RELIABLE CAPACITY PROVISIONING AND ENHANCED REMEDIATION FOR DISTRIBUTED CLOUD APPLICATIONS


## Accompanying Document for

# Deliverable D5.3: Artificial Data Traces and Workload Generator Models

**Work Package 5. Data Collection, Visualisation and Analysis**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme Under Grant Agreement Number 732667

| | |
|---|---|
| **Title:** | Artificial Data Traces and Workload Generator Models |
| **Author(s):** | Jörg Domaschka (UULM), Rafael Garcia (IMDEA), Thang Le Duc (UMU, TIETO), Mark Leznik (UULM), P-O Östberg (UMU), Sergej Svorobej (DCU), Linus Närvä (Tieto), Héctor Humanes (SATEC), Miguel Ángel López (SATEC) |
| **Editor(s):** | Jörg Domaschka (UULM), Mark Leznik (UULM), Rafael Garcia (IMDEA) |
| **Reviewed By:** | Paolo Casari (IMDEA), Keith Ellis (INTEL) |
| **Document Nature:** | Report |
| **Date:** | 30/09/2019 |
| **Dissemination Level:** | Public |
| **Status:** | FINAL |
| **Copyright:** | Creative Commons Attribution No Derivative Licences (CC BY-ND 4.0) |

## Revision History

| Version | Editor (s) | Date | Change |
|---|---|---|---|
| 0.1 | Thang Le Duc (TIETO) | 24/06/2019 | Initial ToC |
| 0.2 | Rafael García (IMDEA) | 8/22/2019 | Introductory chapter. IMDEA's contribution. |
| 0.3 | Mark Leznik (UULM) | 27/08/2019 | Added section of GAN-based Workload Generation |
| 0.3.1 | Thang Le Duc (TIETO) | 30/08/2019 | Draft of Traffic Propagation based Workload Generation |
| 0.3.2 | Sergej Svorobej (DCU) | 03/09/2019 | Simulation System Model Data Sets |
| 0.3.3 | Linus Närvä, Thang Le Duc (TIETO) | 03/09/2019 | Workload data traces for Use Case A and Use Case D1 |
| 0.4 | Rafael García (IMDEA) | 05/09/2019 | Consolidated release candidate. |
| 0.4.1 | Jörg Domaschka (UULM) | 11/09/2019 | Restructuring and alignment with Data Management Plan |

| | | | |
|---|---|---|---|
| 0.5 | Jörg Domaschka (UULM) | 15/09/2019 | Compiled new version |
| 0.5.1 | Jörg Domaschka (UULM) | 19/09/2019 | Reworked Tieto section and Simulation section |
| 0.5.2 | Linus Narva (TIETO) | 18/09/2019 | Clarification and in-depth description of Tieto section |
| 0.5.3 | Thang Le Duc (TIETO) | 19/09/2019 | Introduction of Section 4.2 and Reworked section 4.4 |
| 0.5.4 | Héctor Humanes (SATEC) | 19/09/2019 | Data sets C1 and C2 |
| 0.6 | Jörg Domaschka (UULM) | 20/09/2019 | Compiled new version |
| 0.6.1 | Sergej Svorobej (DCU) | 20/09/2019 | Updated Section “5.1.4 Example Data Set” |
| 0.6.2 | Rafael García (IMDEA) | 20/09/2019 | Formatting, Chapter contributions IMDEA |
| 0.6.3 | Linus Närvä (TIETO) | 23/09/2019 | Clarifications on input models for use case A workload traces, and a description of abbreviations in the measurements. |
| 0.6.4 | Minas Spanopoulos | 23/09/2019 | Improvements to section 5.2 |
| 0.6.5 | Mark Leznik (UULM) | 23/09/2019 | Complied new version |
| 0.7 | Jörg Domaschka (UULM) | 23/09/2019 | Released for internal review |
| 0.7.1 | Jörg Domaschka (UULM) | 24/09/2019 | Polishing of text |
| 0.7.2 | Keith Ellis (INTEL) | 24/09/2019 | Reviewed |
| 0.7.3 | Paolo Casari (IMDEA) | 24/09/2019 | Reviewed |
| 0.7.4 | Mark Leznik (UULM) | 24/09/2019 | Description of GAN data sets |
| 0.8 | Jörg Domaschka (UULM) | 24/09/2019 | Compiled new version |
| 0.9 | Jörg Domaschka (UULM) | 30/09/2019 | Removed section 5, composed release draft. |
| 0.9.1 | Paolo Casari (IMDEA) | 01/10/2019 | Approval and typos |
| 1.0 | Jörg Domaschka | 01/10/2019 | Final version |

## Executive Summary

The objective of the *WP5 – Data Collection, Visualization and Analysis* of RECAP is to provide the necessary tools for managing and refining the data needed for the rest of the work packages. This includes the collection as well as the generation of data.

Within this work package, *Task 5.3 Artificial Workload Generation* is responsible for the generation of a collection of datasets with artificial workloads, that complement the real data traces collected from industrial partners. Moreover, because publicly available workload data is scarce we provide the data as public data sets.

This document is a companion report to Deliverable D5.3 which is of type “dataset”. The aim of the report is to describe the collection of datasets that constitute D5.3 and the mathematical techniques (structural time series models, generative adversarial networks, and workload based on traffic propagation) by which one can artificially generate and/or augment such datasets.

The datasets described include real data traces collected by industrial partners and artificial data traces generated by the use of statistical models and neural networks. Each published data set can be used by the scientific and industrial community as a starting point for the modelling and experimental validation of distributed edge and cloud applications, facilitating the repeatability of the results.

An overview of all data sets published by the project is available at https://data.recops.eu/d53/.

# Table of Contents

# Table of Figures

# 1 Introduction, purpose and scope

The objective of the *WP5.- Data Collection, Visualization and Analysis* of the RECAP project is to provide the tools required to collect, manage and refine the data needed for the work packages WP6 (workload modelling), WP7 (simulation) and WP8 (application modelling). *Task 5.3 Artificial Workload Generation* is responsible for the generation of a collection of datasets with artificial workloads that complement the real data traces collected from industrial partners.

The goal of Task 5.3 is to gather and refine workloads representative of distributed cloud and edge computing applications. These workloads have been used not only for simulation purposes, but also for the evaluation of systems deployed on real infrastructures (testbeds and live systems).

Part of the delivered traces are in the track of being published as OpenData assets to fill the current lack of data on cloud infrastructure behaviour and to provide a starting point for modelling and experimental repeatability. The progress of publication of all assets in presented in this document as well as the respective DOIs are accessible under https://data.recops.eu/d53/.

## 1.1 Related Work

For development, evaluation and profiling, researches often rely on artificial data (Amlan Kar, 2019). This can be attributed to a lot of factors. Often times simple artificial data eases the development of an algorithm, allowing for quick prototyping before moving forward to production data. Artificial ground truth measurements where original ground truth data is not available can be generated for evaluation and testing purposes. In machine learning applications, artificial data is used for pre-training models first to achieve better accuracy, before moving to real data.

In the research field of and around cloud computing, monitoring virtual and physical infrastructure data is crucial for the purposes of load balancing, load prediction and designing of smart provisioning algorithms. The provisioning of such data sets is time consuming and delays the start of the actual task (such as designing a new algorithm) so that the use of already existing data is beneficial for many commercial and academic (research) projects.

Yet, the amount of available data set is very limited: Google has released a large trace of anonymized cluster job logs[1]. Yahoo has released search data (albeit with users being exposed in the process[2]); and Alibaba has published 270 GB of Open Source Data related to their data centres[3]. This shortage of open, publicly available data sets can be explained on the one hand by the effort of curating and releasing the data, but even more by reasons of commercial sensitivity and privacy of users and customers. This holds more as researchers were able to re-construct privacy sensitive information from multiple publicly available data sets, none of which contained sensitive information in itself (Narayanan & Shmatikov, 2006).

In order to improve the situation, RECAP has made an effort to release as many traces from live systems as possible. Yet, as with other data sets, the overall amount of traces that can be made public, is limited as described above.

The generation of artificial data offers the ability to not only create data sets representing the original one in terms of its (statistical) properties, but also supplementing missing data for various applications in a process called imputation (Bokde, Beck, Álvarez, & Kulat, 2018). In the current

[1] https://github.com/google/cluster-data
[2] https://www.nytimes.com/2006/08/09/technology/09aol.html
[3] https://github.com/alibaba/clusterdata

age of statistical analysis and machine learning, the specific user data is of no particular interest. Rather, data sets reflecting the statistical properties of the user mass are of far more value and interest, since they offer the possibility of performing e.g. efficient classification and predictions tasks.

Hence, the generation of artificial data sets offer several advantages for both researchers and companies alike. These range from guarding user privacy, to the ability to generate arbitrary amounts of data mirroring production behaviour. For that reason, the data sets released with D5.3 include artificial datasets generated to yield the same statistics as the original datasets the data models were trained on.

## 1.2 Structure of this Document

The remaining chapters of the document are as follows:

*Chapter 2 Use Case Overview* repeats a shorted use case description taken from the initial requirements document (D3.1).

*Chapter 3 Workload Data Traces* provides a description of the data traces that have been collected from the real infrastructures of the industrial partners, and that have been made public as a scientific source. They follow the data format presented in previous deliverables (D5.1) (D5.2).

*Chapter 4 Artificial Workload Generation* contains an introduction and relevant references to the underlying models that have been used to generate the artificial workload data traces, together with a description of the published workloads that have been generated.

*Chapter 6* summarises the document and concludes the report.

# 2 Use Case Overview

In order to keep this document self-contained and put the data presented in Sections 3 and 4, in context, this section repeats the description of the use cases from (D3.1) that are the base of all data sets.

## 2.1 Use Case A: Infrastructure and Network Management

As a provider of solutions and services for telecommunications systems as well as a provider of applications for various industry verticals, Tieto focuses on offering new innovative solutions leveraging on the possibilities enabled by 4G-and-beyond mobile technologies in conjunction with fog/cloud computing. This includes solutions for industry verticals like eHealth, eCommerce and automotive. Tieto's use case will demonstrate through the output of RECAP how the profiling and simulation of infrastructure, network functions and service function characteristics can be automated to ensure the desired QoS for the different networks managed by Tieto. Potential solutions would aim at providing support for on-demand service provisioning and capacity allocation of end-to-end services (automatic infrastructure deployment), and support for observability of the system run-time behaviour (monitoring). As part of this use case, Tieto will also provide access for the project to Tieto's testbed, which simulates how a common physical infrastructure spanning over multiple data centres with different characteristics can provide end-to-end communication and content services for different categories of horizontal content services (network slices) with different QoS requirements.

## 2.2 Use Case B: Big Data Analytics Engine

The search for an innovative solution to an existing problem is time-consuming and resource-intensive. Moreover, it involves tedious documentation reviews, and leads to non-systematic new product development, where a comprehensive and visual understanding of the emerging technologies and topics of interest is missing. The mission of Linknovate.com is to facilitate the discovery of emerging technologies, the tracking of competitor and partner activities, as well as to enable users to connect with subject matter experts behind those technologies. Hence, Linknovate.com is an innovative technology to help strategists, technologists, and any other kind of innovators when accessing new, not well understood, technology-enabled markets, and when taking new product development decisions. In particular, it supports them with keeping track of development and innovation despite the exponential growth of innovation and research data and shrinking time to market, as it enables data-driven decision-making and trend forecasting.

For that purpose, Linknovate.com generates insight by aggregating large amounts of research and scientific data, by using data mining and data analytics techniques. As of 2017, this is based on 20 million documents, over 30 million expert profiles, over 2 million entity profiles, and more than 200 million innovation topics. Technology-wise, Linknovate.com uses a distributed architecture composed of several nodes deployed in the cloud. In order to keep reduced response times when request peaks occur, Linknovate currently over-provisions nodes in the cloud. This is a clear source of inefficiency, which can be optimised in order to reduce costs and energy consumption. Dynamic provisioning together with the co-location of nodes and users is therefore the main challenges to be addressed with this use case.

## 2.3 Use Case C: Edge/Fog Computing for Smart Cities

Arguably, the current cloud computing paradigm does not provide an appropriate solution for large-scale IoT scenarios like smart cities. Smart city services can be spatially, time and/or privacy

sensitive. Having an approach that hauls data through data networks to be stored and processed in large centralised data centres whereby processed outputs are moved back again to be consumed by city infrastructure, interested entities and authorities simply does not stack up.

Fog/edge computing looks to address this challenge in devising new approaches that allow data produced in a local context to be stored and processed as close as possible to that context (Challenges and Software Architecture for Fog Computing, IEEE Internet Computing, vol 21, no 2, p 44-53, 2017).

Use Case C centres around the SATEC aim of an IoT Data Management System based on the fog computing paradigm. Specifically, Use Case C aims to demonstrate the capabilities of RECAP for automating the reallocation of resources close to the Edge/ Fog computing in order to reduce the latency for SATEC customers, and demonstrate cost savings /ease of management of resources for data centre operators. The goal is to integrate the mechanisms, techniques, key concepts, etc. generated by RECAP in prototypes that inform and/or are encompassed in the IoT Data Management System.

## 2.4 Use Case D.1: Virtual Content Delivery Networks

Network Functions Virtualisation (NFV) replaces physical network appliances with software running on servers. BT has two NFV use cases: Virtual Content Distribution Networks (vCDN) and “Cloud Connect Intelligence” that are detailed in the following.

Content Distribution Networks (CDNs) offer a distribution service to content providers that puts content on caches closer to the content consumers or end-users. A network operator such as BT is likely to have hardware from several CDN operators deployed at strategic points in its network. This creates several potential issues: it is hard to organise sufficient physical space (in exchange buildings for instance) to support all the CDN operators; a lot of energy is needed to power and cool all the equipment; a lot of physical effort is needed when a new CDN operator arrives or an existing one disappears. Such factors make it attractive to consider a Virtual CDN (vCDN) approach that aims to replace the multiple customised physical caches with a standard server and storage running multiple virtual applications per CDN operator.

RECAP will provide BT and CDN operators with the tools to plan the optimum location and amount of resources required to deploy the vCDN systems and infrastructure just in time to optimise CDN performance and resource utilisation.

# 3 Workload Data Traces

This section introduces the data sets that have been collected as part of the work in the project and that have been released as open data. Following the use case oriented nature of the project, this chapter is organised around the use cases as well as (D3.1) (D3.2).

The data sets as well as their descriptions differ largely in size and detail. This is an immediate consequence of the considerations about which data and information can be released, and their commercial sensitivity. Furthermore, it is a consequence, of the complexity of the individual scenarios used to create the respective data traces.

## 3.1 Use Case A Infrastructure and Network Management

For use case A, we provide three different data sets (A.1, A.2, A.3). They all stem from a series of experiments run over the TIETO testbed, as presented in (D4.3) and comprise performance metrics for virtualised service function chains from the telecommunications domains under various types of simulated user behaviour, e.g., movement. This setup is described in Section 3.1.1.

In order to generate the data sets, different behaviour drivers are used, named Traffic loop, Traffic model, and City simulator. They vary regarding their expressiveness and hence also with respect to the complexity of the scenarios they can create. The data sets generated from these drivers are presented in Section 3.1.2, Section 3.1.3, and Section 3.1.4.

More detailed data sets are available in the project, but remain disclosed due to commercial sensitive information[4].

### 3.1.1 Testbed description

The TIETO testbed comprises a set of physical servers managed by OpenStack, cf. (D4.3). Applications are run on that testbed as sets of Virtual Machines. For the creation of profiling data in the scope of the project, TIETO's NFVSim benchmark suite has been used.

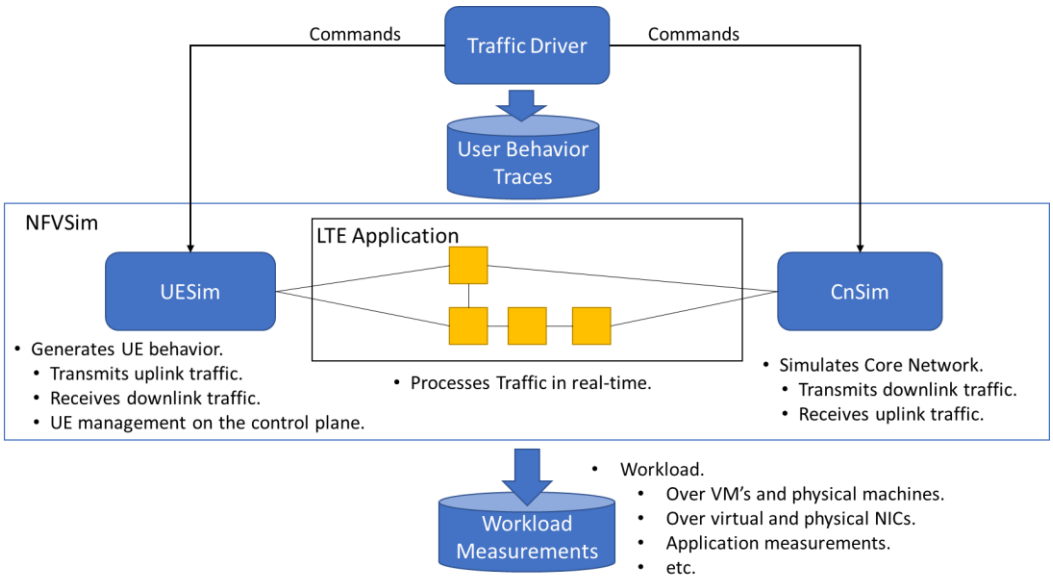



*Figure 1 High-level view on testbed set-up*

[4] Contact information of TIETO: Johan Forsman, johan.forsman@tieto.com

NFVSim emulates behaviour in the core and the edges of LTE networks, and hence creates an impact on the LTE services (application components) residing between core and edge. Figure 1 illustrates the overall set-up of NFVSim. For the generation of the data sets described in this document, the LTE application sitting in the centre consists of four components, all with a scaling factor of one: (i) a serving gateway (SGW); (ii) a packet data network gateway (PGW); (iii) eNodeB (control-plane), (iv) eNodeB (user-plane).

The LTE application resides between the traffic generator for the uplink (UESim)[5] and the traffic generator for the downlink (CNSim)[6] that both feed data into the LTE service chain. It is important to understand that the traffic generators (UESim and CNSim) generate the workload on the NFVSim application by pushing traffic into the system. Yet, they do not decide themselves which data to generate.

The decision of which data to generate, and when, is the task of the traffic behaviour drivers that send instructions to UESim (e.g., create UE, release UE) and CNSim (e.g., set data rate). The three traffic generators used in the scope of this document vary with respect to their flexibility and complexity.

### 3.1.2 Data set A.1: Traffic loop

**Scope:** This data set has been generated with the Traffic loop tool, a simple tool that allows the user to implement scripted traffic scenarios and run them in NFVSim. Due to the high level of control given over the traffic volumes and number of users, Traffic loop datasets are useful to profile different workload aspects. For instance, the test case *1000_voice_calls* was used to profile the resources consumed by a single end-user-service (voice call). Other cases include capacity tests and similar. This data set captures simple traces of hard coded tests that expose a low level of dynamicity compared to the traces that are based on the Traffic model (cf. Section 3.1.3) and City Sim (Section 3.1.4).

The Traffic loop dataset is by far the smallest data set for this use case due to its limitations (which is mainly that it is very time consuming to write extensive Traffic-loop tests). The Traffic-loop dataset also tends to focus on idealized scenarios, e.g. constant traffic rate, square wave shaped traffic rate etc.

**Infrastructure:** The data was generated using the TIETO testbed connected to the Traffic loop traffic behaviour driver, cf. Section 3.1.1.

**Methodology:** To generate the data set, one run of the Traffic-loop suite was executed. This run covers multiple scenarios, some of which are nested in sub-scenarios. Overall, the following scenarios have been executed:

- `traffic_1cells_10000ues`: A test that displays some dynamicity (albeit hard-coded) and in total sets up 10000 UEs. All UEs connect to the same cell.
- `traffic_6cells_10000ues`: Very similar to the previous test but uses six cells instead of just one. UEs are equally distributed over the cells.

[5] UE: user equipment.
[6] CN: core network

- `1000_im`: Sets up 1000 UE's that constantly use the instant-messaging (IM) end-user-service[7] (similar to the one used in the city simulator). There's no dynamicity and only one cell is used. UEs just set up, consume, and uses a constant rate of traffic until the end of the experiment.
- `1000_voice_calls`: Similar to 1000_im but uses the voice end-user-service instead of IM.
- `1000_voice_plus_data`: Similar to 1000_im and 1000_voice but uses the voice+data end-user-service instead of IM
- `traffic_pyramid_10_steps_a_20_sec`: This scenario deals with a step-wise increase and decrease in traffic and number of connected UEs. In the first ten steps, traffic volumes are gradually increased. Once they have reached the maximum, traffic volumes start to decline over 10 steps until they are back at zero. The plot of network traffic as well as number of connected UEs resembles a pyramid. Only one cell is used.
- `traffic_alternating_10_peaks_a_20_sec`: This scenario creates alternating traffic volumes and number of connected UEs. The network traffic volume as well as the number of connected UEs resembles a square wave when plotted. Only one cell is used.

The dataset was collected via the RECAP monitoring pipeline using Logstash and InfluxDB, cf. (D5.2) deployed in the TIETO testbed. The files containing the dataset have been exported from the InfluxDB database.

```
[deployment]
recap-cnsim [ACTIVE]: compute-4.domain.tld
recap-uesim [ACTIVE]: compute-4.domain.tld
recap-sgw [ACTIVE]: compute-3.domain.tld
recap-pgw [ACTIVE]: compute-3.domain.tld
recap-enb-upf [ACTIVE]: compute-1.domain.tld
recap-venb [ACTIVE]: compute-1.domain.tld
```

*Listing 1 Structure of `trafficloop__meta.txt`.*

```
name,       time,               event, run_id,  test_case
testloop,   1566894968855000000, begin, 8504,   suite_all
testloop,   1566894968918000000, begin, 27673,  init_testbed
testloop,   1566894989482000000, end,   27673,  init_testbed
testloop,   1566894989539000000, begin, 17307,  traffic_6cells_10000ues
testloop,   1566895294779000000, end,   17307,  traffic_6cells_10000ues
testloop,   1566895294846000000, begin, 18573,  tear_down_testbed
testloop,   1566895305178000000, end,   18573,  tear_down_testbed
testloop,   1566895305243000000, begin, 22811,  init_testbed
```

*Listing 2 Structure of `trafficloop_data__scenarios.csv`*

**Data format:** The data set is delivered as a zipped file that contains several files and directories. The root directory contains a file `trafficloop__meta.txt` (cf. Listing 1) and a directory `trafficloop_data`. `trafficloop__meta.txt` provides the mapping from virtual machines and the services they host to physical servers. The services in the file map to the names of the services in

[7] More accurately, traffic corresponding to the service as Traffic loop has no concept of end-user services. Just traffic volumes. The City simulator on the other hand has concepts of end-user-services and the number of traffic volumes given in the three end-user-service oriented tests correspond to those in the City simulator. Also note that the service voice+data is modelled as a single end-user-service.

Section 3.1.1 as follows: recap-cnsim → CNSim, recap-uesim → UESim, recap-enb-upf → eNodeB user plane, recap-venb → eNodeB data plane, recap-sgw → SGW, recap-pgw → PGW.

The directory contains a set of csv files with timeseries data as well as a file `trafficloop_data__scenarios.csv` file. This file contains start and end times of events that occurred during the run (e.g., the start of individual scenarios, cf. Listing 2) and hence can be used to separate the different test cases in the time series files. The format of the CSV files with time series is subject to Section 3.1.5.

**Use in the project:** Data generated with traffic loop has been used to complement City simulator data to characterize workload associated with specific traffic volumes and end-user services, cf. (D8.3). It will be used as well in the Use Case A validation as a complement to City simulator, cf. (D3.3).

### 3.1.3 Data set A.2: Traffic model

**Scope:** This data set has been generated with the Traffic model tool, a tool that enables dynamic stochastic models for traffic generation and runs them in NFVSim. Traffic model uses stochastic models to simulate UE behaviour. However, as the models are very simple (compared to the City Simulator case) the traffic patterns become redundant in longer runs (hours). The Traffic-model dataset positions itself between the City Simulator dataset and the Traffic-loop dataset both in terms of complexity and of dataset size.

**Infrastructure:** The data was generated using the TIETO testbed connected to the Traffic model traffic behaviour driver, cf. Section 3.1.1.

**Methodology:** To generate the data set, two runs of the Traffic model tool were executed, one lasting 1 hour and one that lasts 12 hours. The mathematical/stochastic models that have been used are as follows:

- Parameters that specify the scale of the system, i.e. the number of cells, UEs, etc.
- A set of traffic classes that define service characteristics (packet sizes, packet rates, etc) and a frequency which is a weight that represents the probability of traffic within the class, relative to the other traffic classes.
- A list of periods. Periods control UE attach/detach rates and are used to control the traffic volumes over time. At least one must be specified. Once the list of periods has been iterated, traffic-model will restart from the beginning of the list.
- The input model used in these tests (`input.json`) are provided with the traces.

The dataset was collected via the RECAP monitoring pipeline using Logstash and InfluxDB, cf. (D5.2) deployed in the TIETO testbed. The files containing the dataset have been exported from the InfluxDB database.

```
[deployment]
recap-cnsim [ACTIVE]: compute-4.domain.tld
recap-uesim [ACTIVE]: compute-4.domain.tld
recap-sgw [ACTIVE]: compute-3.domain.tld
recap-pgw [ACTIVE]: compute-3.domain.tld
recap-enb-upf [ACTIVE]: compute-1.domain.tld
recap-venb [ACTIVE]: compute-1.domain.tld

[trafficmodel]
test case: sample-model.json
runtime: 12 h
```

*Listing 3 Structure of `trafficmodel-12h__meta.txt`. The structure of `trafficmodel-1h__meta.txt` is similar.*

```
{
    "runtime": "10m",
    "avg_packet_size": 11200,

    "total_users": 100000,
    "max_concurrent_users": 10000,
    "max_per_cell": 1200,
    "active_users": 0,

    "number_of_cells": 9,

    "periods":
    [
        { "name": "standard", "ue_attach_rate": 0.35, "ue_release_rate": 1.0, "duration": "1m" },
        { "name": "quiet", "ue_attach_rate": 0.01, "ue_release_rate": 100.0, "duration": "1m" }
    ],

    "traffic_classes" :
    [
        {
            "type": "voice",
            "ue_hold": 90,
            "dl_packet_size": 476,
            "ul_packet_size": 476,
            "dl_rate": 33,
            "ul_rate": 33,
            "frequency": 0.54
        },
```

*Listing 4 Head of `input.json` the traffic model tool config file for specifying statistical properties of the run.*

```
name,                time,                 event,  run_id, test_case
trafficmodel-1h,     1566828730450000000,  begin,  22623,  trafficmodel_sample_model
trafficmodel-1h,     1566832330450000000,  end,    22623,  trafficmodel_sample_model
```

*Listing 5 Structure of `trafficmodel-1h_data__scenarios.csv`. The `trafficmodel-12h_data__scenarios.csv` is similar.*

**Data format:** The data set is delivered as two zipped files, one for the 1h run and one for the 12h run. Each of the zip files contains several files and directories. The root directory contains a file `trafficmodel-<run>__meta.txt`, `<run>` with being either `1h` or `12h`, cf. Listing 3. These files provide the same information as `trafficloop__meta.txt` in Section 3.1.2, and additionally specify the length of the run and reference `input.json` defining the model to be used as input (cf. Listing 4).

In addition, the zip file per data set contains a directory `trafficmodel-<run>_data`. This directory hosts a set of csv files with timeseries data as well as a file `trafficmodel-<run>_data__scenarios.csv` file. This file contains start and end times of events that occurred during the run (e.g., the start of testbed set-up or the start of the actual scenario, cf. Listing 5) and hence can be used to separate the different test cases in the time series. The format of the CSV files with time series is subject to Section 3.1.5.

**Use in the project:** Reference dataset to which the City Simulator dataset can be compared to.

### 3.1.4 Data set A.3: City Simulator

**Scope:** The city simulator drives traffic behaviour by utilizing an agent-based modelling of citizens that navigate through a city and utilize mobile services. It is by far the most extensive driver since its models operates on the basis of individual. This allows for the highest degree of dynamicity.

**Infrastructure:** The data was generated using the TIETO testbed connected to the Traffic model traffic behaviour driver, cf. Section 3.1.1.

**Methodology:** The data has been created in two long runs of City simulator. One run was active for 44 hours, and the other for 35 hours. Using a simulation speed-up factor of 4, this corresponds to a simulation time of seven and five days respectively. Both runs cover the same scenario starting on 5th June 2018 at 21:00 and using a population model of the city of Umeå in Northern Sweden[8]. This model incorporates information about the topology (streets, rivers, …) of the city, its areas (residential area, educational area, industrial area, office area), and population information such as population density and their statistical communication preferences. In addition, it introduces a topology for a mobile network. Based on this information, it generates artificial, but realistic patterns of movement over the city. For instance, employees will move to industrial and office areas during the day, while students will move to university and children to school.

The dataset was collected using the RECAP monitoring pipeline using Logstash and InfluxDB, cf. (D5.2) deployed in the TIETO testbed. The files containing the dataset have been exported from the InfluxDB database.

```
[deployment]
recap-cnsim [ACTIVE]: compute-4.domain.tld
recap-uesim [ACTIVE]: compute-4.domain.tld
recap-sgw [ACTIVE]: compute-3.domain.tld
recap-pgw [ACTIVE]: compute-3.domain.tld
recap-enb-upf [ACTIVE]: compute-1.domain.tld
recap-venb [ACTIVE]: compute-1.domain.tld

[citysim]
scenario: umea (no event)
time scale factor: 4
throughput scale factor: 1
simtime start time: 2018-06-05 21:00:00
```

*Listing 6 Structure of `citysim-35h__meta.txt`. The structure of `citysim-44h__meta.txt` is similar.*

```
name,          time,                  event,  run_id, test_case
citysim-35h,   1567172767178000000,   begin,  2190,   init_testbed
citysim-35h,   1567172787930000000,   end,    2190,   init_testbed
citysim-35h,   1567173038000000000,   begin,  15545,  citysim_umea_scenario
citysim-35h,   1567300938000000000,   end,    15545,  citysim_umea_scenario
```

*Listing 7 Structure of `citysim-35h_data__scenarios.csv`. The structure of citysim-44h_data__scenarios.csv is similar.*

[8] This population model has been created based on a data set from Umeå municipality. This data contains information on where people live and work and cell placement in the city. We cannot share this data set due to privacy reasons.

**Data format:** The data set is delivered as two zipped files, one for the 44h run and one for the 35h run. Each of the zip files contains several files and directories. The root directory contains a file `citysim-<run>__meta.txt`, `<run>` with being either `44h` or `35h`, cf. Listing 6. These files provide the same information as `trafficloop__meta.txt` in Section 3.1.2, and additionally specifies attributes of the simulation including the model to be used as input, scaling factors for time and load, as well as the simulation start time. The model data provides information of initialized end-user services, UE to Cell connectivity, UE attaches, etc. It is described in detail in Section 3.1.5 (measurements prefixed with citysim).

In addition, the zip file per data set contains a directory `citysim-<run>_data`. This directory contains a set of csv files with time series data as well as a `citysim-<run>_data__scenarios.csv` file. This file contains start and end times of events that occurred during the run (e.g., the start of testbed set-up or the start of the actual simulation, cf. Listing 7) and hence can be used to separate the different test cases in the time series. The format of the CSV files with time series is subject to Section 3.1.5. This data set contains the same types of csv files as the data sets in Sections 3.1.2 and 3.1.3, this data set also contains time series data of City simulator itself. For the sake of coherence, this aspect of the data set is also presented in Section 3.1.5.

**Use in the project:** The city simulator allows us to simulate dynamics spanning over network geography and user behaviour. Datasets generated with this tool will be used to drive the traffic in the Use Case A validation.

### 3.1.5 Measurement descriptions

This section explains the schema and semantics of the csv files of the data sets presented in Section 3.1.2, Section 3.1.3, and Section 3.1.4. All files contain time series data and follow the same structure illustrated in Listing 8.

```
name,          time,      metric1-name, metric2-name, ..., tag1-name, ...
metricset-name, timestamp, metric1-value, metric2-value, ..., tag1-value, ...
metricset-name, timestamp, metric1-value, metric2-value, ..., tag1-value, ...
metricset-name, timestamp, metric1-value, metric2-value, ..., tag1-value, ...
metricset-name, timestamp, metric1-value, metric2-value, ..., tag1-value, ...
```

*Listing 8 Structure of CSV files containing measurement time series*

The file starts with a header line denoting the name of the column. Each further line starts with the name of the metric set that the measurement in this line belongs to. The second column contains the timestamp the measurement was taken. Further columns either contain the values measured for specific metrics or the values of tags that have been set for the measurement. As described in (D5.1), tags represent meta-information attached to a measurement. Their main purpose is to enable the grouping of measurements with the same meta-data.

Table 1 presents an overview of CSV files used in the different data sets. It shows that all types of files are used in all three data sets, with the exception of files with file names starting with “citysim”.These can only be found in the City simulator data set from Section 3.1.4.

A significant detail is that the City Simulator and NFVSim handle packets per second (pps) and packet sizes differently. The City Simulator keeps track of pps and packet sizes of individual end-user-services as seen in the citysim.service_initialized metric set (Table 20). Before the traffic is generated, the pps and packet sizes from the City Simulator will be translated to the corresponding values in NFVSim, such that the total traffic volume is the same.

Table 2 to Table 23 describe the content of the files, identifying the columns used, their type (metric or tag) and a brief description of the metric semantics in cases where this is not self-explanatory. A more detailed discussion on system metrics and tags is available in (D5.1). Table 2 lists some of the abbreviation used.

*Table 1 Overview of CSV files used by the data sets*

| file name | description | Traffic loop | Traffic model | City sim |
|---|---|---|---|---|
| host.cpu.csv | Metrics for cpu consumption on physical and virtual layer, as shown in Table 3. Measured through Metricbeat. | ✓ | ✓ | ✓ |
| host.diskio.csv | Metrics for io load on physical and virtual layer, as shown in Table 4. Measured through Metricbeat. | ✓ | ✓ | ✓ |
| host.filesystem.csv | Metrics for file system load on physical and virtual layer, as shown in Table 7. Measured through Metricbeat. | ✓ | ✓ | ✓ |
| host.memory.csv | Metrics for memory load on physical and virtual layer, as shown in Table 6. Measured through Metricbeat. | ✓ | ✓ | ✓ |
| host.network.csv | Metrics for network load on physical layer, as shown in Table 7. Measured through TIETO proprietary tool. | ✓ | ✓ | ✓ |
| vm.app.cell_counter.csv | Counter metrics for cells as show in Table 8. | ✓ | ✓ | ✓ |
| vm.app.cell_timer.csv | Timer metrics for cells as shown in Table 10. | ✓ | ✓ | ✓ |
| vm.app.ctrl_meas.csv | Statistics for mobility management entity (MME) as shown in Table 12. | ✓ | ✓ | ✓ |
| vm.app.ctrlplane_traffic.csv | Statistics on control plane traffic as shown in Table 13. | ✓ | ✓ | ✓ |
| vm.app.global_counter.csv | Global counter metrics as show in Table 14. | ✓ | ✓ | ✓ |
| vm.app.uesim_user_stats.csv | Statistics on set-up UEs as shown in Table 16. | ✓ | ✓ | ✓ |
| vm.app.userplane_traffic.csv | Statistics on user plane traffic as shown in Table 17. | ✓ | ✓ | ✓ |
| vm.network.csv | Metrics for network load on physical layer, as shown in Table 18. Measured through Metricbeat. | ✓ | ✓ | ✓ |
| citysim.cell_datarate_change.csv | Statistics on CitySim as shown throughout Table 19, Table 20, Table 21, Table 22, Table 23 | | | |
| citysim.service_initialized.csv | | | | ✓ |
| citysim.ue_attach.csv | | | | ✓ |
| citysim.ue_release.csv | | | | ✓ |
| citysim.user_traffic.csv | | | | ✓ |

*Table 2: Abbreviations in the measurement description.*

| Abbreviation | Full Name |
|---|---|
| RRC | Radio Resource Control |
| UECNTX | UE Context |
| ERAB | Enhanced Radio Access Bearer |
| S1SIG | S1 Signalling |
| UPC | User Profile Component |
| RACH | Random Access Channel |
| NAS | Non-Access Stratum |
| MME | Mobility Management Entity |

*Table 3 host.cpu*

| Point | Type | Description |
|---|---|---|
| cores | metric | Number of cores of the entity (VM or server) |
| hostname | tag | Name of the host |
| idle | metric | Idle CPU metric |
| iowait | metric | IOwait CPU metric |
| irq | metric | IRQ CPU metric |
| machine_uuid | tag | UUID that identifies the VM instance or server instance |
| metric_layer | tag | Metric layer, either virtual for data from with a VM or physical for data from a host. |
| nice | metric | Nice CPU metric |
| softirq | metric | Softirq CPU metric |
| steal | metric | Steal CPU metric |
| system | metric | System CPU metric |
| test_id | tag | Identifier for the testbed deployment |
| user | metric | User CPU metric |

*Table 4 host.diskio*

| Point | Type | Description |
|---|---|---|
| devicename | tag | Identifier of the measured io device |
| hostname | tag | Name of the host |
| io | metric | |
| machine_uuid | tag | UUID that identifies the VM instance or server instance |
| metric_layer | tag | Metric layer, either virtual for data from with a VM or physical for data from a host. |
| read_bytes | metric | |
| read_count | metric | |
| read_time | metric | |
| test_id | tag | Identifier for the testbed deployment |
| write_bytes | metric | |
| write_count | metric | |
| write_time | metric | |

*Table 5 host.filesystem*

| Point | Type | Description |
|---|---|---|
| available_space | tag | |
| devicename | tag | Device the file system resides on |
| file_count | metric | |
| free_files | metric | |
| free_space | metric | |
| machine_uuid | tag | UUID that identifies the VM or server instance |
| metric_layer | tag | Metric layer, either virtual for data from with a VM or physical for data from a host. |
| mountpoint | tag | Mount point of the file system |
| test_id | metric | Identifier for the testbed deployment |
| total_space | metric | |
| used_space | metric | |
| hostname | tag | Name of the host |

*Table 6 host.memory*

| Point | Type | Description |
|---|---|---|
| actual_free | metric | |
| actual_used | metric | |
| free | metric | |
| hostname | tag | Name of the host |
| machine_uuid | tag | UUID that identifies the VM instance or server instance |
| metric_layer | tag | |
| swap_free | metric | |
| swap_total | metric | |
| swap_used | metric | |
| test_id | tag | Identifier for the testbed deployment |
| total | metric | |
| used | metric | |

*Table 7: host.network*

| Point | Type | Description |
|---|---|---|
| devicename | tag | Name of the network device (e.g. eth0) |
| hostname | tag | Name of the host |
| in_bytes | metric | Accumulated number of received bytes |
| in_drops | metric | Accumulated number of received dropped packets |
| in_errors | metric | Accumulated number of received packet errors |
| in_packets | metric | Accumulated number of received packets |
| machine_uuid | tag | UUID that identifies the server instance |
| metric_layer | tag | Metric layer (always physical in this measurement) |
| out_bytes | metric | Accumulated number of sent bytes |
| out_drops | metric | Accumulated number of sent packets dropped |
| out_errors | metric | Accumulated number of sent packet errors |
| out_packets | metric | Accumulated number of sent packets |
| test_id | Tag | Identifier for the testbed deployment |

*Table 8: vm.app.cell_counter*

| Point | Type | Description |
|---|---|---|
| cell_id | tag | Local Cell Id of the cell. This Value should be unique in NFVSim |
| Counter | tag | Identifier for the counter, ie. counter type (cf. Table 9) |
| hostname | tag | Name of the VM |
| machine_uuid | tag | UUID that identifies the vm instance |
| test_id | tag | Identifier for the testbed deployment |
| value | metric | Counter value |

*Table 9: Counter types used in vm.app.cell_counter*

| Counter name | Description |
|---|---|
| RRC.ConnEstabAtt | RRC protocol statistics |
| RRC.ConnEstabSucc | |
| RRC.ConnEstabFail | |
| RRC.ConnMax | |
| UECNTX.RelReq | UECNTX statistics |
| UECNTX.RelSuccNbr | |
| ERAB.EstabInitAttNbr | ERAB statistics |
| ERAB.EstabInitSuccNbr | |
| ERAB.EstabInitFailNbr | |
| ERAB.UsageNbrMax | |
| S1SIG.ConnEstabAtt | S1SIG Statistics |
| S1SIG.ConnEstabSucc | |

*Table 10: vm.app.cell_timer*

| Point | Type | Description |
|---|---|---|
| cell_id | tag | Local Cell Id of the cell. This Value should be unique in NFVSim |
| high_usec | metric | Highest sampled value in microseconds |
| hostname | tag | Name of the vm |
| low_usec | metric | Lowest sampled value in microseconds |
| machine_uuid | tag | UUID that identifies the vm instance |
| mean_usec | metric | Mean of sampled values in microseconds |
| samples | metric | Number of samples |
| test_id | tag | Identifier for the testbed deployment |
| timer | tag | Identifier for the timer, cf. Table 11 |

*Table 11: Timer types used in vm.app.cell_timer*

| Counter | Description |
|---|---|
| RRC.ConnEstabTimeMean | Monitors duration of RRC Connection establishments. |
| ERAB.EstabTimeMean | Monitors duration of ERAB establish time |

*Table 12: vm.app.ctrl_meas*

| Point | Type | Description |
|---|---|---|
| **hostname** | tag | Name of the vm |
| **machine_uuid** | tag | UUID that identifies the vm instance |
| **mme_bytes_rec** | metric | Accumulated bytes received by MME |
| **mme_bytes_sent** | metric | Accumulated bytes sent by MME |
| **mme_pkt_rec** | metric | Accumulated packets received by MME |
| **mme_pkt_sent** | metric | Accumulated packets sent by MME |
| **test_id** | tag | Identifier for the testbed deployment |

*Table 13: vm.app.ctrlplane_traffic*

| Point | Type | Description |
|---|---|---|
| **hostname** | tag | Name of the VM |
| **machine_uuid** | tag | UUID that identifies the VM instance |
| **rxBytes** | metric | Accumulated received number of Bytes |
| **rxPackets** | metric | Accumulated received number of packets |
| **test_id** | tag | Identifier for the testbed deployment |
| **txBytes** | metric | Accumulated transmitted number of bytes |
| **txPackets** | metric | Accumulated transmitted number of packets |

*Table 14: vm.app.global_counter*

| Point | Type | Description |
|---|---|---|
| **counter** | tag | Name/type of the counter (cf. Table 15) |
| **hostname** | tag | Name of the vm |
| **machine_uuid** | tag | UUID that identifies the VM instance |
| **test_id** | tag | Identifier for the testbed deployment |
| **value** | metric | Value of the counter |

*Table 15: Counter types used in vm.app.global_counter*

| Counter | Description |
|---|---|
| ENB.CreateUpfBearer<br>ENB.ReleaseUpfBearer | Counters to monitor UPF Bearer creation and releases in the ENodeB |
| MME.InitialCtxSetupReq<br>MME.ErabSetupReq<br>MME.UeCtxReleaseReq | MME (Mobility Management Entity) Setup and releases of different resources |
| UPC.CreateUpfBearer<br>UPC.ReleaseUpfBearer | Counters to monitor UPF Bearer creation & releases in the UPC |
| UESIM.RachTrans<br>UESIM.RachTrans<br>UESIM.RachRetrans<br>UESIM.RachSuccess<br>UESIM.RachFailures<br>UESIM.RrcConnReq<br>UESIM.RrcConnRej<br>UESIM.NasAttachReq<br>UESIM.NasDetachReq<br>UESIM.NasAttachComplete | Various statistics on resource management from the UE's. |

*Table 16: vm.app.uesim_user_stats*

| Point | Type | Description |
|---|---|---|
| **hostname** | tag | Name of the vm |
| **machine_uuid** | tag | UUID that identifies the VM instance |
| **setup_ues** | metric | Number of setup UEs at the point in time |
| **test_id** | tag | Identifier for the testbed deployment |

*Table 17: vm.app.userplane_traffic.*

| Point | Type | Description |
|---|---|---|
| **hostname** | tag | Name of the VM |
| **machine_uuid** | tag | UUID that identifies the VM instance |
| **rxBytes** | metric | Received number of bytes (accumulative) |
| **rxPackets** | metric | Received number of packets (accumulative) |
| **test_id** | tag | Identifier for the testbed deployment |
| **txBytes** | metric | Transmitted number of bytes (accumulative) |
| **txDrops** | metric | Transmitted number of packet drops (accumulative) |
| **txPackets** | metric | Transmitted number of packets (accumulative) |

*Table 18 vm.netwok*

| Point | Type | Description |
|---|---|---|
| **devicename** | tag | Name of the measured network device |
| **hostname** | tag | Name of the VM |
| **in_bytes** | metric | |
| **in_drops** | metric | |
| **in_errors** | tag | |
| **in_packets** | metric | |
| **machine_uuid** | metric | UUID that identifies the VM instance |
| **metric_layer** | metric | Metric layer, either virtual for data from with a VM or physical for data from a host. |
| **out_bytes** | metric | |
| **out_drops** | metric | |
| **out_errors** | metric | |
| **out_packets** | metric | |
| **test_id** | metric | Identifier for the testbed deployment |

*Table 19: citysim.cell_datarate_change*

| Point | Type | Description |
|---|---|---|
| local_cell_id | tag | Local Cell Id (This is unique for all cells in the City Simulator) |
| num_active_users | metric | Number of active users within the cell |
| num_users | metric | Number of users within the cell |
| rate_dl | metric | Downlink rate in packets per second |
| rate_ul | metric | Uplink rate in packets per second |
| run_id | tag | Run identifier (see test_event.test_meta) |
| test_case | tag | Identifier of the testcase |

*Table 20: citysim.service_initialized*

| Point | Type | Description |
|---|---|---|
| dl_packet_size | metric | Downlink packet size for service |
| dl_pps | metric | Downlink packets per second for service |
| duration | metric | Duration of the service |
| label | tag | Label of service (e.g. Voice, IM) |
| run_id | tag | Run identifier (see test_event.test_meta) |
| test_case | tag | Identifier to the testcase |
| ue_id | tag | Identifier of the UE |
| ul_packet_size | metric | Uplink packet size of service |
| ul_pps | metric | Uplink packets per second for service |

*Table 21: citysim.ue_attach*

| Point | Type | Description |
|---|---|---|
| cell_id | tag | Local Cell Id (This is unique for all cells in the City Simulator) |
| run_id | tag | Run identifier |
| test_case | tag | Identifier to the testcase. |
| ue_id | tag | Identifier of the UE |

*Table 22: citysim.ue_release*

| Point | Type | Description |
|---|---|---|
| run_id | tag | Run identifier |
| test_case | tag | Identifier to the testcase |
| ue_id | tag | Identifier of the UE |

*Table 23: citysim.user_traffic*

| Point | Type | Description |
|---|---|---|
| num_active_users | metric | Total number of active users |
| num_users | metric | Total number of users |
| rate_dl_bps | metric | Total downlink rate in bits per second |
| rate_dl_pps | metric | Total downlink rate in packets per second |
| rate_ul_bps | metric | Total uplink rate in bits per second |
| rate_ul_pps | metric | Total uplink rate in packets per second |
| run_id | tag | Run identifier |
| test_case | tag | Identifier to the testcase |

## 3.2 Use Case B Big Data Analytics Engine

**Scope:** This data set shows request/response metrics of Linknovate's production web server. The data set, being based on a production web server, can be used to characterize web server workloads and for the evaluation and development of forecasting models for web based network traffic.

**Infrastructure:** The data has been collected from an nginx web server in the Azure cloud (D3.1), the frontend of the production system.

**Methodology:** The data set is based on raw data coming from the web server. Due to privacy, security and commercial sensitivity, data has been anonymised and aggregated to a granularity of 30 minutes. Any information on potential users such as IP addresses has not been collected.

```
Date,                Count
2017-04-06 13:00:00, 492
2017-04-06 13:30:00, 1984
2017-04-06 14:00:00, 2055
2017-04-06 14:30:00, 2220
```

*Listing 9 First few lines of the CSV file containing the Use Case B data set.*

**Data format:** The publish time series represents a subset of two months of monitoring data. The file is in CSV format (cf. Listing 9) and contains the time interval to which the row refers (time) and the number of user requests the web server had during the interval (count).

**Use in the project:** This data set has been used in RECAP for the creation of a workload prediction model for the use case (D6.2).

## 3.3 Use Case C Edge/Fog Computing for Smart Cities

Use Case C develops a middleware for running IoT applications in Cloud/Edge environments. The application used for demonstrating the features of this application is centred around traffic management in Smart Cities. Therefore, the use case generates two types of data sets. The first is a simulated traffic pattern, while the second is the load on the application when running the traffic pattern.

### 3.3.1 Data Set C.1: Traffic Pattern

**Scope:** This dataset contains simulated traffic data. It describes the number of cars circulating in a large city including their current GPS position. By assuming that each car is sending real-time telemetry information from a collection of sensors, this dataset could be used for the simulation and evaluating of IoT and Edge computing middleware, and in particular, the design of novel communication protocols between the edge layer and the cloud layer, the development of algorithms for the optimal location of data traces and services, and the evaluation of the energy consumption of edge services vs cloud services.

**Infrastructure:** SUMO[9] simulator

[9] https://sourceforge.net/projects/sumo/files/sumo

**Methodology:** The dataset is based on the TAPAS Cologne scenario[10] that describes the traffic within the city of Cologne (Germany) for a whole day. These original files correspond to the network topology (as a graph but also with position of streets and intersections) and the traffic data (for each vehicle, starting point and time, and route followed).

These files and the SUMO simulator are used to obtain the dataset files with traffic simulation in Cologne during 4 hours with a granularity of 1 second running the following command:

```
sumo.exe -c cologne.sumocfg -b 3200 -e 17600 --fcd-output report.xml
```

Here "cologne.sumocfg" is the traffic data file downloaded from the TAPAS Cologne web site, the numbers 3200 and 17600 are the beginning and end simulation time in seconds, and `--fcd-output report.xml` is the destination file where the dataset are stored. Finally, the xml file is converted to a csv representation.

```
lat,lon,destLat,destLon,zone,ip,port,speed,weather,id,co2emission,coemission,fuelConsumption,timestamp
50.9778308936;7.03258484956;50.9858141858;6.94423488734;East;134.60.64.103;1883;0.036;Clouds;38750_38750_359_0;262400.0;16400.0;10000.0;09/07/19 7:00:01
50.9462158176;7.04923870706;50.9300651242;6.95666132456;East;134.60.64.103;1883;0.036;Clouds;34652_34652_352_0;262400.0;16400.0;10000.0;09/07/19 7:00:01
50.9456300072;6.95759197605;50.9162896095;6.98778940178;Center;134.60.64.101;1883;0.036;Clouds;47094_47094_364_0;262400.0;16400.0;10000.0;09/07/19 7:00:01
51.0331586941;6.89152220117;50.9516156516;6.85875491961;North-West;134.60.64.106;1883;0.036;Clouds;50313_50313_368_0;262400.0;16400.0;10000.0;09/07/19 7:00:02
```

*Listing 10 Structure of CSV files containing measurement time series*

**Data format:** The dataset consists of a single csv file with more than two million rows (cf. Listing 10), each one showing amongst others the id of the vehicle and its position at each second of the simulation. In detail the columns in the CSV file have the following semantics: lat, current latitude coordinate of the car; lon, current longitude coordinate of the car; destLat, destination latitude of the car; destLon), destination longitude of the car; zone, city area where the car is located; ip, IP address of the server that handles the requests for a given car. Full ip is obfuscated for security reasons; port, destination port of the server; speed, speed of the car; weather, the current weather; id, id of the car; co2emission, current emission of CO2 of the car; coemission, current emission of CO of the car; fuelConsumption, instant fuel consumption of a car; timestamp, timestamp of the telemetry message.

**Use in the project:** This dataset is injected into the SAT-IoT platform and serves as an input for the Use Case C experiments and validations. It also serves as a workload mechanism to obtain the dataset C.2.

### 3.3.2 Data Set C.2: SAT-IoT Load

**Scope:** This dataset contains the load on the Use Case C sample application, when applying the traffic pattern from data set C.1.

**Infrastructure:** For creating the data set, the Use Case application is rolled out on the UULM testbed (D4.3).

[10] https://sourceforge.net/projects/sumo/files/traffic_data/scenarios/TAPASCologne

**Methodology:** The dataset is generated by injecting the dataset C.1 into the SAT-IoT Platform in order to generate some workload into the system. Depending on the number of cars running into the city, their distribution and requests, the load of the system vary over the time.

The load is monitored using Netdata tool[11] and captured every 30 seconds through a specific purpose monitoring component deployed in the SAT-IoT platform. Then data is fed into the RECAP monitoring chain. The dataset C.2 has been generated using the following process:

1. Stop all processes in the server that may interfere with the metric collection such as system updates, file transfers… etc.
2. Data set C.1 starts to be injected into the platform.
3. Specific Metrics service starts to log the metrics.
4. The process continue until all the data set has been injected. It lasts around 2 hours.

**Data format:** The load measured is two hours long and contains the main server metrics: CPU Usage percentage, disk in usage, disk out usage, used RAM, free RAM, network traffic in, network traffic out, timestamp. The overall format is identical to the data set provided in Section 3.1.3 and has the semantics as discussed in Section 3.1.5.

## 3.4 Use Case D1 Virtual Content Delivery Networks

**Scope:** The data represents the traffic of downloading contents from the caches to serve user requests in BT's Content Delivery Network (CDN). The data has been sampled with an interval of 20 minutes and is measured in Bits Per Second.

**Infrastructure:** Data collected and distributed to RECAP for this use case comes from three caches located in inner-core nodes of the BT network in London, UK.

**Methodology:** Data was continuously collected from 2016 up to time of writing (September 2019) and provided to RECAP for workload analysis and modelling. However, due to proprietary information, only portions of data traces from 2016 to 2017 are publicly open.

The exact time stamps are blurred to hide commercially sensitive information. For further data traces, researchers can contact BT via contact information provided below[12].

Further due to proprietary reasons, data points are divided by a peak value to give a relative traffic measure hiding the absolute traffic level. In addition, peaks have been cropped to 0.85 to hide commercially sensitive information.

```
Datetime,       Location A,     Location B,     Location C
0,              0.022889678,    0.023970027,    0.009318701
1200000,        0.019705819,    0.019692336,    0.007955174
2400000,        0.018054977,    0.016038835,    0.00729537
3600000,        0.015558753,    0.01426691,     0.006146824
```

*Listing 11 First few lines of the CSV file containing the Use Case D1 data set.*

**Data format:** This data set is published a single CSV file (cf. Listing 11). Each data entry in the file is composed of a time stamp and three values of traffic/workload measurement corresponding to three network nodes where data is collected. Note that the given time stamps are relative ones

[11] https://www.netdata.cloud/
[12] Contact information of BT: Peter J. Willis, peter.j.willis@bt.com

which are calculated based on the collecting time of the first data entry, i.e., time stamp of the first entry is '0' and the next one will be the number of milliseconds elapsed from the first one, and so on.

**Use in the project:** This data set serves as the baseline for the generation of the synthetic data presented in Section 4.4. Further, enhanced versions of this data set have been used for vCDN application models presented in (D6.2) and vCDN infrastructure models and optimisation as shown in (D8.4).

# 4 Artificial Workload Generation

This chapter introduces four different approaches to the generation of artificial workload. Each of the following sub-sections contains a brief introduction to the mathematical models used to generate artificial workload data traces. Further, they provide relevant references for those interested in fully understanding the technical details of the utilised approach. Finally, each of the sub-sections lists one or more of the generated data sets.

## 4.1 Structural Models based Workload Generation

RECAP is interested in the availability of datasets describing the evolution of workloads of servers and services (applications), so that stochastic models can be trained to forecast future workloads based on past performance. Section 3 described the datasets collected by the RECAP project from the testbeds and production infrastructure of the industrial partners. However, for many reasons e.g. commercial sensitivity, privacy, etc., these datasets are insufficient for some application research/investigations, hence additional synthetic datasets had to be generated.

However, while additional datasets are desirable it is/was of upmost importance that such artificial datasets preserve the statistical properties of the real datasets collected by the RECAP project. As such, leveraging the experience of the data scientists at Google Search Infrastructure[13] RECAP developed a set of models, based on structural time series, which allowed for the generation of a collection of artificial workloads that resemble those originally collected. In particular, the approach was used to generate synthetic datasets for Use Case B Big Data Engine and Use Case C Edge/Fog Computing for Smart Cities.

### 4.1.1 Background

Structural time series models (Harvey, 1989) are a family of stochastic models for time series that includes and generalizes many standard time-series modelling ideas, including the autoregressive models and moving average models that have been already used in RECAP for forecasting purposes, like for example, ARIMA or Seasonal ARIMA models (Le Duc & Östberg, D6.1, 2018). A structural time series model expresses an observed time series as the sum of simpler components:

$$f(t) = f_1(t) + f_2(t) + \cdots + f_n(t) + \epsilon$$

Where $\epsilon$ is a white error term following a normal distribution of mean 0 and variance $\sigma^2$.

For example, one component might encode a linear trend, a cycle, or a dependence of previous values. Structural time series models allow to identify and encode assumptions about the processes that have generated the original data. In this way, structural time series models makes it possible to generate artificial data traces that have the same statistical properties from the original datasets.

### 4.1.2 Approach

The artificial workloads generated based on structural time series models were generated using the TensorFlow Probability (TFP[14]) library. TFP provides built-in support for fitting and forecasting time series using structural time series models. Because models are built inside the TensorFlow

[13] http://www.unofficialgoogledatascience.com/2017/04/our-quest-for-robust-time-series.html
[14] https://www.tensorflow.org/probability/

platform, workload generation software can take advantage of vectorised hardware (GPUs) and efficiently generate many time series in parallel.

Figure 2 shows an example of simulation. The figure corresponds to the generation of artificial workload traces, that is, number of requests, to the search engine of Linknovate. The original data traces collected from the nginx web server are depicted in blue, and the simulated data trace in orange. The data trace has been generated using a two-component model using a trend component and a cycle component.

The figure shows that the generated trace does not match the actual data set. This is less of an issue, as artificial workloads do not attempt to accurately forecast timeseries. Instead, it is important to generate a time series that has the same statistical properties than the original time series.

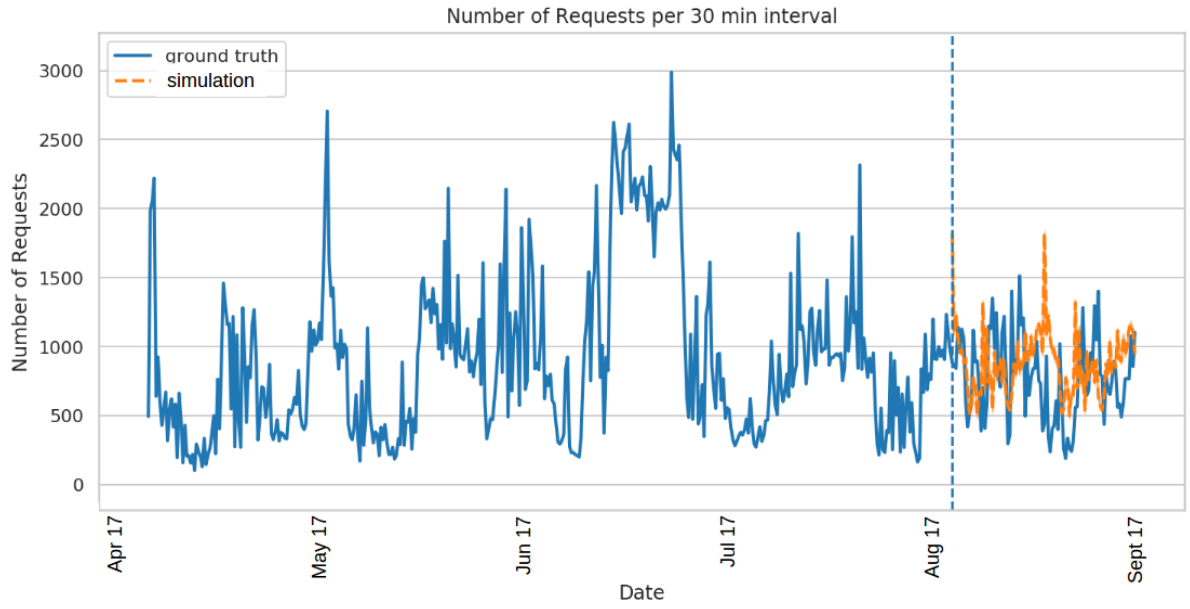



*Figure 2 Simulated Workload for a Search Engine*

### 4.1.3 Data Set STS.1

**Scope:** Workload (number of requests) to a web server, aggregated in 30 minutes' interval.

**Infrastructure:** The dataset was generated using the TensorFlow Probability library, and in particular, the Structural Time Series module.

**Methodology:** Dataset generated using a Monte-Carlo simulation (random walk) applied to a structural time series model.

```
Date,                   Requests
2019-08-06 17:12:32,    492
```

*Listing 12 First few lines of the CSV file containing the STS.1 data set.*

**Data format:** Based on the data format of the Use Case B dataset described in Section 3.2, the generated synthetic dataset is composed of two attributes: an artificial date, and the number of users during that time slot, cf. Listing 9.

**Use in the project:** The purpose of the dataset is the development of models to forecast future workloads (number of requests) given past values. Forecasted values allow to plan in advance the optimal number of servers required to deal with the expected workloads, minimizing operational expenses, and reducing energy consumption.

### 4.1.4 Data Set STS.2

**Scope:** Workload (number of cars) in a city, aggregated in intervals of one minute according to the smart city use case described in Section 2.3 and the corresponding data set from Section 3.3.1.

**Infrastructure:** The dataset was generated using the TensorFlow Probability library, and in particular, the Structural Time Series module.

**Methodology:** Dataset generated using a Monte-Carlo simulation (random walk) applied to a structural time series model.

**Data format:** Based on the data format of the Use Case C dataset described in Section 3.3, the generated synthetic dataset is composed of five attributes: an artificial date, and the number of cars during that time slot for the five identified areas: centrum, east, north, west and south, cf. Listing 11.

```
Date,                Centrum   East   North   West   South
2019-08-06 17:12:32, 492       532    123     4598   9877
```

*Listing 13 Line of the CSV file containing the STS.2 data set.*

**Use in the project:** The purpose of the dataset is the development of models to forecast future workloads (number of cars) given past values. Forecasted values allow to plan in advance the optimal location of the datasets in a cloud/edge based infrastructure. Optimal location saves costs of data transfer and improve the quality of service due to a lower (average) latency.

## 4.2 Regression-Model-based Workload Generation (Data Set RGM)

**Scope:** This data set is generated using regression-based models developed by RECAP for use case D1. The models are constructed using traffic data collected from the production CDN of BT as presented in Section 3.4. Accordingly, the generated data also represents the content-pulling traffic of caches measured in bit per second and aggregated with the frequency of 20 minutes.

**Infrastructure:** Three seasonal ARIMA models are used to generate the data at three corresponding CDN cache locations in London.

**Methodology:** The seasonal ARIMA models are developed using Python 3.6 together with the statsmodels 0.10.0 package[15], presented in detail in (D6.1) (D6.2). Based on these models, data is generated with an API call provided in the statsmodels library; and hence, reserves statistical properties of the original data set from the real production system.

To demonstrate the capability of data simulation of the developed models, a small data set was generated consisting of workload data for the last three months within the period of the real data collection. Because of being generated from the given models, the data set also inherits

[15] https://www.statsmodels.org/stable/release/version0.10.html

characteristics, induced by the protection of commercially sensitive information, from the processed data set presented in Section 3.4.

```
Datetime,       Location A,     Location B,     Location C
16070400000,    0.049722,       0.055718,       0.050941
16071600000,    -0.007088947,   0.020396702,    -0.005875195
16072800000,    0.049214776,    -0.003832634,   0.054279594
16074000000,    0.062507211,    -0.000936094,   0.071077565
```

*Listing 14 First few lines of the CSV file containing the artificial data set of Use Case D1.*

**Data format:** This data set is published as a single CSV file (cf. Listing 14). The format of each data entry is the same as that in the original data set. Note that because data is generated for the last three months of the whole real data collection period, the first timestamp of the data set is the same as the corresponding timestamp in the original data set, instead of a value of "0".

## 4.3 GAN-based Workload Generation

This section presents the approach and results of generating synthetic time-series data by using a Generative Adversarial Network (GAN). Section 4.3.1 presents background on that approach while Section 4.1.2 discusses the overall methodology used. The approach has been used to generate various synthetic data sets from multiple data sets from within the project and from outside of it. Further sections discuss the outcome of that approach.

### 4.3.1 Background

Synthetic data generation using GANs has gained popularity in recent years. From their original purpose of synthesizing images, GANs have prominently been able to generate realistic images of human faces, transfer colours from scenes, just to name a few[16].

In the RECAP case, the overall idea behind the use of GANs to generate synthetic workloads is twofold:

1. On one side, using this approach provides a "what-if" analysis on a dataset, e.g. "how would this workload look for a larger number of nodes?"
2. Additionally, due to its inherent nature, the training goal of a GAN is to estimate the probability distribution of the training data and to generate synthetic samples drawn from that distribution. Hence, when applied to a real dataset, the GAN learns to mimic its statistical pattern. When applied to sensitive data, this has the effect of reflecting the statistical properties of the data, while not exposing sensitive information.

A GAN is based on a combination of two networks, a discriminator D and a competing generator network G (cf. Figure 3). In the training phase, D is trained to distinguish real data from generated data. In parallel, G is trained to fool D by producing better and better fake data that D will accept as real.

[16] https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

### 4.3.2 Approach

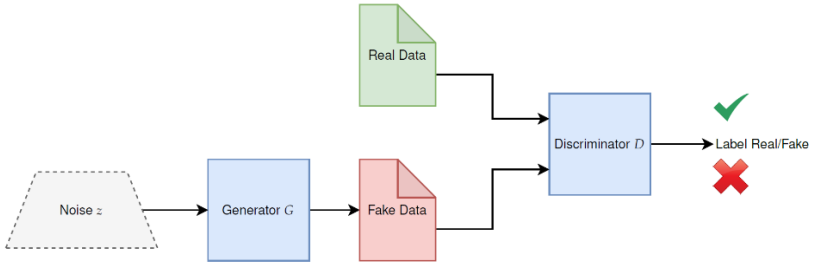



*Figure 3 An exemplary GAN architecture*

The approach is generally applicable and illustrated in Figure 4 which outlines the requisite steps. The entire workflow has been realised using the Python programming language, the Tensorflow framework for machine learning, and Keras a more user-friendly abstraction layer/wrapper for Tensorflow. For reproducibility and easier training, the process was encapsulated into a Dockerized solution, as described in detail in (Schanzel, Leznik, Volpert, Domaschka, & Wesner, 2019).

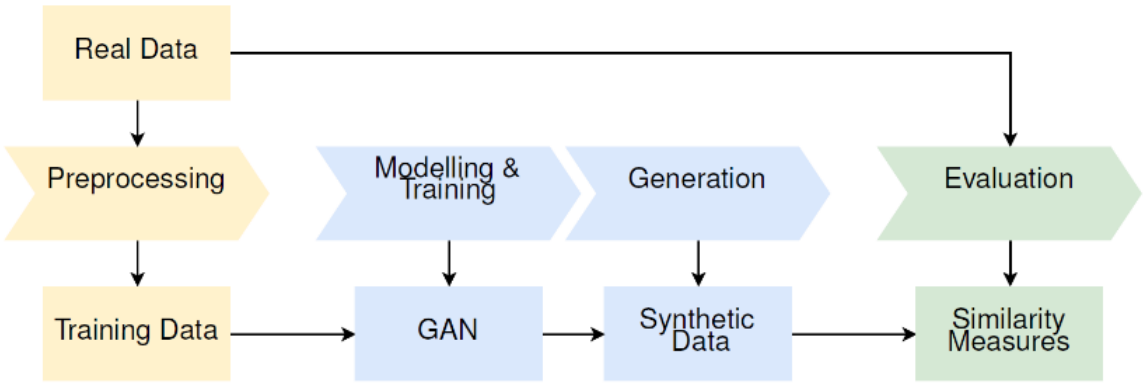



*Figure 4 The processing pipeline for generating artificial workload data*

#### *Network Design*

The discriminator used in this approach is a Recurrent Neural Network (RNN) with two Long Short-Term Memory (LSTM) layers (one hidden LSTM layer), followed by an output layer of one unit for the final classification, resulting in a single scalar value [0; 1]. The generator network consists of two recurrent LSTM layers (one being an input layer), with one fully connected output layer following the last recurrent layer with one unit per time-step. Having two recurrent layers and an additional fully connected output layer shows better results in the experiments. Figure 5 and Figure 6 depict these architectures and outline the respective input, data flow, and output.

Figure 5 shows the input noise term z, network layers and generated samples as the output. An input LSTM layer is followed by one hidden LSTM layer and a fully connected output layer, each having width m for sequences of length m.

Figure 6 shows the input sequences, network layers, and output of class labels. The network consists of an LSTM input layer, followed by a hidden LSTM layer of the same width, and a single fully connected output neuron. Input sequences of size m X n are fed through the network which assigns a class label, telling apart real (y = 0) from generated (y = 1) samples.

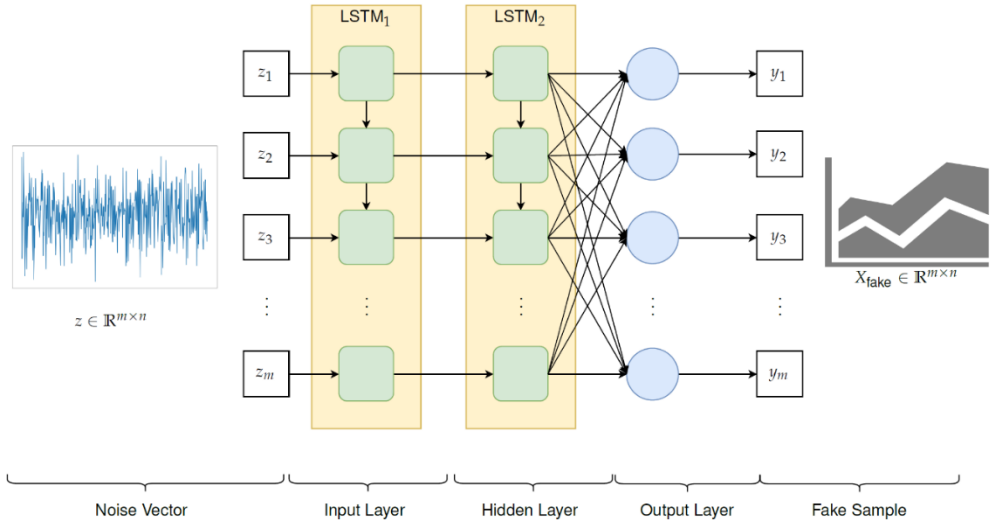



*Figure 5 Generator network layout and data flow*

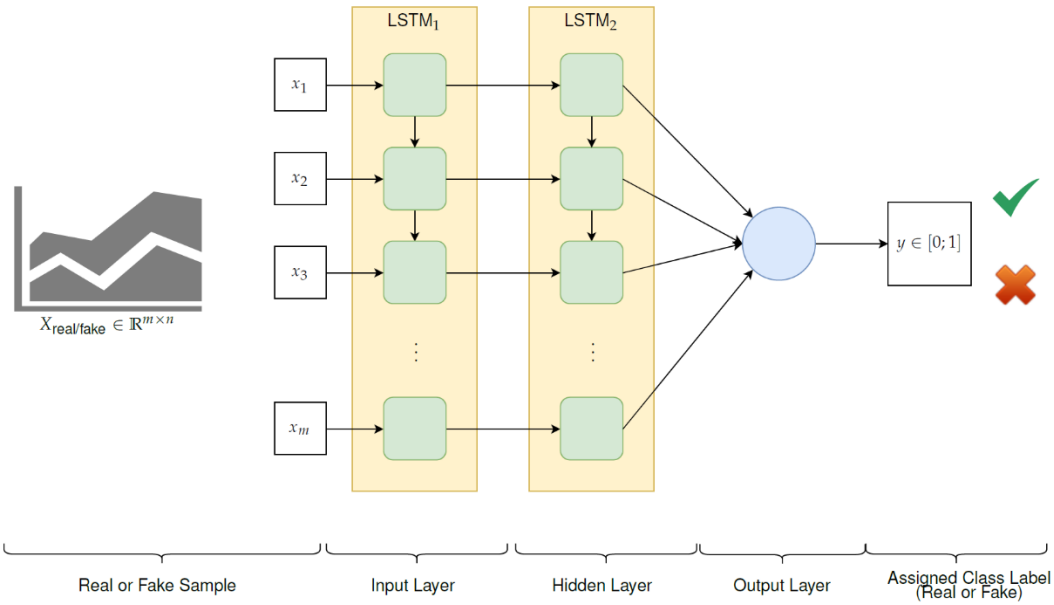



*Figure 6 Discriminator network layout and data flow*

## Pre-processing

The input data needs to be procured and then split into a training and a test data set. The pre-processing workflow takes as input a real workload data set and outputs training data for the GAN training process.

Workload traces are provided as CSV files, representing time-series data. In these files, a row denotes a time-step at a point in time and columns denote dimensions of the measurements at every time-step. The length of the input sequences is a fixed parameter of the developed GAN model. The training data must, therefore, consist of samples with an equal number of time-steps. Data normalization is a crucial preparation step for machine learning applications and can

significantly improve accuracy. The diverse ranges of different features are scaled into a range from 0.0 to 1.0 in a separate pre-processing step.

An additional optional pre-processing step is the clustering of the time series at hand, this is necessary if the workload at hand is very heterogeneous. An example of this can be seen in Figure 7. The clustering then not only allows for a successful training process, but also produces insights into the nature of the input data, e.g. showing the predominant load patterns on a cluster.

#### *Training*

Due to its LSTM design, the GAN used in this approach requires a longer training time when compared to CNN based image artificial neural networks. In our case, a computing cluster with two NVIDIA Tesla K80 GPUs was utilized for the training process.

#### *Data Generation*

After successfully training the model, arbitrary amounts of data plausibly resembling the input training data can be generated. The data however needs to be evaluated in a separate step. The results of this workflow allow to assess if and how a GAN can generate time-series of resource utilization measurements that are similar to real data.

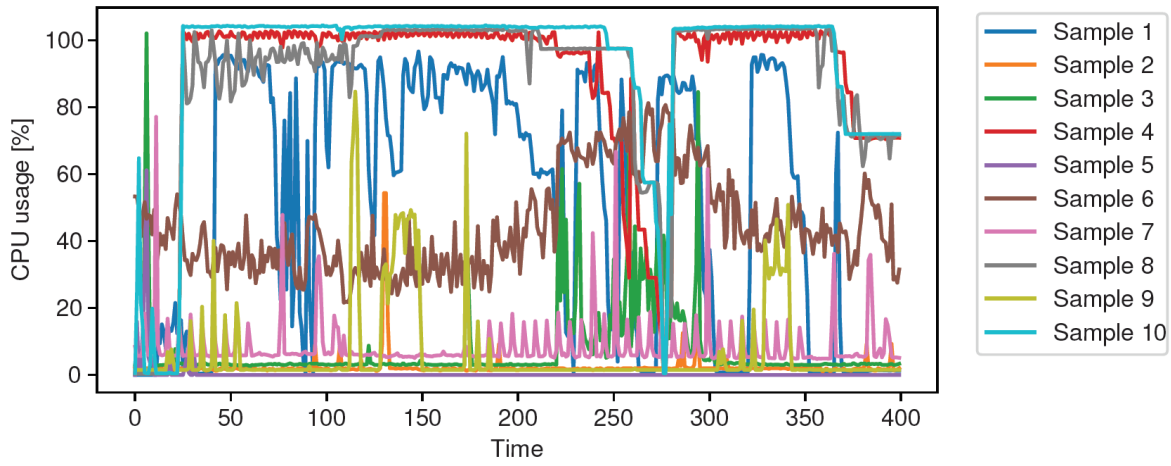



*Figure 7 10 randomly sampled series from a CPU utilization log*

To measure data similarity, an evaluation workflow is defined which takes real and synthetic samples as its input, computes statistical characteristics thereof, and presents them in a comparative view. These metrics include the autocorrelation function (ACF), the Empirical Cumulative Distribution Function (ECDF) and the cross-correlation function (CCF).

### 4.3.3 Limitations

Due to the complexity of LSTM networks, their large size does not allow them to scale out efficiently. This leads to one of the drawbacks of this approach, namely the input data size limitation. For two-dimensional multivariate time series data, this approach currently is able to handle ca. 400 time steps per dimension, when used with a univariate time series input, this number goes up to ca. 800 accordingly. This is to be corrected by using CNN-based networks in future work.

### 4.3.4 Data Set GAN.1 vCDN Univariate Locations

**Scope:** For generating this data set, we take the BT data set from Section 3.4 as a starting point and create individual synthetic time series for each of the locations. This is supposed to demonstrate the capabilities of the approach to create time series that are similar, but not identical to existing time series.

**Infrastructure:** The original data collected and distributed to RECAP comes from three caches located in inner-core nodes of the BT network in London, UK (cf. Section 3.4). The training of the neural network took place on the BinAC cluster[17].

| Cache1@BT, | Cache2@BT, | Cache3@BT |
|---|---|---|
| 0.09291529411764704, | 0.1186058823529412, | 0.12852235294117648 |
| 0.0904658823529412, | 0.12380823529411765, | 0.1610141176470588 |
| 0.09593176470588234, | 0.1201764705882353, | 0.1536258823529412 |
| 0.16791882352941176, | 0.2171729411764706, | 0.21324 |
| 0.3549658823529412, | 0.4603388235294118, | 0.4890552941176472 |
| 0.3412764705882353, | 0.4030776470588235, | 0.41772 |
| 0.3074458823529412, | 0.3532411764705882, | 0.3537764705882353 |
| 0.3197858823529412, | 0.36178470588235295, | 0.3609035294117647 |

*Listing 15 Exemplary Input Data from the vCDN Training Data*

**Methodology:** The procedure is as follows for each location: first, we pick a sequence of 500 consecutive data points for that location from the original data set (cf. Listing 15). The starting point for this sequence is picked at random. This data set is used to train a neural network in a series of epochs (cf. Section 4.3.2). In each epoch, the input data set is fed into the discriminator network. During the training process of each epoch, synthetical data is generated. In a final step, the trained network is used to generate 100 synthetical time series of length 500.

**Data format:** The data set is distributed as a zipped file. Its root directory contains three folders `location<nr>` with `nr` being 1, 2, and 3. Each of these directories contains the same set of files: (1) The `from_datapoint` file contains the randomly selected index in the original data set that is the starting point of the training data set. (2) The `training_dataset.csv` file contains the randomly selected training data set for the respective location (500 lines of scalars). As with the original data, the data points are presented as scalars with normalized measurements in the range of [0,1]. (3) The `models` folder contains both the generator model and discriminator model in .h5 format (The HDF Group). (4) The folder `generated_traces` contains one file for each of the 100 traces generated in the last epoch of the process. The naming schema for the files is `trace_<nr>.csv` with `nr` running from 0 to 99 and each file contains 499 lines of scalar data. Each data point reflect load at a specific point in time; due to the time steps used in the input data, the wall-time difference between two consecutive lines reflects a 20-minute interval. (5) Folder `training_progress` contains diagram files in png format that illustrates the time series generated in particular epochs. Using these files, it is possible to reconstruct the quality of the training by comparing the patterns at an early stage, cf. Figure 8, and at a later stage, cf. Figure 9. The naming scheme for the files is `epoch-<nr>.png` and the folder contains a file for every 10th epoch.

[17] https://uni-tuebingen.de/en/facilities/zentrum-fuer-datenverarbeitung/services/serverdienste/computing/resources/bwforcluster-binac/

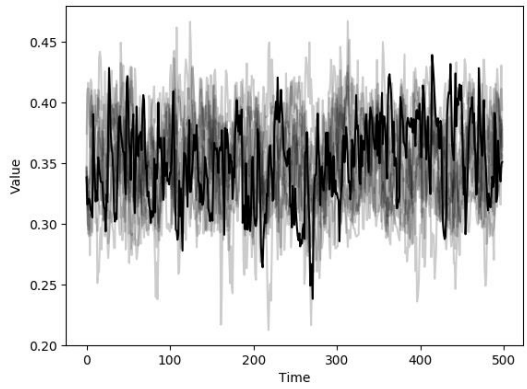



*Figure 8 Generated samples during the first epoch*

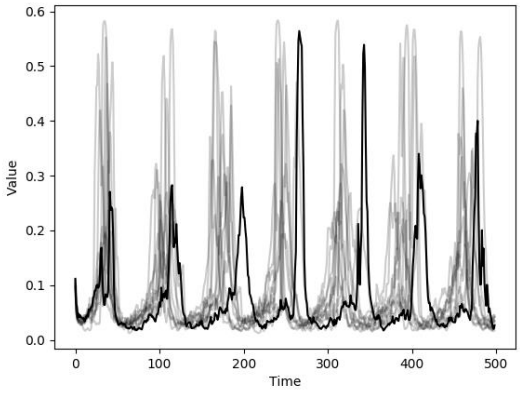



*Figure 9 Generated samples during the 500th epoch*

### 4.3.5 Data Set GAN.2 vCDN Univariate Downsampled Location

**Scope:** For generating this data set, we take the BT data set from Section 3.4 as a starting point and after downsampling it to create individual synthetic time series for each of the locations. As with GAN.1, this is supposed to demonstrate the capabilities of the approach to create time series that are similar, but not identical to existing time series. Yet, in contrast to GAN.1 were a small subset of the entire data set of a month was selected (adapting the used amount of data to technical limitations of the generating hardware), this approach downsamples from 3 data points per hour to 1 data point per hour. In consequence, three times more data points for a month contribute to the generated data set.

**Infrastructure:** The original data collected and distributed to RECAP comes from three caches located in inner-core nodes of the BT network in London, UK (cf. Section 3.4). The training of the neural network took place on the BinAC cluster[18].

**Methodology:** Most of the methodology is identical to the methodology used for GAN.1. The only differences exist due to the downsampling. Here, three samples per hour are merged into a single data point using the mean function. This leads to a total of around 750 datapoints, from which 500 data points were used to generate the output as described in Section 4.3.4.

**Data format:** The data format is identical to the one presented in Section 4.3.4.

### 4.3.6 Data Set GAN.3 vCDN Multivariate Locations

**Scope:** For generating this data set, we take the BT data set from Section 3.4 as a starting point and (in contrast to GAN.1 and GAN.2) create combined synthetic time series for all three locations. This is supposed to demonstrate the capabilities of the approach to process and create multivariate time series that are similar, but not identical to existing time series. The multivariate processing of data allows synthesizing cross site correlations.

**Infrastructure:** The original data collected and distributed to RECAP comes from three caches located in inner-core nodes of the BT network in London, UK (cf. Section 3.4). The training of the neural network took place on the BinAC cluster[19].

[18] https://uni-tuebingen.de/en/facilities/zentrum-fuer-datenverarbeitung/services/serverdienste/computing/resources/bwforcluster-binac/
[19] https://uni-tuebingen.de/en/facilities/zentrum-fuer-datenverarbeitung/services/serverdienste/computing/resources/bwforcluster-binac/

**Methodology:** The methodology is mostly identical to the one described in Section 4.3.4. Yet, as the width of the input matrix is three dimensions, its height can only be one third of the height used for GAN.1 and GAN.2. Hence, for the creation of this data set a total of 166 measurements per location were used as a multivariate input for the GAN. This led to the generation of 100 3-dimesional multivariate data sets with the length of 166 datapoints each.

**Data format:** The data format is identical to the one presented in Section 4.3.4 with the following exceptions: (i) the directory hierarchy with `location<nr>` folders does not exist. Instead all files/directories in one of the `location<nr>` files is located in the root folder of the zip archive. (ii) The content of the `training_dataset.csv` file has three columns instead of one. As in Listing 15, each column refers to a location. In return the length of the file is reduced. (iii) The same holds for files in the `generated_traces` folder. (iv) The diagrams in `training_progress` contains three plots, one for each location. An example of the resulting data can be seen in Figure 10.

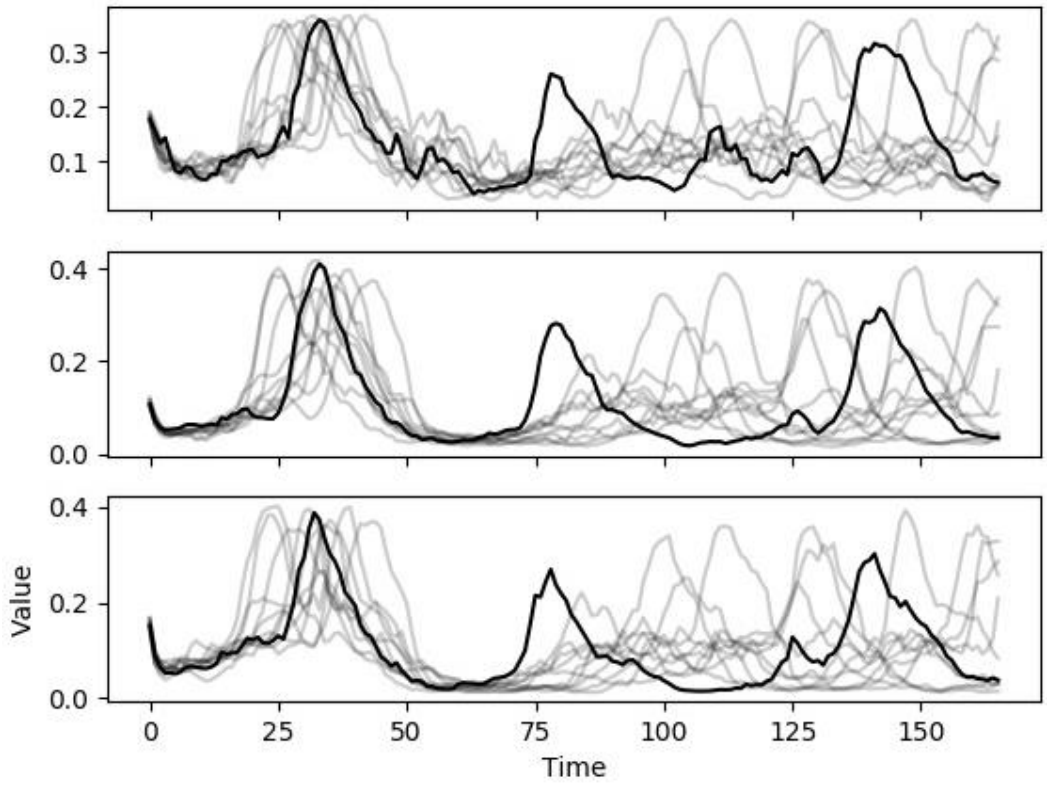



*Figure 10 Several multivariate time series generated by the GAN, showing vCDN caches for three locations with discrete timesteps*

### 4.3.7 Data Set GAN.4 vCDN Multivariate Downsampled Locations

**Scope:** The scope of this data set compared to GAN.3 is the same as the relation between GAN.1 and GAN.2.

**Infrastructure:** The original data collected and distributed to RECAP comes from three caches located in inner-core nodes of the BT network in London, UK (cf. Section 3.4). The training of the neural network took place on the BinAC cluster[20].

[20] https://uni-tuebingen.de/en/facilities/zentrum-fuer-datenverarbeitung/services/serverdienste/computing/resources/bwforcluster-binac/

**Methodology:** The methodology is mostly identical to the one described in Section 4.3.6, but uses the downsampling described in Section 4.3.5.

**Data format:** The data format is identical to the one presented in Section 4.3.6

### 4.3.8 Data Set GAN.5 Correlated CPU-Network Load

**Scope:** For producing this data set, we apply the approach from Section 4.3.2 to samples from the UULM testbed (D4.3) and generate multivariate timeseries for network and CPU load, hence correlating CPU usage and network traffic.

**Infrastructure:** The original data set samples were taken from the UULM testbed (D4.3) where they were captured using the RECAP monitoring pipeline (D5.1). The training of the neural network took place on the BinAC cluster[21].

**Methodology:** Since the start of the project several dozen gigabytes of telemetry data was stored from the servers at the UULM testbed. Based on visual impression, we selected a small subset of the overall data set. It stems from one of the servers in the testbed and covers three metrics in a time range of 24h. The raw metrics captured from the RECAP monitoring pipeline have been preprocessed as follows to keep the data set self-contained. In particular, `host.cpu.idle` (cf. Table 3) has been normalised with `host.cpu.cores`. The network metrics `host.network.in_bytes` and `host.network.out_bytes` (cf. Table 7) have been filtered such that traffic related to Virtual Machines is contained (data net in OpenStack terminology).

With measurements taken every minute, this leads to roughly 1,500 data points per metric. We applied the very same approach as in Section 4.3.4 to pick a 3-dimensional 150-element sample from this data set and then applied the very same strategy to generate data traces as before.

| idle, | datanet.bits_per_sec_in, | datanet.bits_per_sec_out |
|---|---|---|
| 14.59716666668343, | 919364.5333333333, | 675737.2 |
| 14.581499999963366, | 540711.3333333334, | 660163.4666666667 |
| 14.358500000027316, | 548399.8666666667, | 662049.3333333334 |
| 14.383833333337678, | 541546, | 656302.1333333333 |

*Listing 16 Start of the `original_dataset.csv` file*

**Data format:** The files used for this data set are identical to the one presented in Section 4.3.6 with the exception that a further file `original_dataset.csv` has been added to the zipped archive. This file contains the original data set the training the training data set is selected from. As shown in Listing 16, it contains three columns, on for each of three metrics: `idle` shows the server's idle ratio in percent with 100% meaning "all cores idle". `datanet.bits_per_sec_in` shows the average incoming bits per second in the last interval (since the last data point). `datanet.bits_per_sec_out` shows the same for outgoing traffic. This file contains 1,441 data points for each metric.

Hereby a three-dimensional multivariate time series with the length of 150 data points was used as input for the GAN to generate 100 measurements with the same output length of 150 measurements. Originally, the data sets were sampled every minute.

[21] https://uni-tuebingen.de/en/facilities/zentrum-fuer-datenverarbeitung/services/serverdienste/computing/resources/bwforcluster-binac/

### 4.3.9 Data Set GAN.6 Correlated Network Load

Follows the same approach as GAN9 in Section 4.3.8, but considers only the two-dimensional network metrics (in and out), and leaves out the CPU dimension. Due that, the traces used as input (and generated as output) can be enhanced in length and the use of 300 datapoints is possible.

## 4.4 Traffic Propagation based Workload Generation

The previous section introduced a background review of the traffic-propagation-based workload generation method and its appropriateness. This section first introduces a background review of the traffic-propagation-based workload generation method and its necessity. Next, the methodology of this workload generation is revealed (see (D6.2) for further details), and how we implement the method to retrieve artificial data sets. Lastly, a summary of the retrieved data sets (context, methodology, and structure) is presented.

### 4.4.1 Background

As discussed in (D6.2) and (Le Duc, Garcia Leiva, Casari, & Ostberg, 2019), it is often impossible and infeasible to obtain the workload/traffic measurements at any network locations where application components are deployed and operating. However, such measurements/data are required for application modelling/profiling in RECAP. This therefore leads to the demand of synthetic workload generation to fill up such missing workload data.

In WP6, we have developed a model of workload propagation which has been presented in (D6.2) in detail. The model is composed of five different diffusion techniques that are applicable for different use cases and under different assumptions related to the network topology, network links' capacity, and the distribution of users throughout the network. For the sake of workload generation, with this model and real workload data traces collected as time series at a limited number of locations, we can produce the workload traces for any or all network locations.

### 4.4.2 Approach

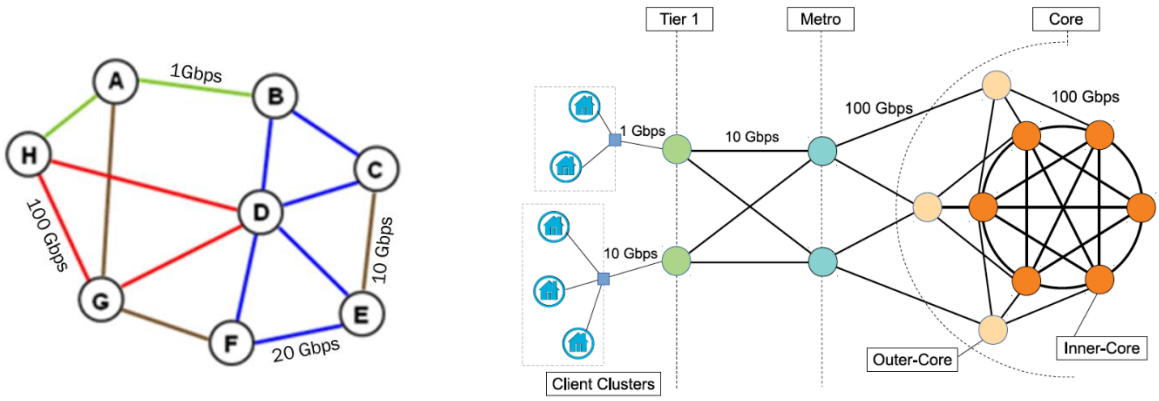
As presented in (D6.2) and (Le Duc, Leznik, Domaschka, & Östberg, 2019), we have implemented five diffusion algorithms for workload generation which can be divided into two main groups: non-hierarchical workload diffusion and hierarchical workload diffusion. Non-hierarchical diffusion algorithms include population-based, location-based, and bandwidth-based algorithm; and hierarchical diffusion ones include hierarchy-based and network-routing-based algorithm. Hereafter are brief descriptions for each algorithm including the assumptions, key inputs, and the main flow and properties (cf. Table 24). Further details about the calculations or formulae used in each task/step of the algorithms can be found in (D6.2); and the pseudo-code of each algorithm can be found in (Le Duc, Leznik, Domaschka, & Östberg, 2019).

Algorithms have been primarily implemented and tested with workload data traces collected as time series at three inner-core nodes of BT's CDN. Preliminary results have been presented in (D6.2). The representative metric of the workload in this use case is the traffic generated at caches when serving user requests. To complete the synthetic workload generation defined by this deliverable, we have finalized a complete implementation of and have executed all the algorithms to generate workload data for a typical scenario in CDN where contents are downloaded from caches centrally located at the inner-core nodes and then transferred to end-users in multi-hop manner. Regarding the required inputs for the proposed algorithms, we use the same settings as in the preliminary tests in (D6.2) in terms of network topology (see Figure 12) and real data measurements. More specifically, in the defined scenario, real traffic measurements collected at

the inner-core nodes in BT's CDN are placed at three inner-core nodes: I1, I2 and I3 in the target network. Five algorithms are executed to propagate the workload/traffic from these inner-core nodes to every other node. Results (as generated workload data traces) are collected at all nodes located from Tier-1 to outer-core layers and provided together with this document as open datasets for public use.

*Table 24: Overview of RECAP's workload diffusion algorithms*

| Diffusion Algorithm | Assumptions | Key Inputs | Description |
|---|---|---|---|
| Population-based | - Non-hierarchical network/application topologies (e.g., telecom networks, P2P applications) (see Figure 11.a)<br>- Homogeneous user behaviour | User distribution in the network | - Iterative refinement algorithms (similar to heat diffusion and spring relaxation equations)<br>- Repeatedly solve state equations to distribute workload to neighbors until the overall load distribution approaches equilibrium<br>- Algorithms highly parallelizable |
| Location-based | | Geographical node locations | |
| Bandwidth-based | | Bandwidth capacity of network links | |
| Hierarchy-based | - Hierarchical network/ application topologies (e.g., broadband networks, CDN application) (see Figure 11.b)<br>- Full mesh network of the inner-core nodes<br>- Multiple shortest path routing<br>- Homogeneous user behaviour | - Network hierarchy<br>- Bandwidth capacity of network links<br>- User distribution in the network | - Hierarchy-based user aggregation to identify the aggregated number of users at every node/location based on bandwidth capacity of neighboring links<br>- Backward workload extrapolation to collect the workload measurements from every node to the inner-code nodes<br>- Inner-Core workload extrapolation to extrapolate workload at every inner-core node (if needed)<br>- Workload propagation to distribute the workload from inner-code nodes to every node in the network |
| Network-Routing-based | | - Network hierarchy<br>- Bandwidth capacity of network links<br>- User distribution in the network<br>- A set of service (inner-core) nodes | - Routing path discovery to identify (shortest) routing paths from client-clusters to the service nodes<br>- Network-routing-based user aggregation (using routing paths)<br>- Backward workload extrapolation<br>- Workload propagation |




*(a) Non-hierarchical network topology including peer nodes without specific levels*

*(b) Hierarchical network topology with multiple levels from tier-1 to inner-core*

*Figure 11. Example of network structure/topology*

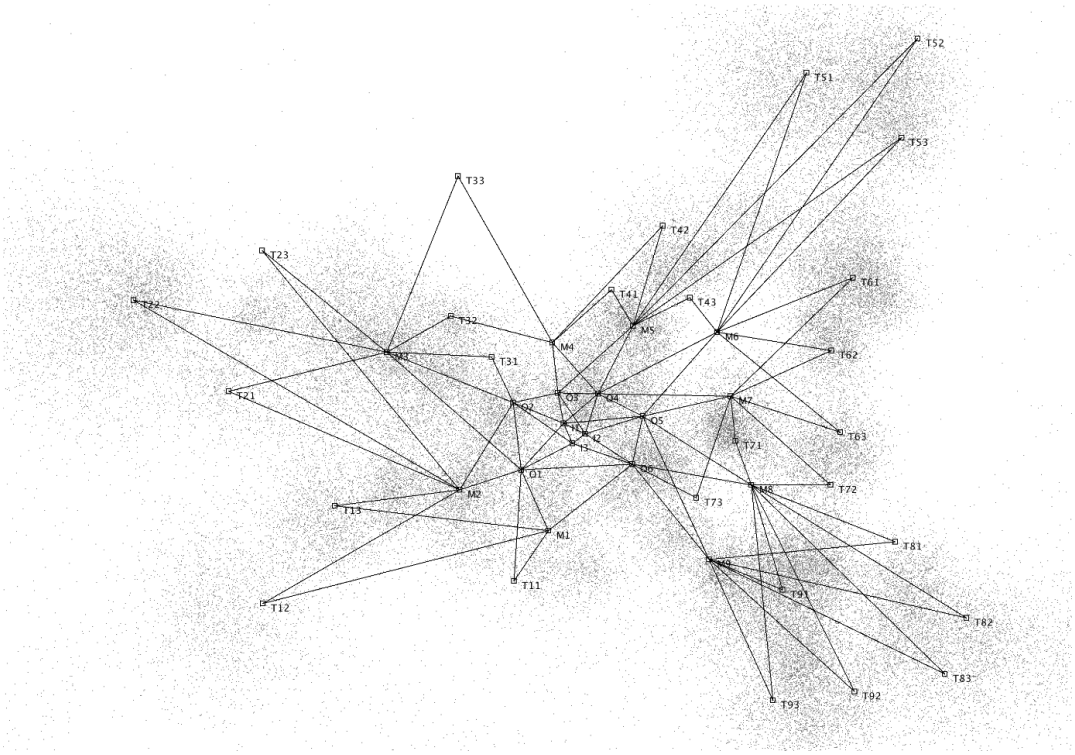


*Figure 12. Network (and population) model of the city of Umeå, Sweden.*

### 4.4.3 Data Set TRG

**Scope:** With the traffic-propagation-based workload generation, RECAP aims to generate data sets for all nodes/locations based on a small set of real traffic measurements (data) collected at a limited number of nodes in the network. Data generated and publicly open in this case is based on the real measurements in BT CDN (cf. Section 3.4) which are deployed in the three inner-core nodes of Umeå network structure (cf. Figure 12). The reason for such a setting is to show the generalisation and flexibility of our proposed workload diffusion algorithms (cf. Table 24).

**Infrastructure:** A workload propagation model is developed for the given setting and presented in (D6.2). Based on this model, we have developed five algorithms to realise this artificial workload generation task.

**Methodology:** The given real workload data sets and the network structure are input to the five algorithms so as to retrieve artificial workload data traces for every node/location in a given network.

**Data format:** The data set is released as one zipped file that contains a sub-directory for each of the five algorithms. Each of the directories contains a set of 42 single column CSV files corresponding to 42 nodes in the network together with one single mapping file (timestamps.csv) that contains the timestamps of the entries in the node files; i.e., a data entry in a node file represents a value of workload which corresponds to the timestamp entry in the mapping file at the same line. This structure avoids duplicating the same list of timestamps in every node file.

# 5 Conclusion

The objective of the *WP5.- Data Collection, Visualization and Analysis* is to provide the tools required for retrieving, managing, and refining the data needed for the RECAP project, and the goal of *Task 5.3 Artificial Workload Generation* is the generation of a collection of datasets with artificial workloads to complement the real data traces collected from the infrastructures of industrial partners. The present accompanying document to *Deliverable 5.3.- Artificial data traces and workload generator models* describes the datasets that has been made public as a result of the continuous monitoring activities of the project, together with a description of the datasets generated artificially and the stochastic models used for their generation.

RECAP's Deliverable 5.3 has contributed to address the scarcity of public available data traces by means of releasing a heterogeneous collection of annotated application datasets and real infrastructure workloads. The datasets published by Deliverable 5.3 can be used by the scientific community as a starting point for the modelling and experimental validation of distributed, edge, and fog computing applications, facilitating the repeatability of the results. The status, location, and DOI of data sets related to the project are accessible at https://data.recops.eu/, those related to this deliverable can be found at https://data.recops.eu/d53.

Deliverable 5.3 not only includes a description of the available datasets, but also, a description of the mathematical techniques that have been used to generate them (structural time series models, generative adversarial networks, and workload based on traffic propagation), so that additional data traces, preserving the statistical properties of the original ones, could be generated.

# 6 References

Amlan Kar, A. P.-Y. (2019, April 25). *Meta-Sim: Learning to Generate Synthetic Datasets.* Retrieved from https://arxiv.org/abs/1904.11621#.

Bokde, N., Beck, M. W., Álvarez, F. M., & Kulat, K. (2018, December). A novel imputation methodology for time series based on pattern sequence forecasting. *Pattern Recognition Letters*.

Domaschka, J., Area, C., López, M. A., Willis, P., Noya, M., Parapar, J., . . . Närvä, L. (2017). *Initial Requirements.* RECAP Deliverable.

Domaschka, J., Griesinger, F., Volpert, S., Willis, P., Sundqvist, T., & Narvä, L. (2019). *Improved Testbed Configuration.* RECAP Deliverable.

Domaschka, J., Lopez, M., Humaes, H., Ocón, J., Willis, P., Noya, M., . . . Le Duc, T. (2018). *Final Requirements and Validation Plan.* RECAP Deliverable.

Domaschka, J., Narvä, L., Östberg, P.-O., Svorobej, S., Garcia, R., Ellis, K., & Griesinger, F. (2019). *Initial Integrated Prototype.* RECAP Deliverable.

Garcia, R., Casari, P., Domaschka, J., Griesinger, F., Forsman, J., Area, C., . . . Le Duc, T. (2018). *Initial Data Acquisition and Analytics Models.* Casari, Paolo; Domaschka, Jörg.

Garcia, R., Casari, P., Domaschka, J., Griesinger, F., Narvä, L., Lopez, M., . . . Willis, P. (2019). *Final Data Acquisition and Analytics Models.* RECAP Deliverable.

Hao , Z., Novak, E., Yi , S., & Li, Q. (2017). *Challenges and Software Architecture for Fog Computing, IEEE Internet Computing, vol 21, no 2, p 44-53.*

Harvey, A. C. (1989). *Forecasting, structural time series models and the Kalman filter.* Cambridge University Press.

Le Duc, T., & Östberg, P.-O. (2018). *Initial Workload, Load Propagation and Application Models.* RECAP Project Deliverable.

Le Duc, T., Garcia Leiva, R., Casari, P., & Ostberg, P.-O. (2019, August). Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. *ACM Computing Surveys, 52*(5), 39. doi:https://doi.org/10.1145/3341145

Le Duc, T., Leznik, M., Domaschka, J., & Östberg, P.-O. (2019). Workload Diffusion Modeling for Distributed Applications in Fog/Edge Computing Environments. *[Submitted].*

Le Duc, T., Östberg, P.-O., García, R., Casari, P., Loomba, R., & Leznik, M. (2019). *Final Workload, Load Propagraion, and Application Models.* RECAP Deliverable.

Loomba, R., Ellis, K., Casari, P., Garcia, R., Duc, T. L., Östberg, P.-O., & Forsman, J. (2019). *Final Infrastructure Orchestration and Optimization.* RECAP Deliverable.

Loomba, R., Quinn, R., Ruane, S., Ellis, K., Le Duc, T., Östberg, P.-O., . . . Fernandez Anta, A. (2019). *Final Infrastructure Modelling and Resource Mapping.* RECAP Deliverable.

Narayanan, A., & Shmatikov, V. (2006). How To Break Anonymity of the Netflix Prize Dataset. arXiv.

RECAP Consortium. (2019). *D3.3 Validation Report.* RECAP Deliverable.

RECAP Consortium. (2019). *D4.4 Final System Architecture and Integration.* RECAP Project Deliverable.

RECAP Consortium. (2019). *Final RECAP simulation platform.* RECAP Deliverable.

Schanzel, B., Leznik, M., Volpert, S., Domaschka, J., & Wesner, S. (2019). Unified Container Environments for Scientific Cluster Scenarios. *bwHPC Symposium.* Tübingen.

The HDF Group. (n.d.). HDF5 File Format Specification Version 3.0. *Website*. Retrieved from https://portal.hdfgroup.org/display/HDF5/File+Format+Specification