Edge Data Repositories - The design of a store-process-send system at the Edge

Adrian-Cristian Nicolaescu
University College London
a.nicolaescu@ee.ucl.ac.uk

Onur Ascigil
University College London
o.ascigil@ucl.ac.uk

Ioannis Psaras
University College London
i.psaras@ucl.ac.uk

ABSTRACT

The Edge of the Internet is currently accommodating large numbers of devices and these numbers will dramatically increase with the advancement of technology. Edge devices and their associated service bandwidth requirements are predicted to become a major problem in the near future. As a result, the popularity of data management, analysis and processing at the edges is also increasing. This paper proposes Edge Data Repositories and their performance analysis. In this context, provide a service quality and resource allocation feedback algorithm for the processing and storage capabilities of Edge Data Repositories. A suitable simulation environment was created for this system, with the help of the ONE Simulator. The simulations were further used to evaluate the Edge Data Repository cluster within different scenarios, providing a range of service models. From there, with the help and adaptation of a few basic networks management concepts, the feedback algorithm was developed. As an initial step, we assess and provide measurable performance feedback for the most essential parts of our envisioned system: network metrics and service and resource status, through this algorithm.

CCS CONCEPTS

• **Networks → Network simulations**; **Network performance analysis**; **In-network processing**; **Network management**; *Network measurement*; *Mobile networks*; *Wireless local area networks.*

KEYWORDS

Edge storage, Edge Computing, Data Repositories, IoT, V2I, Service Provisioning

ACM Reference Format:
Adrian-Cristian Nicolaescu, Onur Ascigil, and Ioannis Psaras. 2019. Edge Data Repositories - The design of a store-process-send system at the Edge. In *1st ACM CoNEXT Workshop on Emerging in-Network Computing Paradigms (ENCP '19), December 9, 2019, Orlando, FL, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3359993.3366644

1 INTRODUCTION

Edge computing is continuously gaining support by creating opportunities and collaborations [14], as it is gaining interest from the research community. According to several studies [14, 15, 19] and predictions for the following years (2022-24) [3, 5], the problem of upstream bottlenecks for Edge ISPs will also be exacerbated by the need for extra services closer to the Edge[12]. This requirement will be a result of increased network traffic created by connected cars, homes, other Internet of Things (IoT) implementations (e.g. smart cities)[18] and/or augmented/virtual reality (AR/VR) devices[4]. Networks are dispersing, hence resource redistribution is required for a smooth transition towards decentralised or even distributed networks.

Important service provision requirements are set by Edge data generated through peak-hour commuting and Smart City applications. Existing work [14, 15, 19] suggests that user, IoT and vehicular data will overload the network with information in the next years. These designs consider the reverse data flow, but imply very small data quantities. In that regard, another paper [15] considers a few blue-sky approaches to the grave need of data (re)distribution in the upstream flow. Data from the domains mentioned above could, thus, be more accessible and usable at points closer to Edge networks, for efficiency and latency purposes.

"Mobile Data Repositories at the Edge" [13] presents a new approach to in-network data distribution, to improve connectivity and reduce upstream stress, by using persistent, Network-layer storage [20]. The *Proactive Strategy*, proposed by this system, relies on data packets to tell the Data Repository systems how they should be treated, through MetaData (MetaInfo) tags. Adding to this, we introduce the store-process-send concept, which was developed to integrate processing with storage, to increase data delivery efficiency of data originating at the Edge. Thus, we now define two main tags, useful throughout the rest of the paper. The shelf-life is the minimum storage period for storage service provision and the maximum processing window for processing service provision with lower constraints. The freshness period represents a deadline for a time-sensitive data processing service. These help in Data management decisions such as storage time, processing needs or storage/deletion after processing. Considering these concepts, we introduce nodes specialised in data storage and processing, called Edge Data Repositories (EDRs), managed by Edge Repository Provider (ERP) servers (briefly introduced in this paper). This work represents a first step in solving an open question of managing EDR environments [20].

The main challenges associated with these systems are: the lack of appropriate feedback and management protocols and algorithms needed for the development of in-network computing, the unstable nature of connections between (mobile) IoT devices and EDRs and the considerable flows of data that such a system has to deal with, among others that are yet to be considered or discovered. Considering these challenges, this paper provides the following contributions as (partial or full) solutions:

ENCP '19, December 9, 2019, Orlando, FL, USA

ACM ISBN 978-1-4503-7000-4/19/12…$15.00
https://doi.org/10.1145/3359993.3366644

- structure of the underlying system, for which the evaluation and an algorithm are produced (Section 2)
- use-case-led design of EDR clusters, to provide connectivity, Edge processing and storage (Section 3);
- comprehensive study of data storage, processing, shelf-life- and freshness-based services in EDR clusters (Section 4);
- feedback and management algorithm for local EDR performance adaptation, EDR resource and per-service QoS feedback (Section 5);

2 BACKGROUND

The main purpose of EDRs is to provide a stable, secure and efficient storage and processing environment for data produced at the Edge of the Internet. In this section, we define the four connected system parts that affect each other's service provision parameters and functionality, depending on policies, demands and locality (Figure 1).

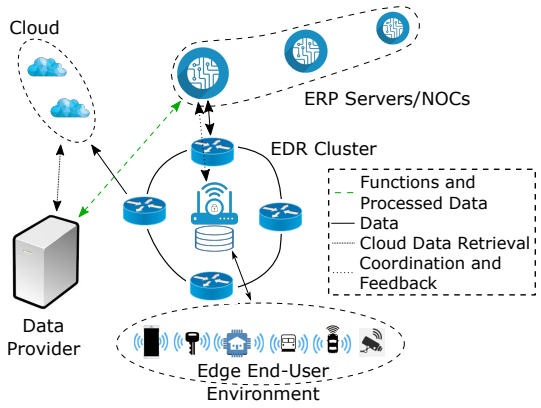


Figure 1: ERP System Diagram

From a data perspective, the primary part of the system is the class of devices/systems producing data. The Edge Data Producers (EDPs) are end-devices or end-systems, which produce data with specific topics, mobility, frequencies, types, scopes and destinations. These devices/systems and their data parameters are associated with specific Application/Data/IoT Providers and their users.

Technically, the most important part of this system are the EDR clusters, which provide a secure, stable and service-rich environment for Edge Data storage and processing. Security can be ensured through service-specific resource allocation, in containers or enclaves. EDR clusters mainly provide stability by storing and processing (mobile) data locally, and providing persistent network services for the respective data. EDR clusters are also the hubs that connect this entire system, efficiently and adaptively. An EDR cluster's management, security and service provision policies are monitored and managed by its local ERP server, through its service provision and resource allocation feedback and management algorithms.

ERP management entities (Network Operations Centre [NOC] and servers) represent key commercial entities. These may be of a central (ERP NOC), or local (cluster servers) management type. ERP management entities represent the commercial and policing part of the system, which is mostly concerned with ERP system security and management policies. These entities govern security bootstrapping, function and data coordination, service provision policing, system trade-off monitoring, performance monitoring and management of commercial policies between their systems and other entities.

The other commercial part of the system is the Data Provider. This commercial and network entity provides its service and security policies to its own set of users and negotiates commercial, network and security policies with the ERP(s). Associated EDPs and the functions provided for data processing are also managed by these providers. The main concerns of these providers lie in bootstrapping EDPs with data generation certificates and providing secure functions to the network. Security bootstrapping assures EDRs that producers provide secure data, with appropriate parameters.

An equivalent system to this is an SDN-controlled system. Here, instead of control tables, data names, services and resources are manipulated in the network control plane. Similarly, ERP servers are controlled by policies dictated by and provide updates to ERP NOC(s), which set the main operating policies, analogously to SDN controllers, guided by a (set of) network-control application(s).

This paper tackles one of the most basic, but very important challenges that this system faces: EDR-specific management and feedback. Considering the above analogies, we develop a feedback algorithm using structures employed in SDN (tables and organisation) for systems control, and a SNMP-like communication protocol to convey the outputs of the algorithm (and later inputs from the ERP servers). The EDR resembles a SNMP Agent and its associated resources and metrics resembling MIB objects. The organisational structures providing the services in each EDR resemble "match-plus-action" (control) tables. Thus, the control structures of EDRs should be SDN-based, while the communication protocol should be SNMP-like. This will be presented through the commands table and its associated algorithms, for processing and storage services, in Sub-section 5.2.

3 DATA REPOSITORY DESIGN

The main purpose of EDR-based systems was defined and presented in the last section. Now we shall examine each part of the system in more detail.

EDRs were developed to be modular, configurable and scalable. The functionalities illustrated below demonstrate these features.

Storage: All repositories benefit from network persistent storage capability, being able to buffer and retain data. Depending on service requirements, this functionality may suffice for certain services.

Processing: All repositories have processing capabilities for application-specific service provision. Processing is done independently of line-speed operations, in the application layer.

Storage association and management - After a message is received by a repository, it is added to storage management and the newest processing message is then processed (Points (1) of Fig. 2).

Message processing and update handling - Each update consists of a processing phase and a depletion phase, on each simulation cycle. The update process is presented in points (2) of Fig. 2.

Main Question - Considering the system design, the main issue will be approached heuristically. The following questions provide

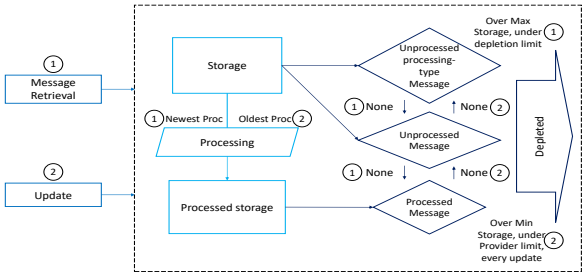


Figure 2: Repository update and handling processes

the main reasoning for our evaluation and are inspired from existing Edge networks research [10, 15]: What is the algorithm that determines the most efficient resource configuration, based on available resources, services provided, traffic analysis and EDP types? Could this algorithm provide simple yet effective service quality feedback and local adaptivity, for specific services, under certain conditions?

4 EVALUATION

Consider that storage is used within limits, and while the packets' shelf-lives have not expired, the packets are stored in EDRs. In this context, it is important to assess the percentage of data processed within the freshness period or shelf-life or deleted/offloaded after shelf-life expiry. If the EDRs start overflowing or the packets' shelf-life expires, before they are processed, they will be considered unsatisfied. These shall be briefly covered in sub-section 4.3, and later in more detail, in sub-section 5.2.

4.1 Setup and Assumptions

An EDR cluster was implemented and evaluated using an extension[1] of The Opportunistic Network Environment (ONE) simulator[2] [6]. The following assumptions were used for testing and evaluation.

- processing power was measured in threads, assuming equal power in all threads [7]
- one thread uses a specific amount of time for a service type
- function execution takes a finite amount of time per content type, assuming execution sessions established
- for simulation efficiency, the smallest step of processing time is 0.1s (realistically it could be smaller)
- EDPs, EDR cluster, Data Providers and Cloud Providers are securely bootstrapped, for data and function verification
- all EDPs and EDRs use their private keys and pre-fetched certificates/trust anchors to produce secure data
- freshness period accounts for only one processing cycle
- under service strain, the retrieving EDR will compress and send processing messages to the cloud before processing
- compression is executed on individual messages, and not on the aggregated messages, as a file, for ease of simulation

[1] The ONE Repos: https://github.com/Chrisys93/the-one-repos
[2] The ONE: http://akeranen.github.io/the-one/

These assumptions will be revisited in further work, to tackle more complex network and management problems. The EDP generation rates used in this paper were based on CISCO predictions [3, 5] for 2022-24, several case studies ([2, 11, 16]) and reasonable approximations. Average data production is 85GB/node/day and varies with type of EDP.

Example use case - An EDR cluster with the specifications:

- a total upstream bandwidth limitation of 10Gbps
- 500 cars, 40 pedestrians and 15 buses served
- three hours accounting for peak traffic hours of the day
- all data required on mobility patterns, generation rates and served file types known
- the cluster deployment area considered to be Helsinki city

The example case cluster is comprised of 80 repositories, distributed in a cell-like structure covering central Helsinki. The mobile nodes generate messages as follows (non-processing:processing): 2:0 per pedestrian per 5s, 0:2 per car per 2s with a processing delay of 0.1s and 1:2 per bus per 2s [17] with a processing delay of 0.2s (where non-processing refers strictly to storage-bound messages). Messages are 1MB in size. Each EDR serves 10GB storage capacity, 10 parallel processing threads and a 10MB/s maximum output bandwidth. Although the cars are the main type of producers studied in this case, all nodes generate realistic network traffic.

4.2 Use cases

The results necessary in algorithm development mainly depend on the type of EDP. The producers are diverse in their movement patterns and service needs. We use the *example use case* to establish 6 different service models for each of the following 4 use cases. One (6th) is a common, "golden standard" model, used for QoS comparisons. All repositories in each scenario provide one service, throughout each simulation scenario and model configuration. Note that storage capacity was reduced to 5GB/EDR in these assessments.

We first consider **sensing equipment** such as temperature and pressure sensors. These devices can produce small amounts of data at specific intervals of time and are generally static. The sensor data is normally ready to be processed and volatile after use. This device type would be most prevalent in **office buildings**.

The second use case considered for our purposes is **video streaming from immobile Edge devices**. **Home** surveillance and doorbell cameras create large data feeds. While this data may be critical, it is not necessarily provided continuously. Although it can be stored for longer periods of time after processing, it may be offloaded to clouds, as long as it is not solicited/required within a certain amount of time.

An example of mobile EDPs generating small-sized data can be **measurement sensors within a car**. Such a sensor could be measuring battery charge, GPS location etc. [5]. These kinds of data can have shorter freshness periods, but may have longer shelf lives, for further network processing and availability. Each data point may be less important and more volatile than larger messages. Thus, this kind of data could be offloaded to the cloud periodically.

The most complex example of large-sized data could be **video or LIDAR data** [5], created by a **public transport vehicle (e.g. TfL Bus**, Underground etc.). These kinds of data can normally have larger freshness periods, if they are to be processed. The shelf lives

of these Data packets could be longer, due to their longer relevance period for Edge aplications. Once stored and satisfied, Data packets could either be offloaded or deleted periodically.

4.3 Evaluation for algorithm development

Looking at Fig. 3a, we can clearly see the cases in which most processed messages are fresh. Thus, the results for message processing efficiency percentages can be compared, to determine the mean processing efficiency for specific models and parameters, throughout the cluster.

On the storage side (Fig. 3b), different non-processing data shelf-lives affect satisfaction rates of offered storage services. We can note that shelf lives determine the maximum storage capacity required for a service. Thus, *Storage Mode* is used to keep messages within storage in underused repositories, for both efficiency and better storage service provision. The number of messages stored longer than their shelf-life (overtime) is normally low, unless repositories are configured in storage mode. If messages are stored overtime, however, storage is used inefficiently and storage satisfaction may decrease, if used inappropriately. Thus, storage mode is a valid and useful option only for services with reduced or infrequent burst traffic. These network parameters help in developing a management and feedback algorithm meant to help ERP servers in decisions on service placement, resource allocation and QoS management.

5 MANAGEMENT AND FEEDBACK ALGORITHM

5.1 Parameter descriptions

Assessing the processing and storage capabilities and associated QoS, we can ascertain three different ratios that can be used as system metrics: fresh processing rates, shelf-life processing rates and ingress traffic.

Fresh processing rates refer to the capability of the EDR to process all ingress messages within their freshness periods. This is obtained by scaling the approximated per-message processing delay with the freshness period. **Shelf-life processing rates** are used for determining the total processing capability of EDRs. It is assessed in relation to the shelf-life of processing messages and the associated *fresh processing rate*, in the context of the type and expected processing delay of the ingress serviceable messages. The **message influx**, as the name indicates, is a measurement of the ingress processing message traffic targeted at the respective service. It was also concluded that non-processing message shelf-life does not influence the quality of processing service on the repositories.

The storage mode setting is one of the parameters under EDR control that can directly affect QoS and resource efficiency. However, more importantly, **the shelf-life** associated with **the size and type** of any ingress messages being accepted as part of a service model are the determining factors of storage usage and storage QoS. Most importantly, the common parameter which affects both storage and processing, **the message influx** has one of the highest impact factors to be considered in our algorithm.

Note that EDR **upload limits** put a limit on the amount of messages serviceable in a specific amount of time. However, upstream bandwidth can be utilised better by changing the compression rate(s) and compression-to-deletion ratio(s) associated with the service(s).

The normal SNMP commands are general and more applicable to routing, thus, some need to be adapted to our system's requirements. We provide Table 1 for reference and demonstrate the use of some of the commands within Algorithms 1 and 2

Command Name	Actors	Description
Performance Update	server to EDR	Periodic request for QoS update
Heartbeat	EDR to server	Periodic update on system performance, configuration and resource usage
Configure	server to EDR	Change MIB object values
Trap	EDR to server	Asynchronous Warning or Notification (& suggestion) message
Response	EDR to server	Response to server

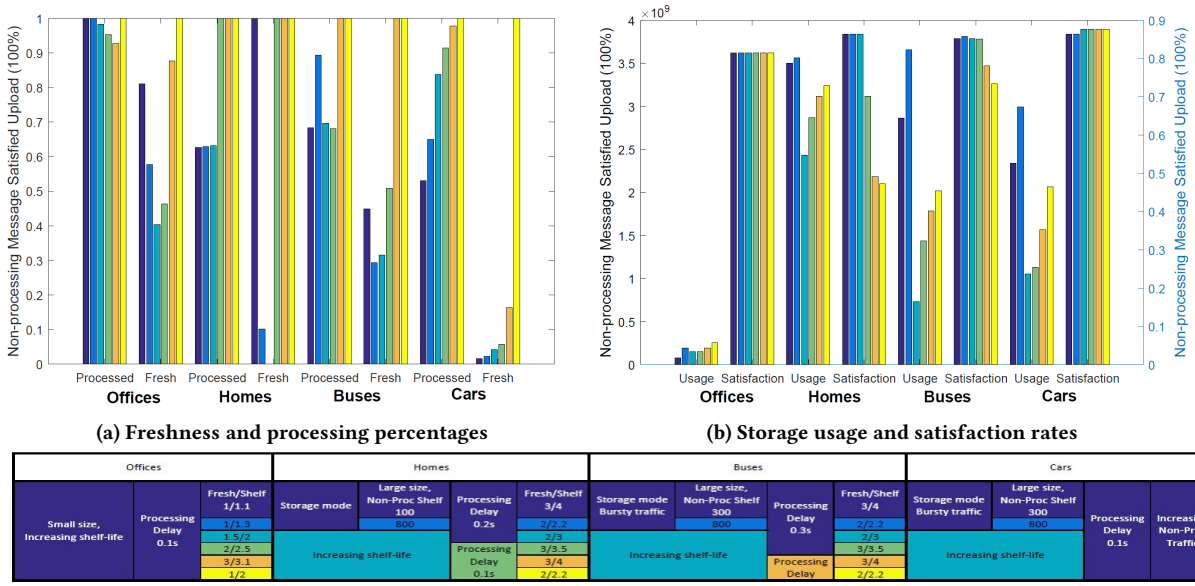
Table 1: Table of algorithm generated and SNMP commands

5.2 Algorithm

Now all the necessary metrics and values associated with the use cases studied have been explained. An EDR management and feedback algorithm that returns a measurable trade-off can now be presented.

The algorithm outputs **(warnings, notifications, performance updates and heartbeats)** can be used to determine needed changes, in accordance with the existing data types and service model(s). The algorithm is meant to adapt resources and give the feedback needed by the ERP server to dynamically apply different service- and resource-based optimisations within the cluster. A representation of this communication process is provided through Fig. 4, Algorithm 1 and 2.

The point of these algorithms is to provide an SDN-like management structure, to operate just above the network layer, depending on individual EDR performances (diagnosed through application-layer SNMP-like messages). With the server having its own view of the EDR cluster, it can obtain and optimise manageable object values in MIB records and service provision tables, for adapting the system and services for the network environment. In other words, EDRs send heartbeat and "trap"-like update messages to ERP servers through this algorithm. While gathering this information, EDRs also apply local resource optimisation measures, for currently running services. Depending on their cluster-wide view of the EDRs, ERP Servers can then "decide" on what actions to take, to optimise service placement, for performance and QoS (to be discussed in future work). Following these actions, the feedback process can restart.



(a) Freshness and processing percentages

(b) Storage usage and satisfaction rates

Figure 3: Evaluation of different simulation conditions using the general node configuration

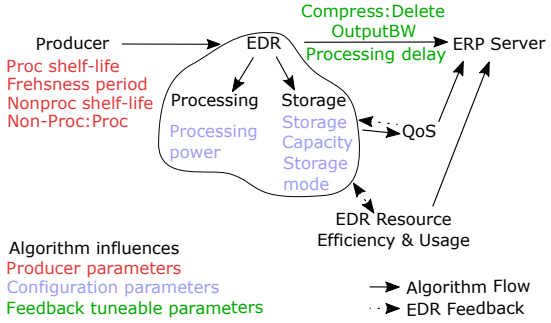


Figure 4: Main structure of algorithm

As sub-figure 3a shows, the processing success rates depend mostly on the ratio between the freshness period and shelf-life. However, the processing success rates depend on the scale of the freshness period and shelf-life, compared to the function execution times (processing delay), which are generally very small. The ratio between the first two has to be close to 1 to have any effect on the success rates. Thus, we consider the traffic, delay, freshness period and shelf life, as the most relevant parameters for dynamic adaptation and QoS, to be fed back to ERP servers.

Moving on to the parameterisation of processed message freshness and sub-figure 3a, we can infer that the main decision parameter changes to the processing delay. However, the freshness period and shelf-life still play an important role in determining the "freshness" of processed messages. Here, the ratio between the freshness period and processing delay represents a ratio that does not change the processing rates by much by itself. However, this ratio is a potent modifier of the impact that the ratio between the delay and the difference between shelf-life and freshness period has.

Algorithm 1
Fresh-Processing Service QoS and Resource Allocation

Require: Processing Service feedback on each heartbeat
if All Power Used Continuously **then**
Include Power Usage in heartbeat
if Processing Success Rates > required **then**
Processing Power decrease within policy limits
else if Processing Success Rates > acceptable **then**
QoS value warning on next QoS update
Warning of lacking resources
else
Include Service Provision status in heartbeat
Service Provision Reduction imminent Warning
end if
else if Processing Success Rates > required **then**
Service provision increase notification
if ERP Service Increase POSITIVE **then**
Include Service Provision status in heartbeat
end if
else if Processing Success Rates > acceptable **then**
case i
1: Processing Power decrease within policy limits
2: Provider policy change needed notification
else
case i
1: Processing Power increase within policy limits, Service provision warning, Include Processing Power Usage in heartbeat
2: Provider policy change needed notification
end if

We can also note that the most important modifier of both processing success and freshness rates is the ratio of input non-processing to processing messages, representing ingress traffic.

Algorithm 2
Non-Processing Service QoS and Resource Allocation

```
Require: Storage Service feedback on each update
  Include Service Provision status in heartbeat
  Include Storage Usage in heartbeat
  if Overflowing storage then
    if StorageMode ON then
      StorageMode = OFF
    else
      Storage Capacity Decrease within policy limits
      QoS value warning on next QoS update
    end if
    if Non-ProcSatisfaction < acceptable then
      Service provision decrease
      Warning of lacking resources
    end if
  else if Non-ProcSatisfaction > required then
    Service provision increase notification
  else if Max Service Storage Capacity then
    if Non-ProcSatisfaction < required then
      QoS value warning on next QoS update
      Storage Capacity Increase within policy limits
      Provider policy change needed notification
    end if
  else
    if No awaiting services AND bursty traffic then
      StorageMode = ON
    end if
  end if
```

Clearly, when more storage is used, messages' shelf-lives limit satisfaction rates. Considering this, the shelf-life associated with a specific service determines its satisfaction rates and whether the receiving EDR storage overflows. This naturally also depends on the size of messages and traffic. More exact values for this tipping point of one of the models, as shown in Fig. 3b, are: between 500 and 600s non-processing shelf-lives, processing shelf-life under 50s, a rate of 1 processing message for each 4 non-processing messages generated each second, a size of 1MB/message and static EDPs.

In the last three models of the cars and buses use cases we can see that, as shelf-life increases, the importance of the "non-proc:proc" traffic ratio decreases with respect to storage occupation. On the other hand, we can see that when sizes of messages are bigger and EDPs more stable/static, the importance of the "non-proc:proc" ratio is much higher. In this case, the more positive the ratio, the higher the chance for messages to be stored less than their shelf-life values (above a specific threshold - see Fig. 3b).

6 DISCUSSION AND FUTURE WORK

A study on **data and function aggregation, (re)placement and distribution** (considering resource allocation efficiency) would be worth investing more effort in, considering the findings in this paper. Furthermore, an **ERP server algorithm for QoS and EDR resource allocation** should be established.

In rare cases, more storage capacity, processing power and/or coverage can be useful for providing services effectively. In such cases, ***"Second Level"* repositories** are a potential solution, to supplement service capacity at the Edge and provide more management capabilities, for more localised decisions. These supplementary repositories could either be logically instantiated through dedicated units, meant to serve either as 1st level or 2nd level EDRs at all times. The evaluated scenarios provide an insight into distribution, storage, performance and analysis for data management strategies to be run on 2nd lvl EDRs. These strategies and their underlying systems and provision mechanisms will be further discussed in later work. Otherwise, it would be interesting to study the feasibility of offering **more, complementary services** per repository, where efficiency is high. This would complement the low-storage services with high-storage, for example.

The extensions of this system may further be developed using an Information-Centric Networks (ICN)-based implementation of the cluster ([8, 9] and/or [1]).

7 CONCLUSIONS

The assessment completed using the ONE simulator and its results produced a local management and feedback algorithm. This algorithm provides a first insight into the dynamic management potential of ERP systems. Through the feedback provided, it was demonstrated that the development of an EDR cluster management algorithm for the ERP system can be advantageous for different network conditions and certain service offerings. These concepts shall be developed, to form a more complete system, accounting for more aspects of networking not covered in this paper, like content-driven networking, security, commercial and management policies.

These evaluations are important as they evaluate which trade-offs should be considered more important for different service models. We also provide the feedback algorithm, for design and adaptation of systems that integrate EDR clusters. This has potential for further development, accounting for further complex parameters and metrics (e.g. data aggregation and locality). There is a wide range of developments which can be based on this system, depending on their final purpose of deployment.

ACKNOWLEDGMENT

This work is partially supported by EPSRC grant EP/N509577/1, the EPSRC INSP fellowship (EP/M003787/1) and EU H2020 DECODE project, under grant agreement number 732546.

REFERENCES

[1] Onur Ascigil et al. 2017. A keyword-based ICN-IoT platform. *Proceedings of the 4th ACM Conference on Information-Centric Networking - ICN '17* September (2017), 22–28. https://doi.org/10.1145/3125719.3125733

[2] Junguk Cho et al. 2016. ACACIA: Context-aware Edge Computing for Continuous Interactive Applications over Mobile Networks. In *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*. ACM, New York, NY, USA, 375–389. https://doi.org/10.1145/2999572.2999604

[3] Cisco. 2018. Cisco Visual Networking Index : Forecast and Trends. (2018).

[4] Kiryong Ha et al. 2014. Towards wearable cognitive assistance. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14* (2014), 68–81. https://doi.org/10.1145/2594368.2594383 arXiv:arXiv:1011.1669v3

[5] Joel Obstfeld. 2019. Global engineering:Enabling a Connected Transport future. http://coseners.net/wp-content/uploads/2019/07/Connected-Car-Architecture.pdf

[6] Ari Keränen et al. 2009. The ONE simulator for DTN protocol evaluation. *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques* (2009). https://doi.org/10.4108/ICST.SIMUTOOLS2009.5674

[7] Shweta Khare et al. 2018. Scalable Edge Computing for Low Latency Data Dissemination in Topic-Based Publish/Subscribe. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. 214–227. https://doi.org/10.1109/SEC.2018.00023

[8] Michal Krol et al. 2018. Computation offloading with ICN. *Proceedings of the 5th ACM Conference on Information-Centric Networking - ICN '18* (2018).

[9] Michał Król and Ioannis Psaras. 2017. NFaaS. *Proceedings of the 4th ACM Conference on Information-Centric Networking - ICN '17* 11 (2017), 134–144. https://doi.org/10.1145/3125719.3125727

[10] I Lujic and H Truong. 2019. Architecturing Elastic Edge Storage Services for Data-Driven Decision Making. In *13th European Conference on Software Architecture (ECSA)*. https://www.researchgate.net/publication/333371908_Architecturing_Elastic_Edge_Storage_Services_for_Data-Driven_Decision_Making

[11] Eton Manor. 2012. CASE STUDY Wi-fi access at London 2012 The London 2012 Olympic Park wi-fi network delivered 100 per cent availability throughout the Games. (2012).

[12] H. Moustafa, E. M. Schooler, and J. McCarthy. 2017. Reverse CDN in Fog Computing: The lifecycle of video data in connected and autonomous vehicles. In *2017 IEEE Fog World Congress (FWC)*. 1–5. https://doi.org/10.1109/FWC.2017.8368540

[13] Ioannis Psaras et al. 2018. Mobile Data Repositories at the Edge. In *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. USENIX Association, Boston, MA. https://www.usenix.org/conference/hotedge18/presentation/psaras

[14] M. Satyanarayanan. 2017. The Emergence of Edge Computing. *Computer* 50, 1 (Jan. 2017), 30–39. https://doi.org/10.1109/MC.2017.9

[15] Eve M. Schooler et al. 2017. An Architectural Vision for a Data-Centric IoT: Rethinking Things, Trust and Clouds. *Proceedings - International Conference on Distributed Computing Systems* (2017), 1717–1728. https://doi.org/10.1109/ICDCS.2017.243

[16] TfL. 2000. Case study: Transport for Londom. (2000), 1–2.

[17] Timespace 2018. *Timespace V400*. Timespace. Issue 4.

[18] Andrea Zanella et al. 2014. Internet of Things for Smart Cities. *IEEE Internet of Things Journal* 1, 1 (2014), 22–32. https://doi.org/10.1109/JIOT.2014.2306328

[19] Ke Zhang et al. 2017. Mobile-Edge Computing for Vehicular Networks. *IEEE Vehicular Technology Magazine* 12, June (2017), 2–10. https://doi.org/10.1109/MVT.2017.2668838

[20] Lixia Zhang. 2019. The Role of Data Repositories in Named Data Networking. In *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. 1–5. https://doi.org/10.1109/ICCW.2019.8756944