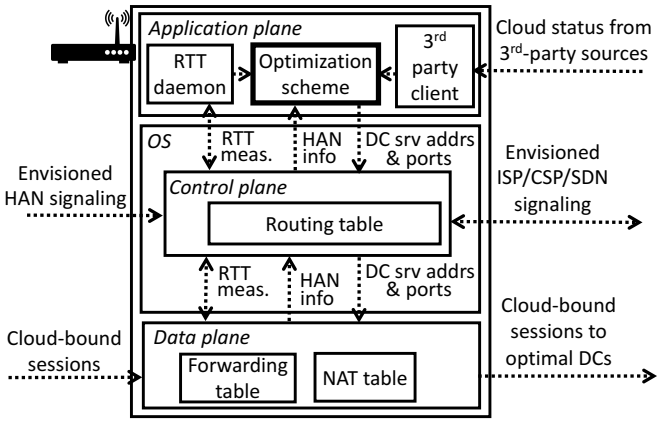

Fig. 6. *Full–fledged implementation using conventional router architecture.* Optimization scheme and related modules reside inside application plane. Modules and other sources feed their collected information into the scheme, which populates NAT table with Cloud service IP addresses and TCP/IP ports, according to which router rewrites headers of Cloud bound traffic.

Heavy–hitters or overly–sensitive sessions can trigger an optimization run and benefit from more up–to–date assignment. So, too, can the gateway itself after it observes significantly degraded Cloud status at a datacenter that serves an active session. In the case of an empty solution space, the last good assignment S' or fallback is used to ensure continuity.

Potentially–negative impacts of traffic steering and splitting on upstream will manifest through changing Cloud status and will thus be resolved in subsequent optimization runs. In the case of many distributed optimization–scheme instances, this can lead to undesired upstream oscillations, which must be addressed (e.g., by regulating the extent of assignment changes the optimization creates – by using sufficiently large ratio of history to the optimization frequency). Traffic swings, resultant from en–masse deployment of the scheme, can also be resolved by regional or global external control, such as SDN signaling. Care must also be taken when dealing with session greediness or misleading 3rd–party information.

The limits of improvement, yielded by the optimization, depend on the accuracy of predicting the Cloud status and the active smart–home session set. Additional engineering enhancements can be made to the scheme to further increase the yield, such as deferring new Cloud sessions spawned near before optimization–run completion or using high optimization frequency to leverage more up–to–date information.

For a commoditized service with stateful Cloud sessions capable of being split or steered, a service–state replication interval on the Cloud end has to be set such that an application functionality is not disrupted by an adjusted assignment. Our optimization model does not include Cloud backend costs, but under the converged prices and pay–as–you–go Cloud subscription, the bill should not change as the total Cloud backend usage remains the same.

To expedite optimal–assignment computations, the entire optimization calculation can be carried out elsewhere (e.g., Cloudlet or Fog) and resulting assignments delivered back to the gateway. Or just the incremental changes in smart–home needs and Cloud status can be used to calculate an incremental change to assignments (instead of entire program

computation). Such a change–point detection better reflects the presumably–small need to adjust assignments following an initial or sporadic computation of the entire program.

VI. Conclusion

In this work, we introduced the assignment–optimization scheme for Cloud–bound smart–home traffic, which mitigates impact of datacenter outages and degradations, reduces adverse network effects on smart–home applications and precalculates assignment for future Cloud sessions.

Results of the validation and experimental evaluation using major Cloud providers show that the optimization scheme averts between 60% and 100% packets from problematic datacenters. Also, the optimization's potential to reduce adverse network effects is as much as 42%. The scheme is general enough and its applicability is not limited to smart homes and Clouds. Environments with heavy network traffic and diverse application mix like schools and small industries would perceive even greater improvements.

Smart–home users, Cloud tenants and Cloud providers would benefit from such a scheme. Users should get better services through reduced adverse network effects and Cloud providers would have traffic from their problematic datacenters diverted naturally. Cloud tenants have long lacked a feedback from Cloud providers regarding causes of their user issues. Cloud providers, on the other hand, focus more on SLAs than customer satisfaction. Our scheme logs Cloud status information at a network edge globally, allowing for accountability and properly incentivizing all stakeholders. Ideally, a signaling among all aforementioned stakeholders would reconcile interests best and result in even more improved services.

Variations of our optimization task can be introduced by considering different measures of Cloud status, minimizing Cloud bill as objective function, tightly controlling the extent of changes for a better stability, or approximating the optimal solution from a LP corresponding to the relaxed $\mathcal{NP}$–hard MILP. State–of–the–art solutions report performance using diverse incompatible metrics – a fair comparison across these is much needed. Another challenges include development of proper Cloud–status prediction models, means of Cloud–status information sharing and instrumenting optimization scheme for large–scale deployments like Industry 4.0 or Smart cities.

Acknowledgments

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS17/091/OHK3/1T/13. We thank anonymous reviewers for their comments.

References

[1] É. Leverett, R. Clayton, and R. Anderson, "Standardisation and certification of the Internet of Things," 2017.

[2] O. Tomanek, P. Mulinka, and L. Kencl, "Multidimensional cloud latency monitoring and evaluation," *Computer Networks*, vol. 107, pp. 104–120, 2016.

[3] Z. Li, M. Liang *et al.*, "The cloud's cloudy moment: A systematic survey of public cloud service outage," *arXiv preprint arXiv:1312.6485*, 2013.

[4] O. Tomanek and L. Kencl, "Security and Privacy of Using AllJoyn IoT Framework at Home and Beyond," in *2016 2nd International Conference on Intelligent Green Building and Smart Grid (IGBSG)*. IEEE, 2016, pp. 1–6.

[5] "Amazon AWS," https://aws.amazon.com/.

[6] "Microsoft Azure," https://azure.microsoft.com/.

[7] "Nest," https://nest.com/.

[8] "Amazon Echo," https://developer.amazon.com/echo.

[9] T. Karagiannis, C. Gkantsidis *et al.*, "Homemaestro: Distributed monitoring and diagnosis of performance anomalies in home networks," *Microsoft Technical Report MSR-TR-2008-161*, 2008.

[10] Y. Ito, H. Koga, and K. Iida, "A bandwidth allocation scheme to meet flow requirements in mobile edge computing," in *2017 6th International Conference on Cloud Networking (CloudNet)*. IEEE, 2017, pp. 1–5.

[11] "Apache Kafka distributed streaming platform," http://kafka.apache.org/.

[12] T. Akidau, A. Balikov *et al.*, "MillWheel: Fault-tolerant stream processing at internet scale," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1033–1044, 2013.

[13] "Amazon Kinesis," https://aws.amazon.com/kinesis/.

[14] F. Bonomi, R. Milito *et al.*, "Fog computing and its role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[15] "AWS Greengrass," https://aws.amazon.com/greengrass/.

[16] "Azure IoT Edge," https://azure.microsoft.com/campaigns/iot-edge/.

[17] R. Rocha, "The IoT Architecture at the edge." [Online]. Available: https://www.iotcentral.io/blog/the-iot-architecture-at-the-edge

[18] Q. Zhu, R. Wang *et al.*, "IoT gateway: Bridging wireless sensor networks into Internet of Things," in *2010 IEEE/IFIP 8th International Conference on Embedded and Ubiquitous Computing (EUC)*. IEEE, 2010, pp. 347–352.

[19] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *International Conference on Future IoT and Cloud (FiCloud)*. IEEE, 2014, pp. 464–470.

[20] G. Sun, H. Yu *et al.*, "Optimal provisioning for virtual network request in cloud-based data centers," *Photonic Network Communications*, vol. 24, no. 2, pp. 118–131, 2012.

[21] J. Doyle, R. Shorten, and D. O'Mahony, "Stratus: Load balancing the cloud for carbon emissions control," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 1–1, 2013.

[22] R. S. Couto, S. Secci *et al.*, "Latency versus survivability in geo-distributed data center design," in *Global Communications Conference (GLOBECOM)*. IEEE, 2014, pp. 1102–1107.

[23] A. Forestiero, C. Mastroianni *et al.*, "Hierarchical approach for efficient workload management in geo-distributed data centers," *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 97–111, 2017.

[24] R. Tripathi, S. Vignesh, and V. Tamarapalli, "Optimizing green energy, cost, and availability in distributed data centers," *IEEE Communications Letters*, vol. 21, no. 3, pp. 500–503, 2017.

[25] A. C. Zhou, B. He *et al.*, "A declarative optimization engine for resource provisioning of scientific workflows in geo-distributed clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 647–661, 2017.

[26] M. M. S. Maswood and D. Medhi, "Optimal connectivity to cloud data centers," in *2017 6th International Conference on Cloud Networking (CloudNet)*. IEEE, 2017, pp. 1–6.

[27] A. N. Toosi, C. Qu *et al.*, "Renewable-aware geographical load balancing of web applications for sustainable data centers," *Journal of Network and Computer Applications*, vol. 83, pp. 155–168, 2017.

[28] "Planetlab: An open platform for developing, deploying, and accessing planetary-scale services," http://planet-lab.org/.

[29] "Heat map of Internet connected devices," https://www.shodan.io/.

[30] O. Tomanek and L. Kencl, "CLAudit: Planetary-scale cloud latency auditing platform," in *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*. IEEE, 2013, pp. 138–146.

[31] "Microsoft Azure status," http://status.azure.com.

[32] "AWS Service Health Dashboard," http://status.aws.amazon.com/.

[33] "Cloud Harmony," http://cloudharmony.com/.

[34] A. Li, X. Yang *et al.*, "CloudCmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 1–14.

[35] V. Uhlir, O. Tomanek, and L. Kencl, "Latency-based benchmarking of cloud service providers," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*. ACM, 2016, pp. 263–268.