Workload Diffusion Modeling for Distributed Applications in Fog/Edge Computing Environments

Thang Le Duc
Department of Computing Science
Umeå University
90187 Umeå, Sweden
thang@cs.umu.se
thang.leduc@tieto.com

Mark Leznik, Jörg Domaschka
Institute of Information Resource Management, Ulm University
89081 Ulm, Germany
{mark.leznik,joerg.domaschka}@uni-ulm.de

Per-Olov Östberg
Department of Computing Science
Umeå University
90187 Umeå, Sweden
p-o@cs.umu.se

ABSTRACT

This paper addresses the problem of workload generation for distributed applications in fog/edge computing. Unlike most existing work that tends to generate workload data for individual network nodes using historical data from the targeted node, this work aims to extrapolate supplementary workloads for entire application / infrastructure graphs through diffusion of measurements from limited subsets of nodes. A framework for workload generation is proposed, which defines five diffusion algorithms that use different techniques for data extrapolation and generation. Each algorithm takes into account different constraints and assumptions when executing its diffusion task, and individual algorithms are applicable for modeling different types of applications and infrastructure networks. Experiments are performed to demonstrate the approach and evaluate the performance of the algorithms under realistic workload settings, and results are validated using statistical techniques.

CCS CONCEPTS

• **Networks** → **Traffic engineering algorithms**; *Network simulations*; *Network measurement*; • **Computing methodologies** → **Model development and analysis**; • **Mathematics of computing** → Network flows.

KEYWORDS

Workload generation; load propagation; load diffusion; edge computing; fog computing; cloud computing

ACM Reference Format:
Thang Le Duc, Mark Leznik, Jörg Domaschka, and Per-Olov Östberg. 2020. Workload Diffusion Modeling for Distributed Applications in Fog/Edge Computing Environments. In *Proceedings of the 2020 ACM/SPEC International Conference on Performance Engineering (ICPE '20), April 20–24, 2020, Edmonton, AB, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3358960.3379135

ICPE '20, April 20–24, 2020, Edmonton, AB, Canada

ACM ISBN 978-1-4503-6991-6/20/04…$15.00
https://doi.org/10.1145/3358960.3379135

1 INTRODUCTION

Workload analysis and modeling aim to construct models that provide insights into workload behavior and enable predictions of workload changes over time. Workload analysis has gained increased attention as workload characteristics have been demonstrated to have great impact on the performance and resource utilization of distributed systems - in particular cloud/fog/edge applications where network awareness and accurate understanding of workload variations are required to efficiently evaluate and optimize systems [13]. To facilitate workload modeling, large volumes of workload data from real systems are needed. Unfortunately, numerous studies (including, e.g., [5, 14, 21, 23]) have demonstrated a real lack of publicly available and usable data traces, driving a need for tools and techniques for synthetic and artificial workload generation for distributed systems.

For distributed fog/edge applications, workload behavior models for individual application component need to be complemented with *workload propagation models* [17], which model how workloads propagate through or among components in distributed applications, and how workload fluctuations at component level impact the performance of other components, the application as a whole, and the infrastructure resources (servers and networks) onto which components are deployed. Conceptually, lacking any of these models can be interpreted as incomplete knowledge about the workload behavior of applications, which illustrates the importance of capturing the complete behaviour of applications and workloads in monitoring and data collection. However, in reality it is often infeasible to obtain correlated workload measurements at all network locations where application components are deployed, especially in large scale production systems and/or fog/edge computing environments. This shortage of workload data availability introduces difficulties and inaccuracies in workload analysis and modeling, and drives a need for artificial workload generation techniques capable of augmenting available monitoring data.

This work focuses on an area not thoroughly explored in literature: extrapolation of supplementary workloads at all (application / infrastructure) node locations of distributed systems, so that given measurements at a limited number of locations fog/edge applications can completely model their workload behaviors as well as the propagation of workloads within applications. This paper tackles the problem using a comprehensive approach of workload diffusion, which is formed based on various factors including application types, the distribution of application users throughout the network infrastructure, the bandwidth of network links, neighborhood clustering information, and the network routing principles.

A general framework for workload diffusion is proposed and five algorithms using different diffusion techniques for workload generation are presented. These techniques are grouped into two categories: non-hierarchical and hierarchical diffusion. More specifically, with the given workload traces measured at some network (source) nodes, the non-hierarchical diffusion disseminates workload data to other (destination) nodes omnidirectionally, while the hierarchical algorithms use information about the application / network hierarchy to constrain workload propagation to predetermined directions. Non-hierarchical diffusion algorithms include location-based, link-capacity-based, and population-based diffusion; and the hierarchical ones are neighborhood-based and network-routing-based diffusion. Note that workload data considered in this paper are given as *time series*. The diffusion algorithms presented are thus required to not only calculate workload propagation patterns for individual timestamps, but also construct artificial workload time series at node locations.

To the best of our knowledge, this is the first paper addressing workload diffusion for distributed applications within fog/edge computing environments. The key contributions of the paper include: (i) an investigation of the problem of workload generation based on a limited number of measurements for different types of distributed applications in diverse network architectures; (ii) a framework for workload diffusion defining five algorithms that employ individual techniques to generate workloads using different types of knowledge of applications and underlying networks; and (iii) experiments that illustrate the approach and evaluate the performance of the algorithms. Validation of results is presented along with a discussion of the underlying modeling techniques.

2 LITERATURE REVIEW

A wide range of approaches has been proposed for analysis and modeling of distributed workloads in the literature, as evident by the number of surveys of workload analysis and modeling techniques. [3, 11, 13] review large bodies of work adopting both statistical analysis and machine learning to address the problem of workload modeling and prediction. Various methods of data processing, feature selection, and workload classification are surveyed, and applications of the techniques are discussed. Several recent studies address the increasingly common problem of diversity and heterogeneity in application and system types. [1, 7, 19] propose different approaches to characterize workloads and/or applications, and formulate resources usage patterns to construct workload profiles. With such understanding of the workload behavior, workload models can be derived and utilized for future predictions.

The topic of workload generation has been extensively studied and numerous applications (including, e.g., [5, 9, 15, 20, 23, 26]) have been demonstrated. A comprehensive survey on workload generators for web-based systems is carried out by [6]. The survey presents an analysis on characteristics and properties of workloads from various types of web applications, followed by an investigation on different types of workload generators and their operational as well as data models. As mentioned in [24], most of the proposed schemes fall into two categories: data-driven and model-driven techniques. Generally, data-driven techniques reproduce workloads using sampled historical workload, while model-driven techniques construct mathematical models of historical workloads and generate new workloads using parametrizations of the developed models. Two recent studies on workload generation shown below well illustrate these approaches. [4] introduces a technique to dynamically generate scalable workloads that is realized in a prototype system called Durango. The system integrates multiple tools to enable application performance modeling and simulation, and workload generation. [16] presents a workload generator called CloudPump, which adopts a generalized workload model developed with various core concepts, e.g., the operation, connection, transaction, session, in order to be capable of flexibly implementing any workload types. The generator operates in two modes (request initiator and workload consumer), and also supports synthetic workload generation and workload profile replaying. Although these studies have investigated various types of workload and application, and also applied multiple approaches for workload generation, neither aims at an analysis on the mutual effects of workload variations at different network nodes that exist in fog/edge applications.

3 MODELING TECHNIQUES

3.1 Workload Time Series

A time series is a set of measurements at different time stamps which are aligned in a chronological order, e.g., $\{x_1, x_2, ..., x_k\}$ with k time stamps. It can be either discrete time series or continuous time series [2]. In this paper, workload of a node is measured as traffic exiting from the node, as detailed in 5.1, and then provided as a time series. Assume that workload time series are measured with the same set of time stamps, manipulations on workloads are applied merely to the measured values without a consideration on the time stamps. In this way, a workload can be interpreted as a vector (of workload values); hence, the operator '$\times$' between a scalar and a workload is defined as a multiplication of the vector by the scalar and results in a workload. Similarly, the operator '+' of two workloads is defined as a vector addition, and a sum of multiple workloads, denoted by $\sum^{\star}[\cdot]$, is defined as the sum of vectors. Both also result in a workload.

3.2 Network Model

The network infrastructure is modeled as a graph $G = (V, E)$, where V and E represent the set of nodes and the set of edges of the network, respectively. An edge $(u, v) \in E$ represents a link connecting nodes $u, v \in V$, with which a bandwidth capacity $B(u, v)$ is associated. Every link is assumed to be full duplex and symmetric, i.e., $B(u, v) = B(v, u)$, and that the traffic in one direction does not affect the traffic in the opposite one on a link. A node is associated with coordinates including latitude and longitude to indicate its geographical position in the network deployment plan. In addition, a node v is also associated with a number of users $m(v)$, i.e., there are $m(v)$ users attached to node v. For simplicity, all algorithms assume homogeneous user behavior, i.e., all users behave similarly in all for the algorithms relevant ways, e.g., all users are assumed to issue similar types of requests to services deployed in the network. This (user behavior) assumption is the main simplification in the modeling and future work will addresses relaxation of this assumption through integration of more complex statistical treatment of user behavior.

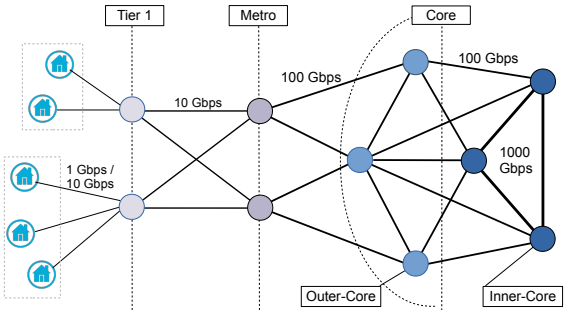


Figure 1: An exemplary four-tier network model.

In a hierarchical network, a node v is assigned a node level $L(v)$. The number of levels can be different from application to application, or depends on the management viewpoint of network operators. Without loss of generality, this paper assumes the presence of at least four hierarchical levels corresponding to four layers (or tiers) of nodes, namely inner-core, outer-core, metro, and tier-1 (T1) access layer as illustrated in Figure 1. These layers are placed in that ascending order, which means the inner-core is the most centralized layer and T1 is the edge of the network topology. In this sense, an inner-core node I is said to be at lower level than an outer-core node O, i.e. $L(I) < L(O)$. This type of model, which essentially captures hierarchy of nodes and redundancy of links, can be seen to be representative of a wide variety of distributed applications and networks, e.g., content delivery networks (CDNs) and the British Telecom BT 21CN production network [12].

For notation, let $N(v)$ denote the set of neighboring nodes of v. A node u is called an upper node of v if $u \in N(v)$ and $L(u) > L(v)$. This also implies that v is a lower node of u. $U(v)$ and $D(v)$ are used to denote the set of upper nodes and lower nodes of a node v, respectively. In addition, assume there is a set of *service nodes* designated as destinations of all user requests in the network and these nodes are located in the inner-core layer. User requests are directionally forwarded towards the service nodes from edge layers to more central layers, and responses from the application/service return to the users in the reverse direction.

If an application adopts network routing in its service chain, a user request of the service is forwarded following predetermined routing paths from the request origin down to service nodes. It is also assumed to use the multi-path routing mechanism [10] in the application. Let $\mathcal{P}(v)$ denote a set of routing paths originated from node v destined to every service node. A path p of length k is defined as a sequence of nodes $\{v_0, v_1, v_2, ..., v_k\}$ in which v_0 is the request origin and v_k is a core/service node. This also implies that path p includes a set of links $e_i = (v_i, v_{i+1})$ or it is trivial to derived such links of path p. Given a set of paths $P \equiv \mathcal{P}(v)$, let $D^P(v)$ denote the set of lower nodes of v, each of which is a lower node of v and included in at least one path $p \in P$ going through v.

4 NON-HIERARCHICAL AND HIERARCHICAL WORKLOAD DIFFUSION

To address a broad range of scenarios where different types of information of applications and networks are available, we propose five workload diffusion algorithms classified into non-hierarchical and hierarchical diffusion (based on whether they make use of network hierarchy information or not). Non-hierarchical diffusion performs load propagation based on a discrete spatial model of how heat is diffused in materials in physics or chemistry, and is applicable to non-hierarchical systems such as unstructured peer-to-peer overlays or ad-hoc mobile networks. Hierarchical diffusion relies on a hierarchical network model where the workload propagation is directed through network layers using predetermined routing paths. This diffusion technique is applicable to hierarchical systems such as CDNs or core broadband networks.

With a practical assumption that only some parts of the network can be instrumented and monitored, the basis of the algorithms is that some (data) measurements are available to diffuse, and that workloads primarily propagate among neighboring nodes, where a neighborhood is defined as a cluster of nodes connected to each other or located close to each other. Different algorithms use different neighbourhood definitions and clustering algorithms. Note that the generic terms 'node' and 'network' are used to refer to both application and infrastructure network structures as the algorithms can be used to model and analyze behavior of both applications and infrastructure networks.

4.1 Non-hierarchical Workload Diffusion

The workload diffusion is introduced with three algorithms utilizing different factors in their diffusion processes: the population, the geographical location of network nodes, and the bandwidth capacity of the network links. The key idea of the algorithms is that unless nodes have their own workload measurements they are influenced primarily by their neighboring nodes. The first algorithm (population-based diffusion) performs its diffusion task within a single iteration, while the other execute their tasks in iterations until a predefined maximum number of iterations is reached or a *convergence* state, at which only negligible changes in the diffused workloads of nodes are observed, is obtained.

4.1.1 Population-based Diffusion. This diffusion technique is implemented as in Algorithm 1 which requires the knowledge of the population/number of users $m(\cdot)$ attached to network nodes, and workload measurements $\omega(\cdot)$ of a given set of nodes, denoted by W, as the inputs. Note that $m(\cdot)$ may not be measured in some cases, thus this is checked (if '*undefined*') by the condition in `line 3` of the Algorithm, and the definition of W are used for all the algorithms hereafter. The amount of workload (e.g., application/service requests) arriving to a node is assumed to be proportional to the size of the node population, i.e., the amount of users associated with and/or accessing network services through a specific node. Such a proportion is calculated over the population in a vicinity with a distance of a given radius Δ from that specific node. It is constrained that there must be at least a node having workload measurements in this vicinity; otherwise, Δ should be increased until all nodes are included. Accordingly, the workload of a node is generally calculated using the formula in `line 7`, where the portion of workload diffused to the node is derived as `line 6`. For nodes with unknown population, an average of the population of neighboring nodes in the predefined vicinity is assumed; hence, the amount of workload diffused to such a node is calculated as in `line 5`.

Algorithm 1 Population-based-Diffusion

Input: $G = (V, E)$; Δ; $m(v), \forall v \in V$; $\omega(v), \forall v \in W$
Output: $\omega(x), \forall x \in V$
1: $N_d(v) \leftarrow \{x | x \in V \wedge d(v, x) \leq \Delta\}, \forall v \in V$
2: **for** each node v in V **do**
3: **if** $m(v)$ is undefined **then**
4: $\omega(v) \leftarrow \frac{1}{|N_d(v)|} \times \sum^{\star}_{u \in N_d(v)} [\omega(u)]$
5: **else**
6: $\rho \leftarrow \frac{\sum_{u \in N_d(v)} m(u)}{|N_d(v)|}$
7: $\omega(v) \leftarrow \frac{m(v)}{\rho |N_d(v)|} \times \sum^{\star}_{u \in N_d(v)} [\omega(u)]$
8: **end if**
9: **end for**

4.1.2 Location-based Diffusion. Location-based diffusion performs its workload distribution based on the geographical location of nodes in the network. Algorithm 2 presents the implementation of the diffusion process. The key idea is that the interpolation of workload for a node is accomplished through weighted average influences of neighboring nodes which are selected from a parameterized sized set of nodes located close to the node. Controlling parameters include the maximum size K of the set and the radius Δ defining the neighborhood of the node. More specifically, first, a set of $k \leq K$ closest neighboring nodes of a node v are selected from its neighborhood as shown in `line 1` of the Algorithm. Then, workload propagated to v is the total amount of workloads contributed by these k nodes under a condition that the contributions of neighboring nodes is inversely proportional to their distances (i.e., closely located nodes have higher influence than remote nodes). Such a proportion is derived by formulas listed in `line 2` and `line 3`, and the workload received at a node is calculated as in `line 6`.

Algorithm 2 Location-based-Diffusion

Input: $G = (V, E)$; K; Δ; $\omega(v), \forall v \in W$
Output: $\omega(x), \forall x \in V$
1: $N_k(v) \leftarrow \{x_i \in V, i \in [1, k] \wedge k \leq K \mid$
$d(v, x_1) = \min_{x_j \in V}[d(v, x_j)]$
$\wedge\ d(v, x_{i-1}) \leq d(v, x_i) \leq \Delta, i \in [2, k]\}, \forall v \in V$
2: $r(v, u) \leftarrow \frac{\sum_{x \in N_k(v)} d(v, x)}{d(v, u)}, \forall v \in V, \forall u \in N_k(v)$
3: $\varrho(v, u) \leftarrow \frac{r(v, u)}{\sum_{x \in N_k(v)} r(v, x)}, \forall v \in V, \forall u \in N_k(v)$
4: **repeat**
5: **for** each node v in V **do**
6: $\omega(v) \leftarrow \frac{1}{|N_k(v)|} \times \sum^{\star}_{u \in N_k(v)} [\varrho(v, u) \times \omega(u)]$
7: **end for**
8: **until** *convergence* **or** *max-iterations is reached*

4.1.3 Bandwidth-based Diffusion. As demonstrated in Algorithm 3, the bandwidth-based diffusion is based on bandwidth capacity of the neighboring network links. Particularly, the workload disseminated to a node from one of its neighboring nodes is proportional to the capacity of the link connecting the two nodes. Such a proportion of workload contribution to the node is calculated as in `line 2` of the algorithm, where the set of neighboring nodes of the node is calculated as in `line 1`. The total workload received at a node is an average of the contributions of all its neighboring nodes as shown in `line 5`.

Algorithm 3 Bandwidth-based-Diffusion

Input: $G = (V, E)$; $\omega(v), \forall v \in W$; $B(u, v), \forall (u, v) \in E$
Output: $\omega(x), \forall x \in V$
1: $N(v) \leftarrow \{x | x \in V \wedge (v, x) \in E\}, \forall v \in V$
2: $\delta(v, u) \leftarrow \frac{B(v, u)}{\sum_{x \in N(v)} B(v, x)}, \forall v \in V, \forall u \in N(v)$
3: **repeat**
4: **for** each node v in V **do**
5: $\omega(v) \leftarrow \frac{1}{|N(v)|} \times \sum^{\star}_{u \in N(v)} [\delta(v, u) \times \omega(u)]$
6: **end for**
7: **until** *convergence* **or** *max-iterations is reached*

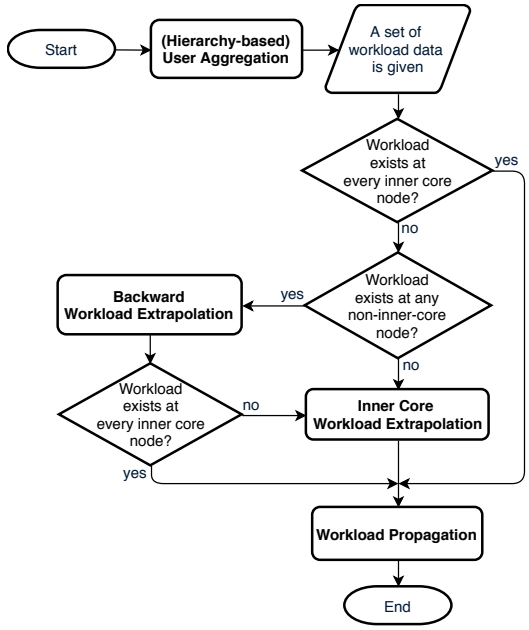
4.2 Hierarchical Workload Diffusion

Hierarchy-based diffusion requires knowledge of the network structure, hierarchy, and links. The network hierarchy implies a directed network flow in the network, i.e., upstream and downstream directions that can be assumed when modeling how network traffic is processed. For this workload diffusion technique, two algorithms are proposed with the flow charts shown in Figure 2. Each algorithm operates in multiple phases which are corresponding to procedures presented in following subsections.

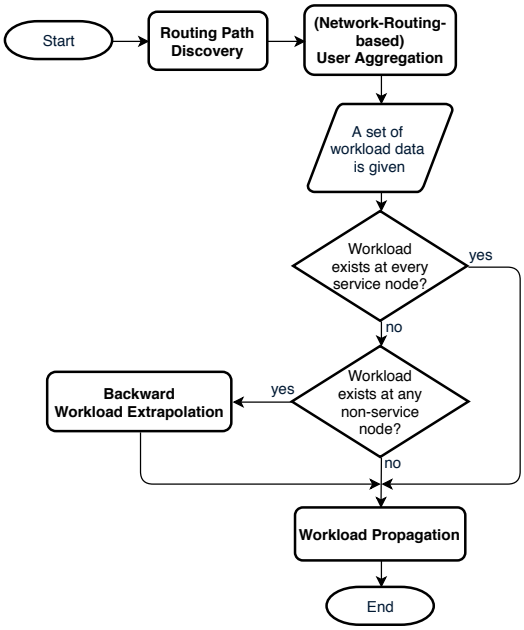
4.2.1 Hierarchy-based Diffusion. As shown in Figure 2a, the hierarchy-based diffusion is composed of four phases executing user aggregation, workload extrapolation, and workload propagation as detailed in Procedures 1 – 4.

The first diffusion phase is to aggregate users attached to all nodes toward the inner-core nodes (see Procedure 1). Due to the homogeneous user behavior, the amount of user requests is proportional to the number of users. The aggregation process is performed layer by layer from the edge (T1) to the network core, which is controlled by two '*for*' loops in `line 8` and `line 10` of the procedure. The total number of users attached to each node is distributed to its lower nodes proportional to the bandwidth capacity of the links connecting corresponding pairs of nodes. The *ratio of user distribution* that a node u contributes to its lower node v is given by the formula in `line 11` and the *aggregated number of users* of v is then calculated as in `line 12`. Both are calculated for every node and returned as the output of the procedure.

The second phase of diffusion is to propagate and extrapolate a workload time series at every node on the way from the nodes with measurements backwards to the core nodes as shown in Procedure 2. The extrapolation is based on *ratios of workload distribution* of the lower nodes to the upper ones. Such a ratio is calculated as in `line 7` of the Procedure. As described, the ratio of workload distribution



(a) Algorithm 4 Hierarchy-based Diffusion



(b) Algorithm 5 Network-Routing-based Diffusion

Figure 2: The flow charts of two hierarchical workload diffusion algorithms.

Procedure 1 Hierarchy-based-User-Aggregation

Input: $G = (V, E)$; $L(v), m(v), \forall v \in V$; $B(u, v), \forall (u, v) \in E$

Output: $\phi(x), \forall x \in V$; $\delta(u, v), \forall v \in V, \forall u \in U(v)$

1: Let l_{max} be the highest level of nodes
2: $S_0 \leftarrow \{x | x \in V \wedge L(x) = 0\}$
3: $S_{l_{max}} \leftarrow \{x | x \in V \wedge L(x) = l_{max}\}$
4: $N(v) \leftarrow \{x | x \in V \wedge (v, x) \in E\}, \forall v \in V$
5: $U(v) \leftarrow \{x | x \in N(v) \wedge L(x) = L(v) + 1\}, \forall v \in V \setminus S_{l_{max}}$
6: $D(u) \leftarrow \{x | x \in N(u) \wedge L(x) = L(u) - 1\}, \forall u \in V \setminus S_0$
7: $\phi(v) \leftarrow m(v), \forall v \in \{x | L(x) = l_{max}\}$
8: **for** each level l from $l_{max} - 1$ downto 0 **do**
9: $S_l \leftarrow \{x | x \in V \wedge L(x) = l\}$
10: **for** each node v in S_l **do**
11: $\delta(u, v) \leftarrow \dfrac{B(u, v)}{\sum_{x \in D(u)} B(u, x)}, \forall u \in U(v)$
12: $\phi(v) \leftarrow m(v) + \sum_{u \in U(v)} \delta(u, v)\phi(u)$
13: **end for**
14: **end for**

Procedure 2 Backward-Workload-Extrapolation

Input: $G = (V, E)$; $L(v), \forall v \in V$; W; $\omega(v), \forall v \in W$; $B(u, v), \forall (u, v) \in E$; $\delta(u, v), \forall v \in V, \forall u \in U(v)$

Output: $\omega(x), \forall x \in V$

1: Calculate l_{max} and $U(v)$ as in **Procedure 1**
2: $X \leftarrow W$
3: **for** each level l from $l_{max} - 1$ downto 0 **do**
4: $S_l \leftarrow \{x | x \in V \wedge L(x) = l\}$
5: $Y \leftarrow \{S_l \setminus X\}$
6: **for** each node v in Y **do**
7: $\gamma(v, u) \leftarrow \dfrac{\delta(u, v)\phi(u)}{\phi(v)}, \forall u \in U(v)$
8: **if** $\{U(v) \cap X\} \neq \varnothing$ **then**
9: $\omega(v) \leftarrow \dfrac{\sum^{\star}_{u \in \{U(v) \cap X\}} [\delta(u, v) \times \omega(u)]}{\sum_{u \in \{U(v) \cap X\}} \gamma(v, u)}$
10: $X \leftarrow X \cup \{v\}$
11: **end if**
12: **end for**
13: **end for**

from a node v to one of its upper nodes u is proportional to the contribution of u to v in terms of the aggregated number of users. Using this ratio combined with workload measurements at upper nodes of node v, the workload time series of v can be extrapolated as shown in `line 9`.

This extrapolation is recursively executed for every lower node of v (if applicable) down to the inner-core nodes. The recursive execution is controlled by two 'for' loops shown in (`line 3` and `line 6`). It is worth noting that the workload of a node v can be extrapolated iff there exists workload data of at least one upper node u of v as illustrated by the condition in `line 8`. This means there is no guarantee for the attainment of extrapolated workload for every node in the network after this phase. Additionally, this backward workload propagation and extrapolation is required only when there exists workload measurements at nodes in upper layers

different from the inner-core layer and there is a lack of measurements at any single inner-core node. If measurements are obtained for every inner-core node, the diffusion algorithm will bypass this extrapolation and continue with next phases (inner-core workload extrapolation and/or workload propagation).

Procedure 3 InnerCore-Workload-Extrapolation

Input: $G = (V, E)$; $L(v), \phi(v), \omega(v), \forall v \in V$
Output: $\omega(x), \forall x \in \{\text{set of inner-core node}\}$
1: Calculate S_0 and $N(v)$ as in **Procedure 1**
2: $X \leftarrow \{x | x \in S_0 \wedge \omega(x) \neq \varnothing\}$
3: $Y \leftarrow S_0 \setminus X$
4: **for** each node v in Y **do**
5: $\omega(v) \leftarrow \dfrac{\sum^{\star}_{u \in \{N(v) \cap X\}} [\phi(u) \times \omega(u)]}{\sum_{u \in \{N(v) \cap X\}} \phi(u)}$
6: **end for**

After backward workload extrapolation, if there still exists an absence of workload data at any inner-core node then the algorithm of inner-core workload extrapolation (see Procedure 3) is invoked. With the measurements and extrapolated workloads of neighboring nodes, workload of an inner-core node is extrapolated based on the weighted sum model [25]. Specifically, workload of v is calculated by the sum of weighted values of all existing workloads of $u \in N(v)$ as shown in `line 5` of the procedure, in which the weight factor for each workload is derived from the aggregated number of users at the node with that given workload. This procedure is expected to complete the missing workload data at the inner-core nodes.

The last phase of the diffusion algorithm, described in Procedure 4, performs a workload propagation which disseminates workloads from inner-core nodes to every network node with no workload measurement. The main flow of the propagation is controlled by two '*for*' loops shown in `line 2` and `line 5`. Workload of a lower node is disseminated to every of its upper nodes based on the ratios of workload distribution previously calculated in Procedure 2, and this is realized by the formula in `line 7` of this Procedure 4.

Procedure 4 Workload-Propagation

Input: $G = (V, E)$; $L(v), \forall v \in V$; W; $B(u, v), \forall (u, v) \in E$;
$\omega(v), \forall v \in \{\text{set of inner-core nodes}\}$
Output: $\omega(x), \forall x \in V$
1: Calculate l_{max} and $D(u)$ as in **Procedure 1**
2: **for** each level l from 1 to l_{max} **do**
3: $S_l \leftarrow \{x | x \in V \wedge L(x) = l\}$
4: $Y \leftarrow \{S_l \setminus W\}$
5: **for** each node v in Y **do**
6: Calculate $\gamma(u, v)$ as in **Procedure 2**
7: $\omega(v) \leftarrow \sum^{\star}_{u \in D(v)} [\gamma(u, v) \times \omega(u)]$
8: **end for**
9: **end for**

4.2.2 Network-Routing-based Diffusion. Figure 2b shows the flow chart of the network-routing-based diffusion which also includes four main phases. Because this diffusion assumes to have knowledge of all routing paths from every node to the service nodes, it is required a routing path discovery before performing user aggregation. Therefore, the first phase of this diffusion is implemented by employing Dijkstra's algorithm to find shortest routing paths. Note that there can be multiple (shortest) paths (with the same hop-distance) from a node to a service node. The phase of user aggregation applies the same assumption of user behavior as in Procedure 1 and also takes the link capacities into account. The key difference is that this aggregation is ended up at the service nodes through a set of predetermined routing paths. This means each service node can obtain an aggregated number of users from the contributions of all its upper nodes; hence, the phase of inner-core workload extrapolation is unnecessary. As a result, the diffusion continues with the phase of backward workload extrapolation and finishes with the phase of workload propagation. These two phases are implemented by Procedure 2 and 4, which are also adopted by the hierarchy-based diffusion (Algorithm 4), but using the set of service nodes (instead of the entire set of inner-core nodes) as the destinations and the sources for the workload extrapolation and propagation, respectively.

Procedure 5 Network-Routing-based-User-Aggregation

Input: $G = (V, E)$; $L(v), m(v), \mathcal{P}(v), \forall v \in V$
Output: $\phi(x), \forall x \in V$; $\delta(u, v), \forall v \in V, \forall u \in U(v)$
1: Calculate S_0, $S_{l_{max}}$, $U(v)$, and $D(u)$ as in **Procedure 1**
2: $\phi(v) \leftarrow m(v), \forall v \in V$
3: **for** each node r in $\{V \setminus S_0\}$ **do**
4: $P \leftarrow \mathcal{P}(r)$
5: $X \leftarrow \{r\}$
6: $\phi^P(r) \leftarrow m(r)$
7: **repeat**
8: $X' \leftarrow \varnothing$
9: **for** each node u in X **do**
10: $D^P(u) \leftarrow \{x | x \in D(u) \wedge \exists p \in P : (x, u) \in p\}$
11: **for** each node v in $D^P(u)$ **do**
12: $\alpha^P(u, v) \leftarrow |\{p \in P \mid (u, v) \in p\}|$
13: $\delta^P(u, v) \leftarrow \dfrac{\alpha^P(u, v) B(u, v)}{\sum_{x \in D^P(u)} \alpha^P(u, x) B(u, x)}$
14: $\phi^P(v) \leftarrow \delta^P(u, v) \phi^P(u)$
15: $\psi(u, v) \leftarrow \psi(u, v) + \phi^P(v)$
16: $\phi(v) \leftarrow \phi(v) + \phi^P(v)$
17: **end for**
18: $X' \leftarrow X' \cup D^P(u)$
19: **end for**
20: $X \leftarrow X'$
21: **until** $X = \varnothing$
22: **end for**
23: $\delta(u, v) \leftarrow \dfrac{\psi(u, v)}{\phi(u)}, \forall v \in \{V \setminus S_{l_{max}}\}, \forall u \in U(v)$

The algorithm of user aggregation in the second phase is depicted in Procedure 5. With the set of all routing paths produced by the first phase, it is possible to assume that each node is associated with a directed acyclic graph (DAG) rooted at the node, which is constructed from a set of routing paths starting at the node and ending at one of the service nodes. The algorithm then distributes users attached to the root node down to the service nodes along the given DAG. An intermediate node in the DAG aggregates users distributed by its upper nodes and continue distributing the users to its lower ones. With a given DAG, the partial aggregated number of users or the number of attached users at a node u is distributed to its lower node v proportionally to the utility of the directed link from u to v, which is defined by a product of the number of paths going from u to v and the capacity of link (u, v). In fact, such a *partial distribution ratio* $\delta^P(u, v)$ is calculated by the ratio of the utility of link (u, v) over a sum of the utility of all links going out from u as given in `line 13`. Using this partial distribution ratio, the *partial number of users* distributed to node v (by node u) $\phi^P(v)$ is calculated as in `line 14`. And the *total aggregated number of users* $\phi(v)$ and the total contribution (in terms of the number of users) $\psi(u, v)$ of u to v are updated with an addition of $\phi^P(v)$ as illustrated in `line 16` and `line 15`. Lastly, the *total ratio of user distribution* that a node u contributes to its lower node v is calculated as in `line 23`. The output of this procedure is the same as that of Procedure 1.

5 EXPERIMENTS AND VALIDATION

5.1 Settings for Experiments

Two experiments are carried out using an artificial network model developed for the city of Umeå, Sweden. The network model has been developed in the RECAP project [17] in collaboration with British Telecom (BT) and is designed to retain the key characteristics of the BT core network but at a smaller scale. The geographical distribution of network nodes is constructed based on census population data of the city (provided by the municipality of Umeå), which is also utilized for making assumptions on the utilization of network links and their bandwidth capacities. The network consists of 45 nodes, among which there are 3 inner-core, 6 outer-core, 9 metro and 27 T1 nodes. Nodes are named based on their type (network hierarchy position), i.e. access tier, metro, outer-core and inner-core are named 'Txx', 'Mx', 'Ox' and 'Ix', respectively (where 'x' represents an identifying index). For redundancy and fault tolerance, every node is connected to at least two lower tier nodes using separate links. Moreover, an outer-core node may be connected to two of its neighboring outer-core nodes. Each inner-core node has connections to the other two inner-core nodes. For each layer, the same bandwidth capacity is assigned to every inner-layer link (if it exists) connecting two nodes located in the layer. Inner-layer links in inner-core layer are assigned the highest capacity. All inner-layer links in outer-core layer and cross-layer links lying between the access tier layer to outer-core layer or between metro layer and outer-core layer or between outer-core layer to inner-core layer are associated with the same capacity. Cross-layer links lying between the access tier layer and the metro layer are assigned the lowest capacity. Besides being motivated by the BT 21CN, all these network settings are adopted to demonstrate the generality of the proposed algorithms, i.e., they are applicable to different types of network. Figure 3 illustrates the network topology of the city together with the dispersion of population throughout the network.

Workload datasets used for experiments are provided by BT through instrumentation of their production CDN system [8]. Data is provided as time series representing the traffic (i.e., the workload) served by three core cache servers in the system. Time series are collected at an interval of 20 minutes for nine consecutive months between 2016 and 2017, and normalized for security reasons. The normalized data is called *proportional traffic* which is used as the label of the y-axis while the x-axis represents the time in the following illustrations of workload.

We have carried out experiments of workload generation in two scenarios. In the first scenario measurements are made at central nodes in the inner core of the network and artificial data is diffused in a single direction out towards the edge of the network. This behavior is representative of how many systems are instrumented during development and illustrates the basic features and operations of the proposed algorithms. In the second scenario, data traces are associated to three random nodes at different network layers and artificial data is diffused in all directions throughout the network. This scenario is representative of ad hoc and unstructured networks and is used to comprehensively verify the functionality and illustrate the flexibility of the algorithms. To further support verification, we also provide reasoning on the affinity and anti-affinity of workload patterns together with a validation on statistical properties of diffused workload traces. Experimental results and analysis are revealed in the following subsections. Note that to highlight specific aspects of algorithmic behavior, workload traces from different nodes are presented in the illustrations of the two experiments below.

5.2 Experimental Scenario 1

As described above, in this scenario real workload traces collected from core caches of the BT CDN are associated to three core nodes ($I1$, $I2$ and $I3$) in the experimental network model (a setting mimicking the real world situation). Figure 4 shows the shape of the given workload data. The five proposed algorithms are executed to diffuse the workload to every node in the network and results are collected as workload data traces for all nodes. Due to paper space restrictions, only a subset of traces at selected nodes is presented and discussed in the following subsections.

5.2.1 Non-hierarchical Workload Diffusion. Figure 5 shows the results obtained by population-based diffusion for node $T21$ and $T22$ and the difference between them. It is apparent that the traffic retrieved for $T22$ is much higher the one of $T21$ although the traffic data are both pulled from the two lower nodes ($M2$ and $M3$). The reason for this is that the diffusion algorithm propagates the traffic proportional to the user population of the nodes, and the population of $T22$ is higher than that of $T21$ as observed in Figure 3. Note that the presented traffic data are time series with a 20-minute sampling interval, but for the sake of clarity, the difference (in red) is shown as a series of the rolling mean values calculated with 1-week frequency. The same approach is applied for the following comparisons.

Figure 6 presents the results obtained by location-based diffusion for node $T21$ and $T22$. As the location-based algorithm models the influence of neighboring nodes proportional to the relative distance

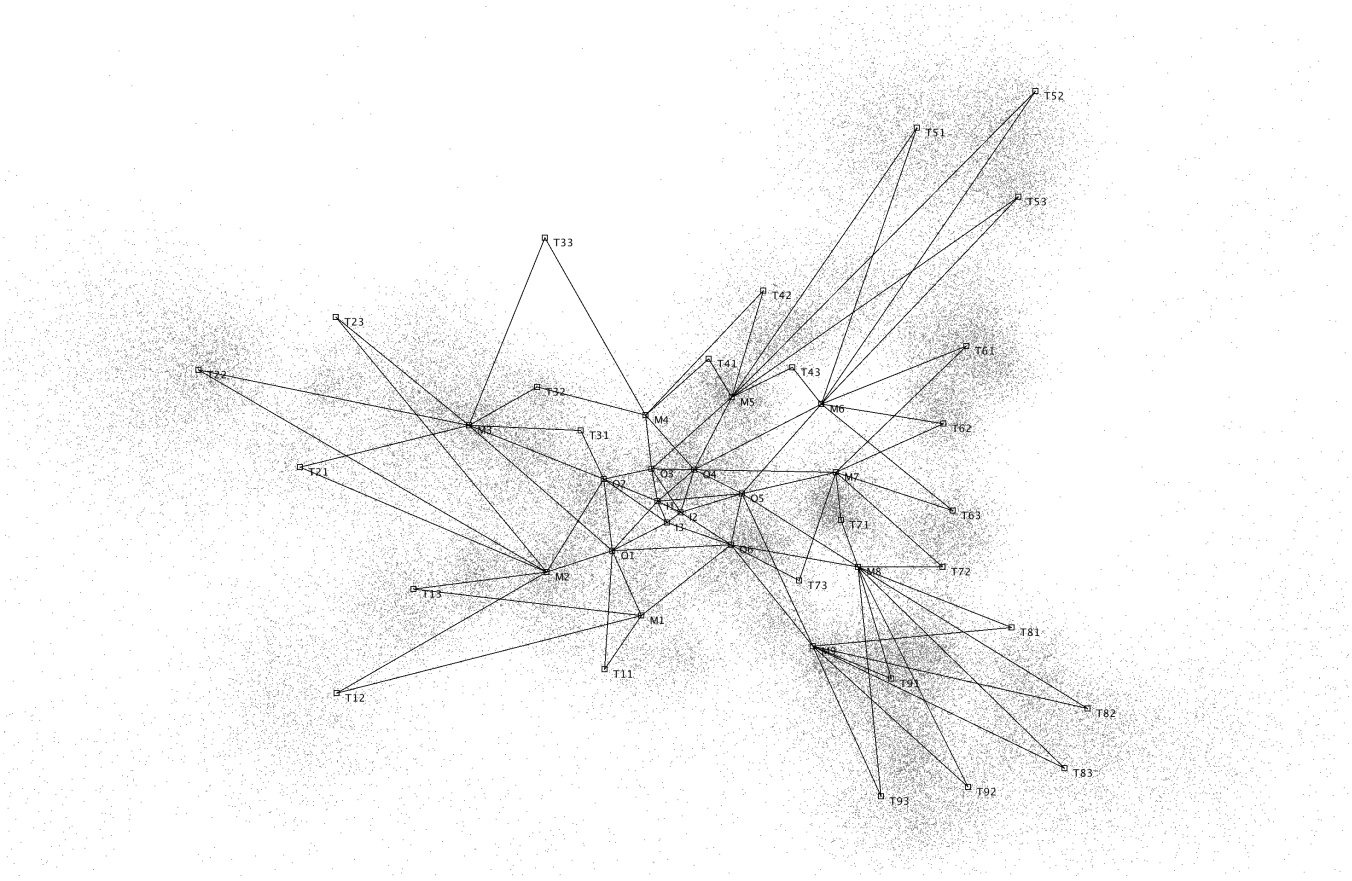
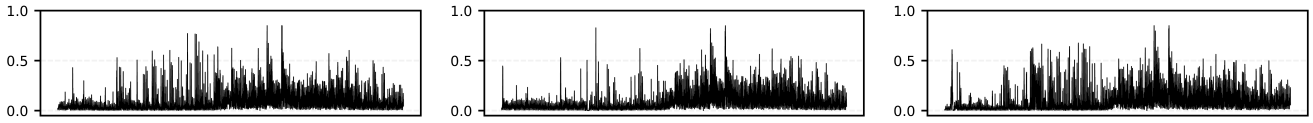


Figure 3: Network and population models for the city of Umeå, Sweden.



(a) Workload of $I1$ **(b) Workload of $I2$** **(c) Workload of $I3$**

Figure 4: Original workload measurements at nodes $I1$, $I2$ and $I3$.

between the nodes (which in this case is small), the node loads are very similar for $T21$ and $T22$.

Identical workload trace are obtained through bandwidth-based diffusion for the two nodes $T21$ and $T22$ as illustrated in Figure 7. The similarity of the traces is due to the diffusion algorithm distributing traffic from two lower nodes $M2$ and $M3$ to $T21$ and $T22$ proportional to their relative bandwidth capacity, which in this case (as described in section 5.1) are the same - i.e. the bandwidth capacities of the links connecting $T21$ and $T22$ to their lower level nodes are symmetric. The red line in the chart shows no difference between the two obtained workload traces.

5.2.2 Hierarchical Workload Diffusion. Figure 8 illustrates the workload traces generated for node $M3$ by two different hierarchical diffusion algorithms: hierarchy-based and network-routing-based. As described in section 4.2, the workload distributed to nodes (from their connecting lower level nodes) by these two algorithms depends on multiple factors including the population of the node, the bandwidth of the links connecting the nodes, and the routing paths conveying the propagated workloads.

The difference factor between the performance of the two algorithms in the experiment is the traffic-propagating paths. As observed from Figure 3, in hierarchy-based diffusion node $M3$ serves a set of nodes {$T21$, $T22$, $T23$, $T31$, $T32$, $T33$}, which means that the traffic collected at $M3$ will be distributed to neighboring nodes proportional to their corresponding populations and the capacity of the links connecting them and $M3$. Note however that when using network-routing-based diffusion, node $T31$ is excluded from

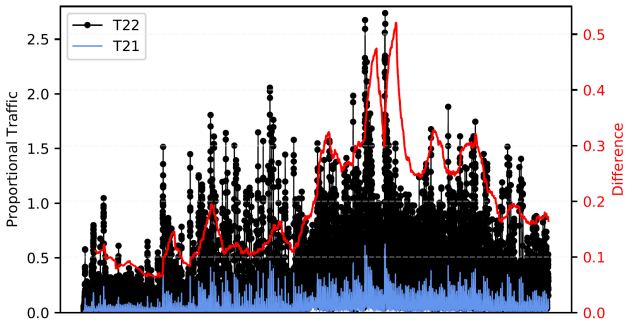


Figure 5: A comparison of workloads generated by population-based diffusion for nodes $T21$ and $T22$.

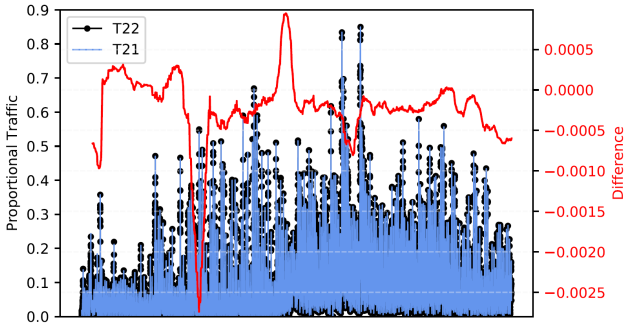


Figure 6: A comparison of workloads generated by location-based diffusion for nodes $T21$ and $T22$.

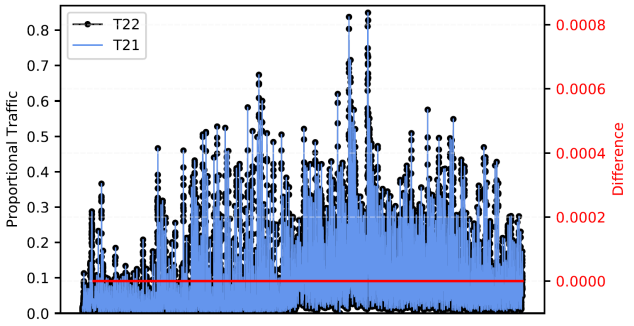


Figure 7: A comparison of workloads generated by bandwidth-based diffusion for nodes $T21$ and $T22$.

the set. This is because of the utilization of shortest paths routing algorithm in traffic propagation, which distributes $O2$ traffic to $T31$ only through the link ($O2$-$T31$) (i.e. never through the propagation path ($O2$-$M3$-$T31$) / via $M3$). This results in the traffic collected at $M3$ in hierarchy-based diffusion being higher than that of network-routing-based diffusion as shown in the figure. Due to the traffic being normalized and the population of $T31$ is low, the traffic propagated through the link ($M3$-$T31$) in hierarchy-based diffusion is

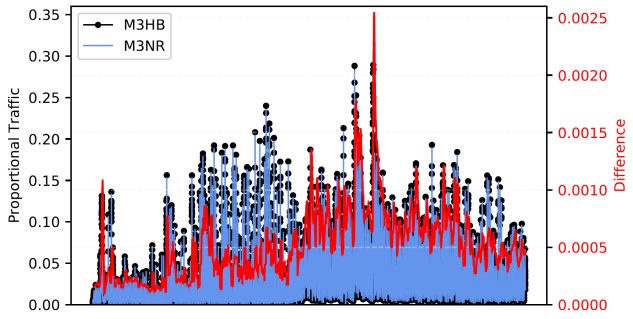


Figure 8: A comparison of workloads generated for node $M3$ by hierarchy-based and network-routing-based diffusion. HB and NR represent *hierarchy-based diffusion* and *network-routing-based diffusion*, respectively.

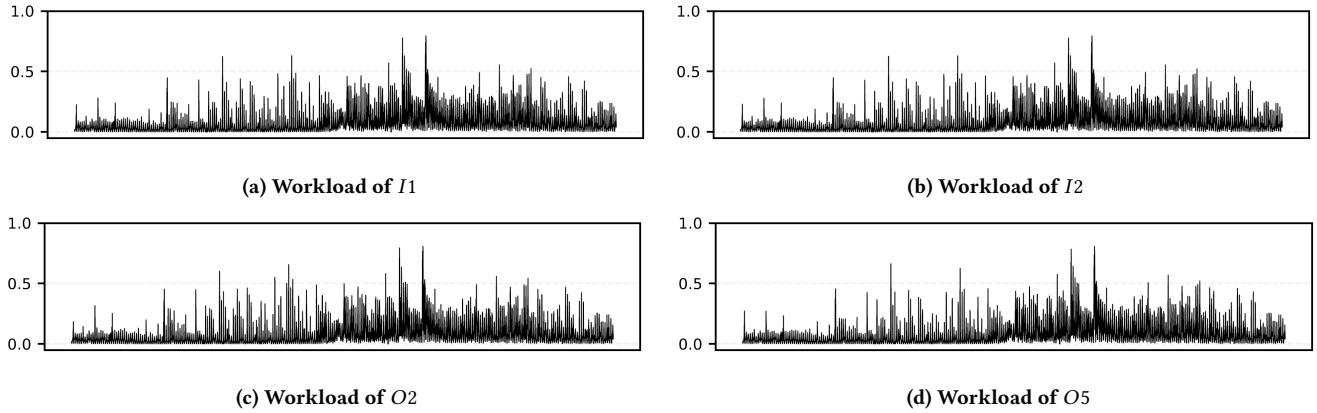
quite small, and the difference between the two obtained workload traces is relatively insignificant.

5.3 Experimental Scenario 2

To illustrate the flexibility of the approach, we also construct a scenario where the real workload traces are associated to non-core nodes ($M2$, $M9$ and $T62$), and execute the proposed algorithms to diffuse workloads in this setting. As for scenario 1, only a selected subset of the diffused workloads is presented and analyzed.

5.3.1 Non-hierarchical Workload Diffusion. Figure 9 illustrates the results obtained by the bandwidth-based diffusion (Algorithm 3). In this case, workload time series generated for two inner-core nodes $\{I1, I2\}$ and two outer-core nodes $\{O2, O5\}$ are presented. According to the algorithm, the extrapolated workloads of nodes are influenced by the workloads of its neighbors at rates proportional to the capacity of the connecting links. As the core of the network is connected with symmetric links, extrapolated workloads at the selected nodes are of almost the same scale, and similar workload patterns are obtained for all inner-core nodes as illustrated in Figure 9a and 9b. The shape of workload of $O2$ is slightly different from that of $O5$ as they are influenced correspondingly by the original workloads given at $M2$ and $M9$ with different shapes (note that other influencers of $O2$ and $O5$ just received extrapolated workloads with smoothed patterns). Workload of $T91$ is derived from only two connecting nodes ($M8$ and $M9$). With the same reasoning above, the workload pattern, evolved over symmetric links, of $T91$ is nearly matched with that of $M9$, but different from that of others with extrapolated workloads (e.g., node $O2$) as illustrated in Figure 11.

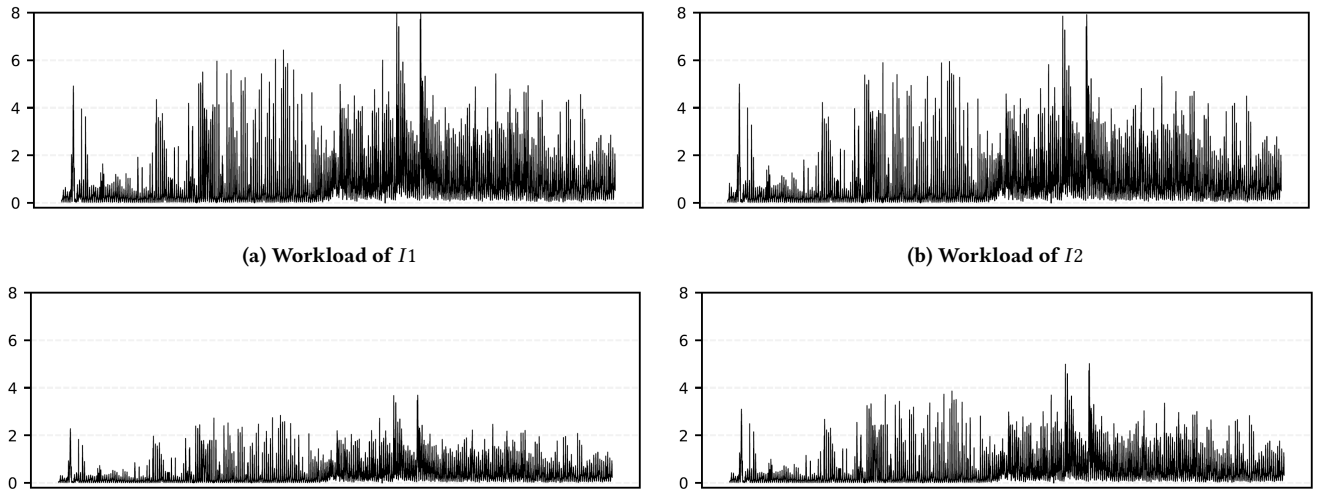
5.3.2 Hierarchical Workload Diffusion. Figure 10 illustrates the results obtained by the hierarchy-based diffusion (Algorithm 4), which include the generated workloads of two inner-core nodes $\{I1, I2\}$ and two outer-core nodes $\{O4, O5\}$. In case of the hierarchical workload diffusion, according to Procedure 2, workload measured at a higher tier node leads to a complete workload extrapolated at its connecting lower tier nodes, and this recursively causes influences (the increase of workload) toward the core nodes. The extrapolated workload of a node is proportional to the size of



(a) Workload of $I1$ **(b) Workload of $I2$**

(c) Workload of $O2$ **(d) Workload of $O5$**

Figure 9: Workloads at nodes $I1$, $I2$, $O2$ and $O5$, generated by bandwidth-based diffusion (Algorithm 3).



(a) Workload of $I1$ **(b) Workload of $I2$**

(c) Workload of $O4$ **(d) Workload of $O5$**

Figure 10: Workloads at nodes $I1$, $I2$, $O4$ and $O5$, generated by hierarchy-based diffusion (Algorithm 4).

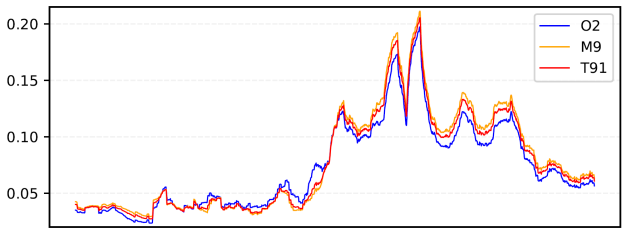


Figure 11: Workload patterns of nodes $O2$, $M9$ and $T91$.

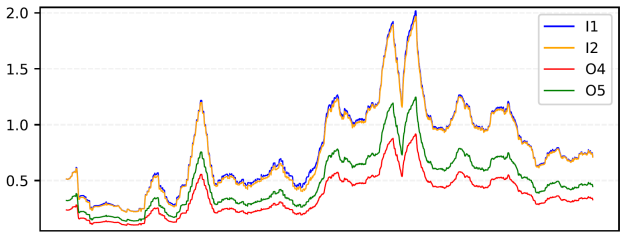


Figure 12: Workload patterns of nodes $I1$, $I2$, $O4$ and $O5$.

population served by the node. Note that the population served by a node is an aggregated population of all its higher tier nodes, and such an aggregated population is obtained by the Procedures (1 and 5) of user aggregation. This implies that the extrapolated workload at a lower tier node will be higher than the workload of any of its connecting higher tier nodes. Based on this, with the workload measurement at $T62$ it is possible to extrapolate complete workloads at $\{M6, M7\}$ (serving all their higher tier nodes). Such

backward extrapolation is recurred down to lower tier nodes which are $\{O4, O5\}$ and finally $\{I1, I2\}$. Among three given workloads, the influence of $T62$ (the lowest tier / inner-most node) on the workloads of nodes $I1$ and $I2$ dominates that of the others. Thus, workload patterns of these two nodes are nearly the same as shown in Figure 12. Additionally, the workloads of these nodes are higher than the workload of their connected higher tier nodes $\{O4, O5\}$.

After the workload extrapolation processes, workloads of all inner-core nodes are available to be distributed to all nodes in the network by Procedure 4. It is apparent that workloads received at nodes $\{O4, O5\}$ are propagated only from nodes $\{I1, I2\}$; hence, the workload patterns of both $O4$ and $O5$ follow the same pattern of $I1$ or $I2$ (see Figure 12), and the extrapolated workload of $O5$ is higher than that of $O4$ (illustrated by Figure 10c and 10d). The reason behind is that the workload of $O5$ serves the highest population nodes of the network (see Figure 3), including the four metro nodes $\{M6, M7, M8, M9\}$ and a large number of T1 nodes including $T5$x, $T6$x, $T7$x and $T8$x.

5.4 Validation

Due to the absence of real measurements for the diffused nodes, it is not possible to directly assess the similarity of diffused workloads and corresponding actual measurements. Instead, we use statistical techniques to evaluate the performance of the less intuitive bandwidth-based and hierarchy-based diffusion algorithms (Algorithm 3 and 4, respectively) as follows:

- The original measurements of nodes $M2$, $M9$ and $T62$ (in the following, referred to as ground truth data) are used for data diffusion using the bandwidth-based and hierarchy-based algorithms. This newly synthesized data is further used to once again diffuse measurements, this time for nodes $M2$, $M9$ and $T62$ (in the following, referred to as diffused data). Hence, using this transitivity diffusion approach, data for the baseline nodes $M2$, $M9$ and $T62$ is created, rendering a ground truth to diffused data comparison possible.
- The *entropy* [22] and *approximate entropy* [18] of the ground truth and diffused workload data is evaluated and analyzed. This provides a comparison of the ground truth and diffused data in terms of added or lost fluctuations and noise in the diffused data.
- The correlation between the ground truth and the diffused data is calculated. This provided a metric for the dependency and association of the diffused data to the ground truth data. This ideally should be both high and positive.

Hereby, for the performed validation, it is worth reiterating that the purpose of the so-called data rediffusion is not to synthesize data identical to the ground truth. Rather, the aim is to diffuse workload measurements reflecting the original ground truth data in its statistical properties, and by extension, proving the applicability of the presented algorithms.

The results of both the entropy and approximate entropy calculation are demonstrated in Table 1. An interpretation and explanation of the shown numbers is in order here. The approximate entropy for the entire dataset shows only a minor deviation for both algorithms in terms of fluctuation of the data (the highest difference being 0.03). This means that the diffused data retains its properties in terms of predictability, which in turn shows that any forecasting application on the diffused is not negated by the presented algorithms. As for the entropy, the bandwidth-based algorithm (Algorithm 3) is observed to succeed in retaining the supposed information properties without introducing any additional noise to the data. This, in terms, is different for the hierarchy-based algorithm, which as seen in the minor approximate entropy increase and the larger entropy increase, introduces noise into the data. As previously stated, this does not harm the forecastability of the data. This behavior can be attributed to the fact that the hierarchy-based diffusion algorithm (Algorithm 4), when diffusing, aggregates the workload over all upper nodes (at the lower one); hence, any noise present in the input measurements is bound to be reflected and to an extent amplified in the diffused data.

Lastly, Figure 13 shows a high correlation between the ground truth and diffused workload data of node $M2$. The scatterplots also show that the data following the same trend and being clustered densely together. This again validates the ability of the algorithms to diffuse the workload data while retaining its core properties.

6 CONCLUSION

This paper addresses the problem of workload generation and proposes a framework with five algorithms to extrapolate and generate workload data for network nodes using diffusion of measurements from a subset of nodes. Each algorithm uses different types of data and knowledge of the network, and employs different techniques to perform its diffusion, and is thus applicable to different situations and types of network. The performance of the algorithms is evaluated using workload data from a production CDN. Results are presented, analyzed, and validated using statistical techniques.

Besides the main target of workload generation to support large-scale distributed application profiling, the proposed framework can also be used to mitigate data privacy concerns in dissemination of data traces collected from sensitive data applications, for example through combinations of data privacy filters and publications of subsets of (anonymized) monitored data and generated artificial data (that can be shown to retain the key statistical properties of the original data without disclosing sensitive information). With the given subset of traces and a network/application graph model, the diffusion algorithms can be tuned to generate workloads for all application components/network nodes as needed. Future work includes further development of application models for service function chains and IoT applications, and adaptation of the diffusion algorithms to such applications for a comprehensive validation of the framework in multiple settings (workload generation and data privacy evaluations in simulation-based emulators, testbeds, and production environments). Standardization and abstraction of the models as well as the proposed algorithms to accomplish the entire framework are also underway.

ACKNOWLEDGMENTS

This work received funding from the European Union's H2020 research and innovation programme under Grant Agreement No. 732667 (RECAP). The authors express their gratitude to Peter Willis and his team at BT who provided valuable insights and data, and to TietoEVRY for their support in the finalization of the work.

Table 1: Entropy and approximate entropy measurements for the rediffused data of nodes $M2$, $M9$ and $T62$. Hereby the values $M2$, $M9$ and $T62$ correspond to the original measurements, whereas HB and BW stands for hierarchy-based and bandwidth-based respectively.

	$M2$	$M2HB$	$M2BW$	$M9$	$M9HB$	$M9BW$	$T62$	$T62HB$	$T62BW$
Entropy	5.6252	6.7778	5.6368	5.7002	7.0710	5.6491	5.7588	6.8891	5.6464
Approximate Entropy	0.6017	0.6344	0.6236	0.5972	0.6245	0.6202	0.6179	0.6257	0.6236

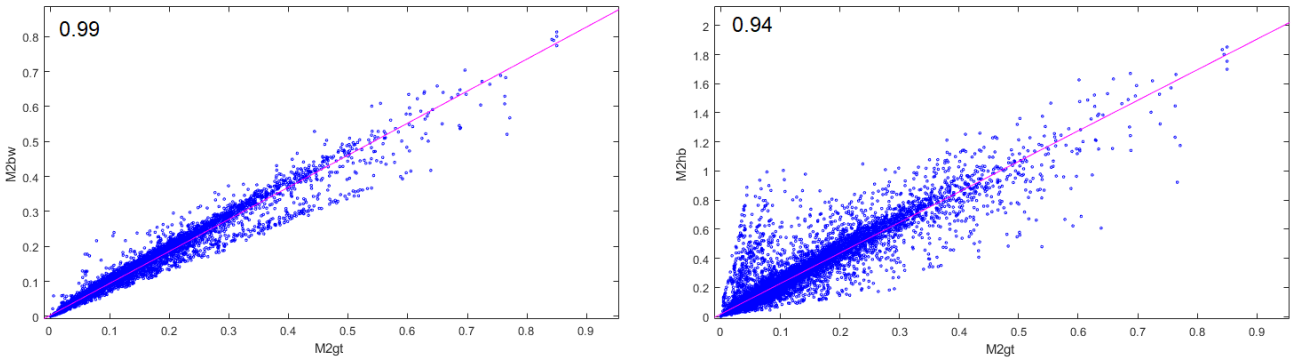


Figure 13: The distribution of the rediffused values and the original measurements for node $M2$. In the left upper corner is the correlation coefficient between the original measurement ($M2gt$) and the rediffusion using bandwidth-based ($M2bw$) or hierarchy-based ($M2hb$) algorithm. The second linear relationship observed in the $M2bw$ plot (below the red line) is explained by the nature of the algorithm described in 4.1.3 that workload of adjacent nodes is averaged for the total workload of a node.

REFERENCES

[1] M. Awad and D. A. Menascé. 2015. Automatic Workload Characterization Using System Log Analysis. In *Proc. Computer Measurement Group Conference on Performance and Capacity.*

[2] P. J. Brockwell and R. A. Davis. 2002. *Introduction to Time Series and Forecasting* (2 ed.). Springer, New York. ISBN: 978-0-387-95351-9.

[3] M. C. Calzarossa, L. Massari, and D. Tessera. 2016. Workload Characterization: A Survey Revisited. *ACM Computing Surveys (CSUR)* 48, 3 (2016), 48.

[4] C. D. Carothers, J. S. Meredith, M. P. Blanco, J. S. Vetter, M. Mubarak, J. LaPre, and S. Moore. 2017. Durango: Scalable Synthetic Workload Generation for Extreme-Scale Application Performance Modeling and Simulation. In *Proc. ACM SIGSIM Conference on Principles of Advanced Discrete Simulation.* 97–108.

[5] J. Chen and R. M. Clapp. 2017. Astro: Auto-Generation of Synthetic Traces Using Scaling Pattern Recognition for MPI Workloads. *IEEE Transactions on Parallel and Distributed Systems* 28, 8 (August 2017), 2159–2171.

[6] M. Curiel and A. Pont. 2018. Workload Generators for Web-Based Systems: Characteristics, Current Status, and Challenges. *IEEE Communications Surveys Tutorials* 20, 2 (Second quarter 2018), 1526–1546.

[7] A. Cuzzocrea, E. Mumolo, and G. Vercelli. 2017. Ergodic Hidden Markov Models for Workload Characterization Problems. In *Proc. International DMS Conference on Visual Languages and Sentient Systems (DMSVLSS).*

[8] M. Leznik et al. 2019. *RECAP Artificial Data Traces.* https://doi.org/10.5281/zenodo.3458559

[9] C. H. G. Ferreira, J. C. Estrella, L. H. Nunes, L. H. V. Nakamura, R. M. Libardi, B. G. Batista, M. L. Peixoto, D. M. Leite, and S. Reiff-Marganiec. 2017. A Low Cost Workload Generation Approach Through the Cloud for Capacity Planning in Service-oriented Systems. In *Proc. ACM International Conference on Internet of Things, Data and Cloud Computing (ICC).* Article 6, 8 pages.

[10] J. He and J. Rexford. 2008. Toward Internet-wide Multipath Routing. *IEEE Network* 22, 2 (March 2008), 16–21.

[11] N. Herbst, A. Amin, A. Andrzejak, L. Grunske, S. Kounev, O. J. Mengshoel, and P. Sundararajan. 2017. Online Workload Forecasting. In *Self-Aware Computing Systems.* Springer, 529–553.

[12] Kitz. 2009. *BT 21CN – Network Topology & Technology.* https://kitz.co.uk/adsl/21cn_network.htm Accessed: February 19, 2020.

[13] T. Le Duc, R. Garcia Leiva, P. Casari, and P-O. Östberg. 2019. Machine Learning Methods for Reliable Resource Provisioning in Edge-Cloud Computing: A Survey. *ACM Computing Surveys* 52, 5, Article 94 (August 2019), 39 pages.

[14] H. Li and R. Buyya. 2007. Model-Driven Simulation of Grid Scheduling Strategies. In *Proc. IEEE International Conference on e-Science and Grid Computing (e-Science 2007).* 287–294.

[15] G. Maddodi, S. Jansen, and R. de Jong. 2018. Generating Workload for ERP Applications Through End-User Organization Categorization Using High Level BusinessOperation Data. In *Proc. ACM/SPEC International Conference on Performance Engineering.* 200–210.

[16] N. Michael, N. Ramannavar, Y. Shen, S. Patil, and J.-L. Sung. 2017. CloudPerf: A Performance Test Framework for Distributed and Dynamic Multi-Tenant Environments. In *Proc. ACM/SPEC International Conference on Performance Engineering.* 189–200.

[17] P-O. Östberg et al. 2017. Reliable Capacity Provisioning for Distributed Cloud/Edge/Fog Computing Applications. In *Proc. European Conference on Networks and Communications (EuCNC).* Oulu, Finland, 1–6.

[18] S. Pincus. 1995. Approximate Entropy (ApEn) as a Complexity Measure. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 5, 1 (1995), 110–117.

[19] D. B. Prats, J. L. Berral, and D. Carrera. 2018. Automatic Generation of Workload Profiles Using Unsupervised Learning Pipelines. *IEEE Transactions on Network and Service Management* 15, 1 (March 2018), 142–155.

[20] G. Rodrigo, E. Elmroth, P-O. Östberg, and L. Ramakrishnan. 2017. Enabling Workflow-aware Scheduling on HPC Systems. In *Proc. ACM International Symposium on High-Performance Parallel and Distributed Computing.* 3–14.

[21] K. Srinivasan, T. Bisson, G. Goodson, and K. Voruganti. 2012. iDedup: Latency-aware, Inline Data Deduplication for Primary Storage. In *Proc. USENIX Conference on File and Storage Technologies.* 24–24.

[22] T. J. Ulrych and R. W. Clayton. 1976. Time Series Modelling and Maximum Entropy. *Physics of the Earth and Planetary Interiors* 12, 2 (1976), 188 – 200.

[23] G. M. Wamba, Y. Li, A. Orgerie, N. Beldiceanu, and J. Menaud. 2017. Cloud Workload Prediction and Generation Models. In *Proc. IEEE International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD).* 89–96.

[24] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu. 2015. BURSE: A Bursty and Self-Similar Workload Generator for Cloud Computing. *IEEE Transactions on Parallel and Distributed Systems* 26, 3 (March 2015), 668–680.

[25] S. H. Zanakis, A. Solomon, N. Wishart, and S. Dublish. 1998. Multi-attribute Decision Making: A Simulation Comparison of Select Methods. *European Journal of Operational Research* 107, 3 (1998), 507 – 529.

[26] K. Zoumpatianos, Y. Lou, I. Ileana, T. Palpanas, and J. Gehrke. 2018. Generating data series query workloads. *The VLDB Journal* 27, 6 (December 2018), 823–846.