The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases

Alsayed Kassem, J.; de Laat, C.; Taal, A.; Grosso, P.

DOI
10.1109/ACCESS.2020.3028051

Publication date
2020

Document Version
Final published version

Published in
IEEE Access

License
CC BY

Link to publication

Citation for published version (APA):
Alsayed Kassem, J., de Laat, C., Taal, A., & Grosso, P. (2020). The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases. *IEEE Access*, *8*, 179909-179920. https://doi.org/10.1109/ACCESS.2020.3028051

Received September 14, 2020, accepted September 24, 2020, date of publication October 2, 2020, date of current version October 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3028051

The EPI Framework: A Dynamic Data Sharing Framework for Healthcare Use Cases

JAMILA ALSAYED KASSEM[1], CEES DE LAAT[2], (Member, IEEE), ARIE TAAL[1], AND PAOLA GROSSO[1], (Member, IEEE)
[1]MutliScale Networked Systems (MNS) Research Group, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
[2]Complex Cyber Infrastructure (CCI) Research Group, University of Amsterdam, 1098 XH Amsterdam, The Netherlands

Corresponding author: Jamila Alsayed Kassem (j.alsayedkassem@uva.nl)

This work was supported by the Dutch Science Foundation in the Commit2Data Program through the Enabling Personalised Intervention Project under Grant 628.011.028.

ABSTRACT To support the current trend of personalised medicine, a collaboration between different healthcare providers is increasingly vital. The main element is the ability to share data among all parties while abiding by a data sharing policy. The EPI (Enabling Personalised Intervention) project addresses the problem of personalised diagnosis by developing real-time monitoring services and digital health twins. The EPI services run over adaptive computing infrastructures which provide more flexibility to accommodate the different requests. This paper proposes the EPI framework to support these novel health services over programmable infrastructure. The framework works on aligning the parties' ability to share data with the policy defined beforehand. We explain the approach by introducing the framework's data sharing logic model. We define the formalism of the logic model to deduce feasible data movements between and possibly satisfy a data collaboration request. We reinforce the framework's logic model by introducing the algorithms running on this federated system to simulate its workflow. We provide three healthcare use cases running on a typical EPI infrastructure. We evaluated our model according to three relevant parameters, performance, feasibility, and aggregation power, and we can conclude that our framework supports the required interoperability between the EPI partners.

INDEX TERMS Data sharing, dynamic infrastructure, healthcare, information flow control, medical information system.

I. INTRODUCTION

Consistent, reliable, and interoperable health data sharing is needed to support different healthcare applications such as decision making, monitoring, and planning. This is evidently important in extreme cases and disasters. As an example of that, the international health data-sharing efforts are facilitating effective combat strategies in the light of the COVID-19 pandemic spread across the world [1].

The exchange of health data is a key enabler to efficient and high-quality healthcare. This concept has been there for over a decade, through which many laws, requirements, and standards were adopted to regulate and preserve the patient's rights for privacy and security [2], [3]. Ultimately, data collaboration between healthcare providers/parties considering the mentioned requirements can empower patients through the healthcare cycle.

The heterogeneity of the network infrastructure across different health care domains brings forth a challenge of ensuring seamless information flow. The EPI[1] project aims to provide self/joint management of medical treatments throughout the healthcare cycle. That can be made possible by effectively enabling data sharing across parties. And as a result, the EPI project intends to support various applications, such as privacy-preserving machine learning (ML) and small data-set models.

The EPI project requires a dynamic infrastructure to adapt to different requirements. The infrastructure should optimally exploit the heterogeneity and programmability of computing/networking resources. The paper proposes the dynamic EPI infrastructure that we set up according to a data-sharing logic model. We aim to align information flow with the policy *i.e.* data sharing rules. The framework defines each node's capabilities with *attributes* and detects heterogeneity

The associate editor coordinating the review of this manuscript and approving it for publication was Jenhong Tan.

[1]https://delaat.net/epi/

with the concept of *areas*. After that, the framework automates the setup of an overlay of virtualised functions to bridge attributes' gaps. The logic model proposed addresses the heterogeneity and provides interoperability, undegraded security, and reliability of information flow across various domains.

The research question to be answered is, *How to support health data sharing for heterogeneous parties collaborating within an infrastructure?*

In this paper, we answer this question, and we contribute to the existing literature of health data sharing frameworks:

- We propose a logic model to determine achievable data sharing within heterogeneous environments of infrastructure
- We match the achievable data sharing with the defined policy to ensure alignment
- We define the concept of EPI areas that maps to the possible data flow
- We implement an algorithm that creates EPI areas
- We evaluate the algorithm according to relevant parameters
- We define a typical EPI infrastructure and its components
- We apply the logic model to the infrastructure running specific EPI use cases

In Section II, we discuss several existing data-sharing frameworks in the context of health to give a background about the topic. After that, we introduce the EPI framework in Section III, the logic model behind it and its components in Section IV. To validate the logic model, we show the simulation experiment and evaluate the results in Section V. In Section VI, we expand on the results' discussion. In section VII, we show how the EPI framework functions with an EPI infrastructure. Moreover, we explain the framework's functionality in the light of different health use cases in Section VIII.

II. HEALTH DATA SHARING FRAMEWORKS

Latest advancements in the field of informatics inspired much of research utilising said technologies in the healthcare domain. Such development is currently essential due to the impractical traditional data sharing methods. According to surveys [4], roughly 15% of patients visiting a doctor were asked to supply their radiology report personally (like on a DVD), and another 5% had to redo their tests.

Healthcare providers realise that and are migrating towards the usage of electronic medical records (EHR). It is a broad consensus that sharing data in the medical spectrum can be beneficial in terms of cost-efficiency, preventing redundancies, cooperation between stakeholders, reliable research and discoveries. Achieving secure health data sharing can result in an efficient and effective healthcare cycle managed by the patients/healthcare stakeholders.

Health data sharing frameworks exist, but they are catered to satisfy a single, specific use case. That makes the architecture rigid and less suited to support different applications. These frameworks do not address the different capabilities present in all network endpoints, and actually assume identical operational possibilities on all participating nodes. This is far from the current state-of-the-art in computing infrastructures.

In this section, we aim to highlight some of the literature. The way that we gathered the literature is as follows. For each query on Google scholar, we chose the top result. We queried "secure sharing infrastructure", and the top result was [5]. The search results were further filtered with a 2018+ time frame to represent more recent research [6].

We queried "secure data sharing infrastructure", and the top result was [7], then added a time filter of 2018 which gave the top result [6]. We queried "health data secure monitoring" with top result [8], and got [9] as a top result for the results published after 2018. The mentioned papers referenced interesting literature like [10]–[13]. Some frameworks provide interoperability of data but for a single use case. Roelofs *et al.* [7] discuss an open-source infrastructure to share medical data internationally for radiotherapy studies. They provide interoperability of data by running a multicentric data mining. While Sartipi *et al.* [5] introduce an infrastructure that integrates PACS (Picture Archiving and Communication System) and HL7 (Health Level 7) and aims to have a homogeneous, and internationally accepted EHR (Electronic Health Records).

Other frameworks rely on single specific technologies or devices. In more recent papers, frameworks leverage blockchain as a single infrastructural data sharing solution. Patel [6] favours an established consensus on a blockchain ledger over third party intermediates. Another framework utilising the same tools is MeDShare [10]. The system integrates smart contracts and access control protocols to provide a tamper-proof ledger of health data.

MedBlock [11] compares with the MeDShare system and shares data via blockchain. Unlike MeDShare, MedBlock does not maintain a ledger of audited behaviour, instead, it provides an information management system. Moreover, Thilakanathan *et al.* [8] propose a platform based on cloud data services. The platform adds a layer of security by proposing a security protocol with ElGamal and proxy re-encryption key exchange schemes.

A different application of medical data sharing is secure monitoring. Anoop and Parani [12] propose streaming an enormous chunk of medical data collected through wearables using BSN (Body Sensor Network) to provide real-time monitoring of the patient's health status. Manogaran *et al.* [13] design a new architecture of IoT environment to store, process, and share big data. Griggs *et al.* [9] build a blockchain-based system to monitor the patient's status remotely. The system uses smart contracts on a private Ethereum network to handle medical information.

Table 1 shows a comparison between these frameworks according to a set of considered features. In light of the frameworks discussed, some elements, or combination of

TABLE 1. Comparison between health data sharing frameworks.

Framework	Tools	Use Case	Security considered	Access control	Data audit	Laws	Dynamic	Interoperability
[5]	PACS, XDS, OpenID	Medical images sharing	✓	✓	✓	×	×	✓
[7]	SQL, DICOM, Key scheme	Radiotherapy research data	×	×	×	✓	×	✓
[6]	Blockchain	Medical images consensus	✓	✓	×	×	×	×
[10]	Blockchain, Cloud	Medical data sharing	✓	✓	✓	×	×	×
[8]	Cloud	Monitoring and sharing data	✓	✓	×	✓	×	×
[11]	Blockchain	Medical data sharing	✓	✓	✓	×	×	×
[13]	Cloud, Fog computing	Smart healthcare monitoring	✓	✓	×	×	×	×
[12]	WBANs	Secure monitoring	✓	✓	×	×	×	×
[9]	Blockchain smart contracts	Remote patient monitoring	✓	✓	✓	×	×	×

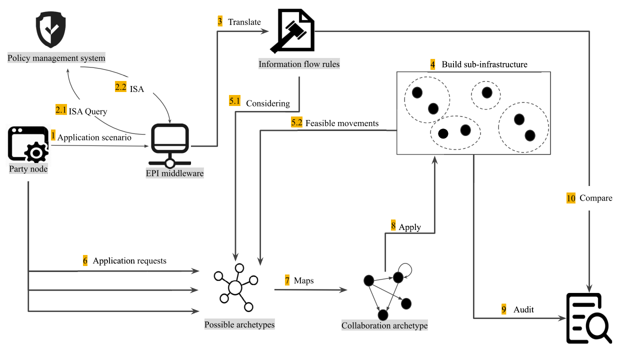


FIGURE 1. A high-level view of the EPI framework architecture after an application setup is initiated.

elements, might be lacking. For instance, some proposals consider security on one hand, but do not address relevant laws and auditing. On the other hand, other frameworks address interoperability and data model usage across different sectors, but do not offer access control and security to protect sensitive data. More importantly, all these frameworks are catered to satisfy a specific use case which makes the architecture rigid and hard to support different use cases.

The majority of the proposed work seems to be application-specific. These infrastructures are mainly static and still offer a "one fits all" standard. None of these frameworks addresses the data movement requirements for generic use cases. In EPI, we formalise a methodology to support data sharing requests across providers with heterogeneous resources, which relies on the programmability of the infrastructure.

III. THE EPI FRAMEWORK

The EPI infrastructure is the collection of all networking/ computing/ and storage *nodes* provided by EPI parties. The EPI framework will cater to any application scenario by building a sub-infrastructure.

Figure 1 illustrates a high-level view of the proposed framework. An *application scenario* initiated by the collaborating parties (1) specifies the resources needed, and the data collaboration goals. The information-sharing agreement (ISA) describes the data sharing policy between parties, *i.e.* a policy is the group of rules governing data movement/usage [14]. The EPI framework queries ISA from the policy management system (2). The received ISA is further translated to a set of rules (3) that are recorded into a log.

The EPI framework uses a logic model (Section IV) to group nodes into areas that describe feasible data movements between them (4). On top of feasible data movements, the framework also considers the information flow rules to determine possible archetype mapping (5). The model describing the pattern of data movement is called *collaboration archetype* [15].

An *application request* is the actual requested data movement between parties. After building the sub-infrastructure, application requests are introduced (6). A single application scenario can refer to multiple application requests, and archetypes represent the aggregation of all possible data movements. When an application request maps to an existing collaboration archetype (7), that is further applied in the sub-infrastructure (8).

An auditing log is maintained of the data movement behaviour (9). The audit log is compared to the rules log, hence provides accountability (10).

IV. THE DATA SHARING LOGIC MODEL

This section explains the logical model of creating and assigning areas nodes, hence creating a distinction of nodes' capabilities. Initially, there exists a clear view of all possible parties (e.g. hospital, research centre, rehabilitation centre). The nodes can partake or initiate an application scenario setup.

We assume that a higher level of the architecture filled out an ISA when a set of parties (healthcare domains) initiates an application scenario. The ISA is assumed to consider all requirements under which data collaborating is allowed, and it gives back a set of permission rules. These permission rules are further translated into rules, which leaves the issue of actually moving the data. The issue arises due to the data movement depending on the compatibility and data interoperability between parties.

A. SUB-INFRASTRUCTURE INITIALISING

A health domain/provider party assigns a number of nodes which are the endpoints of physical/virtual resources. The first step is to establish "who" can offer "what". We represent all resources endpoints included in the infrastructure by the set N of nodes:

$$N = \{n_i \mid i = 1, \ldots, m\}, \tag{1}$$

where n_i represents a single node in the infrastructure and m is the total number of nodes.

Attributes refer to what a node can support in terms of networking, computing, and storage functionalities. Possible attributes can specify software and computing resources: software tools, response scalability, CPUs. It can also specify security and network resources: identity and access control mechanism in place, data encryption and key management, data anonymisation, firewalls, scalable network links. Attributes for storage resources: maximum data storage, database tools, structured EHR/ unstructured. Let A be the set of all possible attributes:

$$A = \{a_k \mid k = 1, \ldots, d\}, \tag{2}$$

where a_k represents a distinct attribute and d is the total number of attributes.

Once the application scenario is made clear, a sub-infrastructure is initialised to support it. The sub-infrastructure is the set of nodes supporting a specific application for a duration of time. Let N_{App} be the set of mentioned nodes, such that:

$$N_{App} \subseteq N, \tag{3}$$

where N is the silo of resources of all parties and N_{App} specifies the nodes the collaborating parties use in the application scenario. The nodes needed to support the application are determined and the larger set N is filtered down. Each node's capabilities are described by a set of infrastructural attributes, which can be any subset of A. Subsequently, every node n_j in N_{App} is assigned $A_j \subseteq A$.

The model inspects nodes within N_{App} and queries the attributes associated with each node. We define a mapping f_{App} between the set of nodes N_{App} and the power set $\wp(A)$ of A, such that $f_{App}(n_j) = A_j$. Eq. (4) notates the many to one relation between nodes and attribute sets:

$$f_{App} : N_{App} \rightarrow \wp(A), \tag{4}$$

f_{App} is used to determine μ, where μ is the set of attribute sets related to nodes in N_{App}:

$$\mu = \{A_j \mid j = 1, \ldots, ||N_{App}||\} = \{f_{App}(n_j) \mid n_j \in N_{App}\}. \tag{5}$$

B. AREAS CREATION

The area abstraction is used to spot heterogeneity between nodes and deduce what movement is supported.

Let Θ_{App} be the set of all areas grouping N_{App} within the sub-infrastructure:

$$\Theta_{App} = \{\theta_p \mid p = 1, \ldots, l\}, \tag{6}$$

where θ_p is a single area associated with a distinct set of attributes, and Θ_{App} is a partitioning on the set N_{App}.

An equivalence relation on N_{App} is defined such that $n_g \sim n_h$ if n_g and n_h belong to the same θ_p and with $n_h \sim n_g$ denoting $f_{App}(n_g) = f_{App}(n_h)$ or $A_g = A_h$. As in Eq. (7), deterministic function α gives the area set Θ_{App} having N_{App} and μ as an input:

$$\alpha(N_{App}, \mu) = \Theta_{App}. \tag{7}$$

As a result, areas map to the total resources and endpoints per application setup within a sub-infrastructure and the differences between said nodes. Based on that, supported data movement *channels* can be deduced from this mapping.

C. SUPPORTED CHANNELS WITHIN A SUB-INFRASTRUCTURE

The logic dictates that data movement is supported when Eq. (8) is satisfied, where $n_g \Rightarrow n_h$ represents a one data movement support between nodes n_g and n_h. That means that a directional movement from n_g to n_h is supported when the capabilities of n_h (A_h) is the same or a superset of that of n_g (A_g)

$$n_g \Rightarrow n_h, iff A_g \subseteq A_h. \tag{8}$$

Supported/unsupported movements are represented by channels that can be utilised via a collaboration between nodes. The α output is further processed and translated into a channel matrix by function β

$$\beta \circ \alpha(N_{App}, \mu) = C. \tag{9}$$

The adjacency C matrix represents all nodes connected by a channel, where a single entry c_{ij} is the channel between the sender node n_i and the receiver node n_j, such that C is a $s \times s$ square matrix with $s = ||N_{App}||$. A single entry c_{ij} is a Boolean value where 0, 1 represents the existence of a channel or the lack thereof, respectively.

D. APPLYING FLOW RULES

To determine and set up the channels is not sufficient by itself to control the information flow within the infrastructure. The application scenario's policy further uses the channels in place to regulate and dictate data movement. The rules describe all required data movements of an application scenario mindful of the policy.

R represents the allowed/denied data movement the application needs to run. The data flow rules R are set in the light of ISA that is placed by a policy and management system. Rules are translated into a matrix form to make it easier to aggregate with other variables:

$$\gamma(ISA) = R. \tag{10}$$

All the rules between the nodes are represented in an adjacency matrix R, such that a single entry r_{ij} refers to the rule between the sender node n_i and the receiver node n_j, such that R is a $s \times s$ square matrix. A single entry r_{ij} is a Boolean value where 0, 1 represents the denied and allowed data movement rules, respectively.

We differentiate between allowed/denied and supported/unsupported data movements. Ideally, rules restrict further supported movements. As an example, when $n_i \Rightarrow n_j$ is supported, this means that $c_{ij} = 1$. Moreover, if data movement utilising c_{ij} is also allowed, this means $r_{ij} = 1$, hence the rule set aligns with channels.

E. BRIDGE ATTRIBUTE GAPS

In other cases, rules and channels might not necessarily align, which can be an issue to run an application successfully.

The condition $A_g \subseteq A_h$ supports data movement from node n_g to n_h. Otherwise, A_g is not a subset of A_h. A bridging function is introduced to apply the missing attributes ϵ_{gh} and by that creating the previously missing channel, if possible. In Eq. (11), ϵ_{gh} indicates the missing attributes

$$\epsilon_{gh} = A_h \cap \overline{A_g}. \tag{11}$$

There is a known set of bridgeable attributes A_δ. A_δ is set a priori. Virtualised functionalities provide bridges on top of existing sub-infrastructure resources, and it is dependent on the capabilities of the infrastructure. The bridging function applies the missing attribute if $\epsilon_{gh} \subseteq A_\delta$. This supports the data movement and ensures compatibility and interoperability within collaborating nodes. The bridging function δ works as follows:

$$\delta(C) = B, \tag{12}$$

where δ takes as an input the matrix of channels, and returns matrix B of bridged channels. B is represented as a $s \times s$ square matrix. A single entry b_{ij} is a Boolean value, where 0 can mean either a none bridgeable channel or $\epsilon_{ij} = \phi$ (the channel already exists), and 1 means a bridgeable channel c_{ij}.

In case an entry $c_{ij} = 0$, and $\epsilon_{ij} \subseteq A_\delta$, this means that $\delta(c_{ij}) = b_{ij} = 1$. The two concerns that arise dealing with bridging functions are as follows:

- The cost associated with the bridging function offered in terms of time and complexity
- There exists an infeasibility ratio that needs to be considered in case $\epsilon_{ij} \not\subseteq A_\delta$; an unsupported data movement is unbridgeable

The three matrices serve as dependable variables that control the information flow. the information flow control (IFC) is computed by the aggregation of the channels, rules, and bridges. IFC is calculated by Eq. (13):

$$IFC = (R \wedge C) \vee B. \tag{13}$$

IFC is a square matrix of the same dimensions $s \times s$. A single entry $f_{ij} = (r_{ij} \wedge c_{ij}) \vee b_{ij}$ of IFC is is a Boolean value, where 0, 1 represent no flow/flow of information, respectively. IFC is the adjacency matrix that describes the aggregated directed graph of all possible archetypes in an application scenario.

A single application request Q is met when Q and IFC overlap, hence the archetype mapping exists. All matrices we mentioned C, R, B, IFC, Q have the same size $s \times s$.

F. APPLICATION REQUESTS

Once a sub-infrastructure is initialised, application request Q utilises the built framework with collaboration instances. An application request is the matrix of requested data movements among nodes in an application scenario sub-infrastructure.

A single entry q_{ij} is a Boolean value that describes a single requested data flow from n_i to n_j, such that 0, 1 map to an absence or existence of a requested directional flow. To satisfy an application request Q, Q should overlap with the matrix of supported, allowed, and bridgeable flows IFC.

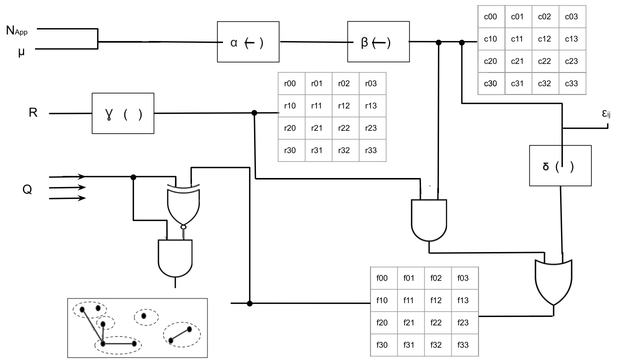


FIGURE 2. The different functions in the EPI framework and their relation with the application request.

TABLE 2. Apply / reject truth table.

f	q	overlap	apply
0	0	1	0
0	1	0	0
1	0	0	0
1	1	1	1

The previously discussed functions interact as shown in Figure 2. N_{App} and μ are the input to create the sub-infrastructure's EPI areas Θ_{App}. Next, the output is used to translate the channels into a matrix using β. Meanwhile, the policies turned rules are also translated into a matrix of allowed/denied flows by γ.

Rules and channels matrices are aggregated, and a single entry aggregation $r_{ij} \wedge c_{ij}$ gives an output true in case of alignment, else false. In case of a false, the δ is introduced to apply ϵ_{ij}.

After that, the IFC matrix is determined by the aligned and bridged channels, where $IFC = (R \wedge C) \vee B$. A set of Q's is then considered by applying the "XNOR" logic gate with IFC to determine the matches between the two matrices. The "XNOR" output is 1 in case of an overlap between q_{ij} and f_{ij}, and 0 is a mismatch.

The logic "AND" is applied to apply (1) or reject (0) a single data movement q_{ij} in an application request Q. That is further clarified in Table 1, where $q_{ij} = 1$ is applied only when it overlaps with $f_{ij} = 1$, otherwise the request is rejected.

V. SIMULATION EXPERIMENT AND EVALUATION

After discussing the framework in Section III, we introduce 4 algorithms that implement the EPI logic model. The first algorithm works by creating different areas associated with distinct attribute sets, such that N_{App} is an input. It appends nodes with the exact matching attribute set to the same area, as shown in Algorithm 1.

The first lines (3-4) are looping over all the nodes in the N_{App} set and query the node's attribute set. There are two cases that this algorithm deals with. Lines 8-11 check whether the area that is associated with a node's attribute set is already there, then it appends it to that area's node set. The lines 14-16

Algorithm 1 Algorithm to Assign Nodes to Areas

```
procedure createAreas($N_{App}$)
    $\Theta \longleftarrow \phi$                        ▷ initialise the set of areas
    for all $n \in N_{App}$ do                         ▷ loop over all nodes
        $A_n \longleftarrow getAttributes(n)$
        $flag \longleftarrow 0$
        for all $\theta \in \Theta$ do
            $A_\theta \longleftarrow getAttributes(\theta)$   ▷ retrieve attributes
            if $A_n = A_\theta$ then
                $\theta \longleftarrow \theta \cup \{n\}$
                $flag \longleftarrow 1$
                break
            end if
        end for
        if $flag = 0$ then
            $\theta \longleftarrow \{n\}$          ▷ create new area with node n
            $\Theta \longleftarrow \Theta \cup \{\theta\}$
        end if
    end for
    return $\Theta$
end procedure
```

are executed if the node belongs to a new area that does not already exist.

Once the logic model assigns all the nodes to areas, it also searches for subset relations between areas (included or distinct), as shown in Algorithm 2. After that, the framework deduces the supported channels between nodes within areas. We determine this relation according to the attribute set associated with each area.

Algorithm 2 Algorithm to Assign Subsets to Areas

```
for all $\theta_i \in \Theta$ do                              ▷ loop over areas
    $superset_i \longleftarrow \phi$        ▷ The set of all supersets to area
    $holder \longleftarrow \phi$
    for $\theta_j \in \Theta$ do                     ▷ compare with other areas
        $A_i \longleftarrow getAttributes(\theta_i)$
        $A_j \longleftarrow getAttributes(\theta_j)$
        if $A_i \subset A_j$ and $i \neq j$ then          ▷ i is a subset to j
            $holder \longleftarrow holder \cup \{\theta_j\}$
        end if
    end for
    $superset_i \longleftarrow superset_i \cup \{holder\}$
end for
```

In Algorithm 2, we loop over created areas twice (lines 1-4) to compare both areas associated attribute set. The algorithm sets a subset-superset relation between areas (lines 7-11).

The model's logic dictates that if an area is a subset (with fewer attributes) of another area, it means that supported channels exist between this area's nodes and the nodes in all superset areas (with more attributes characteristics). Algorithm 3 processes the created areas and deduces channels, as follows.

Algorithm 3 Algorithm to Deduce Channels Between Nodes

```
procedure checkChannels($\Theta$)
    Let $C = \{c_{ij}\}$ be a new sxs matrix with $\forall c_{ij} = 0$
    for $\theta_i \in \Theta$ do                                ▷ loop over areas
        for $n_i \in \theta_i$ do          ▷ loop over nodes in each area
            for $n_j \in \theta_i$ do             ▷ nodes in the same area
                if $i \neq j$ then
                    $c_{ij} \longleftarrow 1$                    ▷ channel exists
                end if
            end for
            if $supersets_i \neq \phi$ then         ▷ $\theta_i$ has supersets
                for $\theta_j \in supersets_i$ do
                    for $n_z \in \theta_j$ do
                        $c_{iz} \longleftarrow 1$
                    end for
                end for
            end if
        end for
    end for
    return $C$                          ▷ the matrix of all channels
end procedure
```

Algorithm 3 loops over nodes in each area (lines 2-3), and initialises existing channels between nodes within the same area (lines 4-6). In lines 9-12 the algorithm checks whether a node's area has a subset-superset relation area if so channels between this node and the nodes in the other area are initialised by setting the corresponding entry in matrix C to true.

After creating areas and deducing their channels, we consider the rules. We assume that the rule matrix was set by the policy management system involved, and then translated by γ. Then, take into account the possible bridging functions that can compensate for missing attributes and add to supported channels C. Function $\delta(C)$ results with the matrix B, where $b_{ij} = 0$ in case channel c_{ij} already exists, i.e. $\epsilon_{ij} = \phi$ or isn't included in A_δ.

The algorithm checks for available bridges (matrix B) and matches it to the missing channels between nodes is as in Algorithm 4.

In Algorithm 4, the algorithm in line 2 loops over all nodes in N_{App} and checks if it has supported channels to other nodes in lines 6-7. In case the channels to other nodes are missing, the algorithm in lines 11-13 checks if the attribute difference between the two nodes is bridgeable. As a result, the algorithm returns the matrix of updated channels with the added bridges.

The framework uses the *IFC* matrix to push the sub-infrastructure setup and support possible application requests. Application requests are successful when Q overlaps with *IFC*. Otherwise, not all data movements within this request might be possible. We deploy the algorithm on a server to emulate a framework prototype.

Algorithm 4 Algorithm to Add Bridges

```
 1: procedure checkBridges(C)
 2:     for $n_i \in N_{App}$ do
 3:         $A_i \longleftarrow getAttributes(n_i)$
 4:         for $n_j \in N_{App}$ do
 5:             $A_j \longleftarrow getAttributes(n_j)$
 6:             if $c_{ij} = 1$ then
 7:                 Channel already exists
 8:             else
 9:                 $\epsilon_{ij} \longleftarrow A_j \cap \overline{A_i}$        ▷ the missing attributes
10:                 $A_\delta \longleftarrow getBridges()$        ▷ retrieve bridges
11:                 if $\epsilon_{ij} \in A_\delta$ then        ▷ bridgeable attributes
12:                     Apply bridging functions
13:                     $c_{ij} \longleftarrow 1$
14:                 end if
15:             end if
16:         end for
17:     end for
18:     return $C$        ▷ updated channels
19: end procedure
```

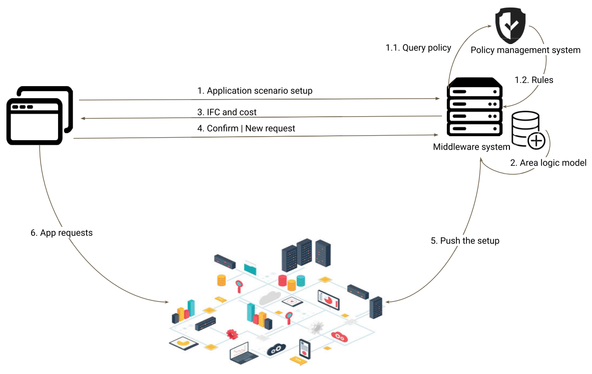


FIGURE 3. The EPI framework prototype and an illustration of its operational work flow interacting with the middleware system.

A. EVALUATION

We introduce a framework prototype in Figure 3 that uses the above methodology. We use that to measure the model's performance. Moreover, we propose formulas to evaluate the infrastructural properties that affect the feasibility and cost of different application requests.

1) PERFORMANCE

We set up a server to run the algorithm in the back-end, and maintain a database of all possible nodes. The server runs on a Dell machine with Ubuntu 18.04 operating system and 8 CPU cores, where the Python script of the logic model does not use multiprocessing.

We initiated an application scenario with the specifying nodes relevant for this request N_{App} (step 1 in Figure 3). The framework redirects the application query to the policy management system to query the regulating rules R of the information flow within this scenario (steps 1.1 and 1.2).

The server runs the logic model and lists the areas Θ_{App}, estimates B, and aggregates it with R (step 2). The middleware system deduces the *IFC* with all possible archetype models, and the setup cost is estimated (step 3). After the confirmation of the user (parties initiating the application) (step 4), the system pushes the IFC to the infrastructural level (step 5). Ultimately, a node then initiates the application requests (Q) to utilise the sub-infrastructure (step 6), and requests are applied or rejected accordingly.

We further evaluate step 2 in Figure 3 that runs the area logic model algorithm, and we estimate the performance of the framework (Algorithms 1, 2, 3, and 4). This measurement is important to test how the algorithm would scale with large input nodes in case of time-critical application requests. The objective is to measure the performance as the number of nodes increases and becomes more distinct (heterogeneous attribute sets).

To define heterogeneity we first calculate the average intersection rate AIR for μ, we consider the intersection rate of all distinct pairs in μ, as in Eq. (14):

$$AIR = \frac{\sum_{i,j;i<j}^{n} \frac{||A_i \cap A_j||}{||A_i \cup A_j||}}{\binom{n}{2}}. \tag{14}$$

To determine the average heterogeneity rate AHR of the nodes' attribute sets, we calculate AHR, such that:

$$AHR = 1 - AIR. \tag{15}$$

The performance variables that we measure are execution time and CPU time. We measure the execution time by recording the elapsed real-time between the invocation of the script and its termination. We also measure the CPU time is by the processing time of the instructions. The recorded time describes the scaling of the delay between sending an application scenario setup request (step 1) and receiving an answer from the *IFC* matrix (step 4), which is relevant in case of time-critical requests.

The algorithms' complexity also determines the performance of the framework. The plots in Figures 4 and 5 show the scaling of response time in real-time and CPU time. The time is measured as the input size increases (number of nodes increases), and as the heterogeneity of the input increases (number of areas increases).

In the plot of Figure 4, we keep the heterogeneity as a constant ($AHR = 0$), and we increase the number of nodes. Notice that a third-degree polynomial nicely fits the data, demonstrating time complexity $O(n^3)$.

The plot in Figure 5 shows the scalability of real and CPU time as AHR increases, where $n = 20$. Notice that a second-degree polynomial nicely fits the data, indicating a scaling as $O(n^2)$.

2) INFRASTRUCTURAL PROPERTIES

The percentage of aggregated nodes within the same area indicates the setup cost of any sub-infrastructure. As an example, if the aggregation power percentage is a 100%,

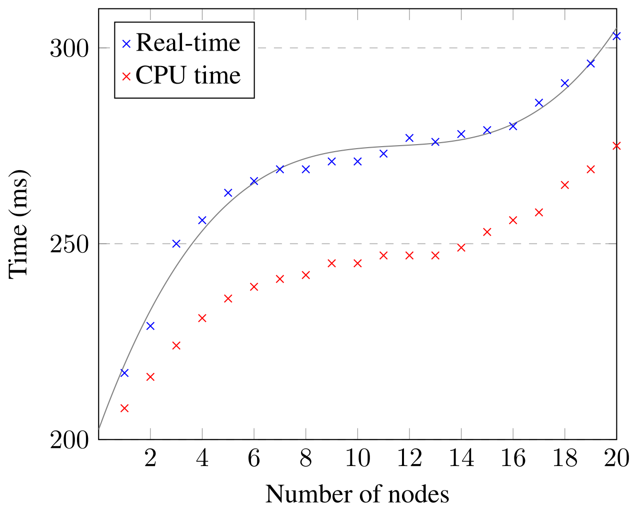


FIGURE 4. The script execution time with respect to the increasing input.

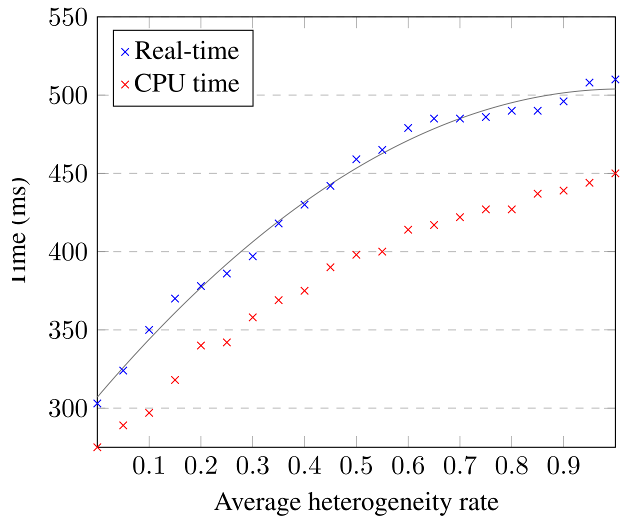


FIGURE 5. The script execution time with respect to increasing AHR.

that means that no extra bridging functions need to be in place before the movement of information between nodes is possible.

The opposite is also true, the lower the aggregation power, the higher the setup cost might be estimated. Moreover, aggregation is directly proportional to the probability of satisfiable application requests. As an example, if the aggregation power is 100% (all nodes belong to the same area) then the probability that supported channels exists (data movement is possible) is 1.

Eq. (16) calculates the number of possible attribute subsets, given that d is the number of all possible distinct attributes, such that:

$$||\wp(A)|| = 2^d - 1. \tag{16}$$

The result of Eq. (17) shows the percentage of nodes being aggregated into distinct areas. The value might differ with different infrastructures. Aggregation power is calculated as follows, where Θ is the set of all areas grouping N:

$$Aggregation = \frac{||\wp(A)|| - ||\Theta|| + 1}{||\wp(A)||} \times 100. \tag{17}$$

The infrastructure also has a property of setup success ratio, where the bridging functions can satisfy all possible missing channels (attributes ϵ_{ij}). The fewer attributes are bridgeable, the higher the failure ratio is. We calculate the success ratio as follows:

$$Success = \frac{||A_\delta||}{||A||}. \tag{18}$$

VI. DISCUSSION

The framework provides clear identification of missing attributes per node and matches it to a bridging function. The available bridging functions directly affect the application requests feasibility and success rate, as defined by the infrastructural properties. The infrastructural properties' measurements rely on the infrastructure itself.

The infrastructure should be ready to bridge any missing attribute to accomplish a complete success rate. Moreover, the success rate also depends on the heterogeneity of the infrastructure, which the aggregation power value reflects. The aggregation power is 100% if the number of areas created is equal to 1, hence all nodes are characterised by the same attribute sets. Subsequently, upgrading the infrastructure improves the success rate of an application request and the aggregation power of areas.

The algorithm of the logic model scales under the complexity of $O(n^3)$. We interpret the algorithm performance as a one time run delay before pushing the setup instructions into the infrastructure level. After that, the complexity and time cost is inversely proportionate to the aggregation power and average intersection rate.

In other words, as the heterogeneity in the infrastructure increases, the time cost increases. The delay follows from the extra steps of the intermediate bridging functions that are associated with a higher heterogeneity and a larger number of areas.

The previously discussed measurements foresee the probability of an application request failure, high setup costs, and delays. Those values are dependant on the infrastructure and can be improved by tailoring the infrastructure resources. The EPI parties can evaluate the infrastructural properties before introducing any application to estimate a future upgrade or to customise the requests accordingly.

Existing health data sharing frameworks focus on primarily security and access control, as shown in Table 1. The EPI framework addresses these two features by considering security and access control attributes. As a result, channels exist to support data movements to nodes that offer the same security functionalities (same area) or more (area superset).

Our proposed framework addresses other features by defining attributes (for example structured data storage) that ensure interoperability and delegating the definition of data sharing rules to the policy management system. Data auditing is also a proposed feature considered by the EPI architecture as shown in Figure 1. The EPI framework adds the EPI logic model to the infrastructural programmability and provides a dynamic and heterogeneous infrastructure. As a result, we contribute to existing frameworks by offering the dynamicity feature.

VII. THE EPI INFRASTRUCTURE

Introducing Network Functions Virtualisation (NFVs) [16] to the EPI infrastructure enables configuring hybrid networks of physical and virtual resources to adapt to different use cases. As discussed before, the programmability of the EPI infrastructure depends on efficiently setting up the bridging functions between areas of nodes. We utilise the NFV services as a tool to provide the bridging functions on top of the initially heterogeneous areas.

Each NFV instance hosted on a virtualised network implements a bridging function. We refer to the set of virtualised network nodes as the NFV infrastructure (NFVI), which is an overlay cloud-like infrastructure on top of the physical one. NFV instances are set up to add missing attributes, such as packet encryption/decryption. To forward and redirect incoming packets to the correct NFV node. Each instance is identified with a unique IP address [17].

We define the series of connected bridging functions as the bridging function chain (BFC), and we assume that the proxy nodes will use a suitable mechanism to perform area-area routing along the BFC. One of the possible technologies that an EPI infrastructure could use is, for example, Segment routing [18]. The routing table is set to enforce packet forwarding along the BFC path. By that applying the abstract set of commands specified by δ (see Eq. (12)).

In Figure 6, we illustrate a typical EPI infrastructure that is layered into physical topology, logical topology, and a control plane. The physical topology layer is the initial topology of the parties' nodes N_{App}. The EPI framework builds the logical topology of areas on a higher layer of Θ_{App}. At that layer, the framework initialises the C matrix by noting supported channels between nodes.

The proxy node hosts the Area-Area packet forwarding to bridge communication between nodes of different areas. The proxy node handles establishing a connection with the NFV instances hosted on the NFVI. In other words, the proxy node's tasks are to 1) Identify the NFV instances required to bridge a channel, 2) Establish a connection with the NFVs, 3) Enable Area-Area routing by enforcing packets redirection along the BFCs. The control plane runs on the highest level and has three components: 1) The NFV management nodes 2) An orchestrator 3) The middleware controller.

Similar to an ETSI-NFV architecture [19], the EPI infrastructure incorporates NFV management nodes and an orchestrator to configure and manage the NFVs lifecycle. The orchestrator also coordinates with the middleware controller to set up the proxy nodes. There is a centralised orchestrator and one NFV management component per proxy node.

The middleware controller performs the following main tasks: 1) Processing the logical topology and assigning

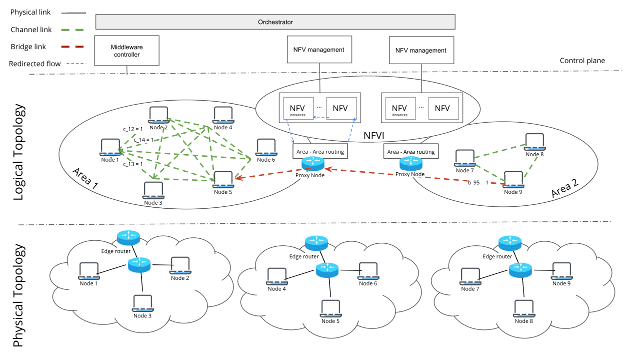


FIGURE 6. The infrastructure topology in any EPI use case.

areas, 2) Configuring the proxy node to enable bridges, and determining the needed BFC of NFVs, 3) Configuring the Area-Area routing tables.

The EPI infrastructure, in a general sense, includes NFV components that are set up to enable the EPI logic model (refer to step 5 in Figure 3). In an EPI use case, we reprogram the EPI infrastructure to adapt by determining the specific BFC of NFVs and effectively communicating with EPI parties. As long as the missing attributes between nodes are bridgeable, and can be virtually provided at a higher layer, then the use case is supported.

VIII. EPI USE CASES

In the EPI project, there are three main use cases to consider. In this section, we introduce these use cases, apply the logic model to determine the IFC, and set up the bridging functions proxies in an attempt to satisfy the requested data movements. Note that in the following use cases, the attributes that we considered are characteristic of the nodes' low-level communication attributes, such as the IP version and data encryption capabilities.

We identify each node as follows $n_i(A_i, IP)$, and the possible attributes are the set A = {Encrypted data (a_1), IPv4 (a_2), IPv6 (a_3)}. If a node has a_1 as an attribute, that means that this node stores maintains, and communicates the data in an encrypted form. A node supports IPv4 and IPv6 addresses if it has attributes a_2 and a_3, respectively. Any incompatibilities between these attributes can result in an unsupported data movement, hence data sharing failure. The three EPI use cases discussed in this section are:

1) Patients to doctor data streaming which is a typical scenario that aims to provide personalized medicine/diagnosis
2) Maintain an EHR and backup storage with multiple data sources
3) ML model sharing to train models on medical data sets

A. PATIENTS-DOCTOR DATA STREAMING

The first EPI use case is built to support patient-doctor data streaming and as a result, providing a personalised diagnosis. Figure 7 illustrates this and shows the framework setup. Nodes n_1, n_2, and n_3 are patients' nodes such that each node simulates the behaviour of a patient's medical data streaming.

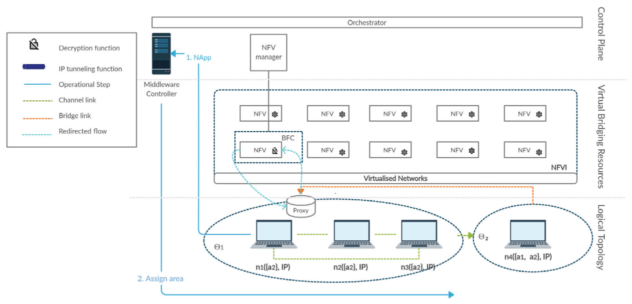


FIGURE 7. The EPI setup for patient to general practitioner data streaming.

These nodes want to send their data to the doctor node hosted on node n_4. Node n_4 has a trained ML model and aims to receive the monitoring data and reply with the patient's personalised diagnosis. In this use case, $N_{App} = \{n_1, n_2, n_3, n_4\}$, and $A_1 = \{a_2\}$ $A_2 = \{a_2\}$ $A_3 = \{a_2\}$ $A_4 = \{a_1, a_2\}$ respectively.

The middleware controller runs the logic model, assigns areas based on these nodes' attributes, deduces IFC, and sets up the area-area proxies. We assume that the policy allows all data movements, and the R matrix entries are initialised all to 1. The bridging functions utilised are the IPv4-to-IPv6 tunnelling and data decryption functions. In case of an unsupported movement between nodes, the middleware controller sets up a proxy that redirects the packets along the BFC to apply the bridging functions. The BFC is identified by the controller to implement the set of commands defined by δ, such that *decrypt* and *tunnel* are two Boolean functions that pre-process the packets to be effectively communicated by nodes.

The NFV applied services in this use case are shown in Figure 7. The incompatibility in this use case is between θ_2 and θ_1, such that $\epsilon_{21} = \{a_1\}$. The proxy is set up to manage area-area routing with BFC: $\delta(c_{21}) = decrypt(c_{21}) = b_{21} = 1$, such that output represents the availability of the bridge between θ_2 and θ_1.

Figure 7 shows the supported channels, and bridgeable channels, and the calculation of IFC is as follows (refer to Eq. (13)):

$$\left(\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} \wedge \begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\0&0&0&1\end{bmatrix}\right) \vee \begin{bmatrix}0&0&0&0\\0&0&0&0\\0&0&0&0\\1&1&1&0\end{bmatrix}$$

As mentioned before, according to this use case data should flow both ways: patients-doctor and doctor-patients. The application requests are expected to be from nodes n_1, n_2, and n_3 to node n_4 and vice versa and the Q matrix shows that. Data movement requests Q are applied once it overlaps with the IFC matrix. That is determined by $\neg(IFC \otimes Q) \wedge Q$ (refer to Table 2), where $\otimes$ represents a logic XOR:

$$\neg\left(IFC \otimes \begin{bmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\\1&1&1&1\end{bmatrix}\right) \wedge \begin{bmatrix}0&0&0&1\\0&0&0&1\\0&0&0&1\\1&1&1&1\end{bmatrix}$$

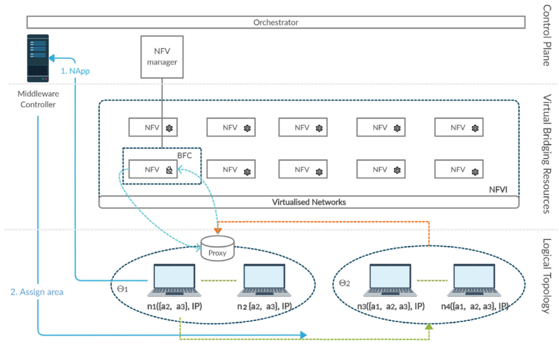


FIGURE 8. The EPI setup for EHR storage and backup use case.

B. EHR AND BACKUP STORAGE

The second EPI use case is presented to maintain an EHR storage node and backup node with multiple data sources. Figure 8 illustrates this use case and the sub-infrastructure setup. In this use case we have medical data sources consisting of two doctors running on n_1 and n_2 nodes. The EHR storage is maintained on n_3, and n_4 acts as a backup. Nodes n_1 and n_2 wants to effectively query (Update/Insert/Get) a patient's medical records from node n_3, and n_3 will update n_4 as a backup node, accordingly. $N_{App} = \{n_1, n_2, n_3, n_4\}$, and $A_1 = \{a_2, a_3\}$ $A_2 = \{a_2, a_3\}$ $A_3 = \{a_1, a_2, a_3\}$ $A_4 = \{a_1, a_2, a_3\}$ respectively.

In this use case the encryption/decryption of data is supported on all nodes. Subsequently, the bridging functions that are required from NFVI are IPv4-to-IPv6/IPv6-to-IPv4 tunneling to support data movements between these incompatible nodes. As an example, $c_{12} = 0$ indicates unsupported data movement, then the proxy enables $\delta(c_{12}) = tunnel(c_{12}) = b_{12} = 1$ through area-area routing.

Figure 8 shows the supported channels (such that all rules' entries are set to 1) and the bridgeable channels. We calculate IFC as follows (Eq. (13)):

$$\left(\begin{bmatrix}1&1&1&1\\1&1&1&1\\1&1&1&1\\1&1&1&1\end{bmatrix} \wedge \begin{bmatrix}1&1&1&1\\1&1&1&1\\0&0&1&1\\0&0&1&1\end{bmatrix}\right) \vee \begin{bmatrix}0&0&0&0\\0&0&0&0\\1&1&0&0\\1&1&0&0\end{bmatrix}$$

In this use case, the expected application requests require the data flow from the medical data sources (might be doctors) nodes n_1 and n_2 to EHR storage node n_3 and vice versa. Moreover, a data flow from node n_3 to the backup node n_4 is expected in the aim to maintain an updated backup storage.

Data movement requests Q are applied once it overlaps with the IFC matrix. That is determined as follows (Table 2):

$$\neg\left(IFC \otimes \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}\right) \wedge \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

C. ML MODEL SHARING

The third use case is proposed to share an ML model and train it over different nodes and that is described in Figure 9. Node n_1 is the ML model owner, and wants to train the model on the different medical data sets stored on n_2, n_3, and n_4 and representing potential hospital parties. Figure 9 shows n_1 initiating the application scenario setup and identifying relevant nodes for this application. In this case, $N_{App} = \{n_1, n_2, n_3, n_4\}$ with $A_1 = \{a_2\}, A_2 = \{a_1, a_3\}, A_3 = \{a_1, a_3\}$, and $A_4 = \{a_1, a_2\}$ respectively.

An example of applying BFC in Figure 9: With the incompatibility between θ_2 and θ_1, $\epsilon_{21} = \{a_1, a_3\}$, the proxy handles packets redirection to create bridges associated with the previously unsupported channel, such that $\delta(c_{21}) = decrypt \wedge tunnel(c_{21})$. Nodes in area θ_1 do not support the encrypted data form that might be sent by nodes in area θ_2. Subsequently, the *decrypt* function, offered by the NFV node, handles decrypting the data to communicate with θ_2 effectively.

Moreover, θ_1 is the area characterised by supporting only IPv4, so the *tunnel* bridging function is offered on top of the physical infrastructure of θ_1 (hosted on the NFV node). The tunnelling function works by encapsulating IPv6 packets into IPv4 packets during transmission so that it is routed normally through IPv4 routers, and vice versa. The chaining of the two bridging functions allows previously unsupported channel $c_{21} = 0$ to be successfully bridged $b_{21} = 1$. After applying these functions, the nodes can effectively receive and process the data communicated.

Figure 9 shows the supported channels and bridgeable channels where we calculate IFC as follows (Eq. (13)):

$$\left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \wedge \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right) \vee \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Data movement requests Q are applied once it overlaps with the IFC matrix:

$$\neg\left(IFC \otimes \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\right) \wedge \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

As a result, the EPI framework provides the previously missing support to effectively share data between EPI parties in

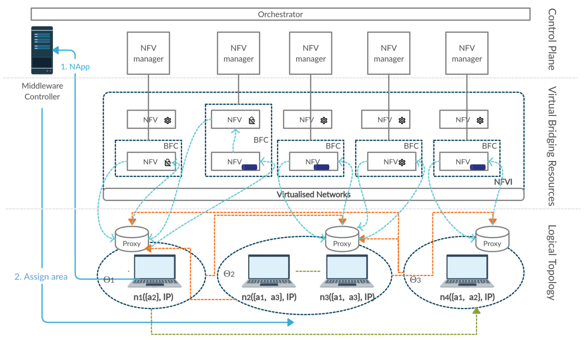


FIGURE 9. The EPI setup for the ML model sharing use case.

different use case scenarios. It is important to highlight that some attributes may never be bridge-able, where $a \notin A_\delta$.

An example of a low-level communication attribute that can never be bridged is the TLS version supported by a specific node. If two nodes in different areas want to communicate, they must have the same attribute value, *i.e.* support the same TLC version. There is no VNF or other mechanism that can bridge this gap.

IX. CONCLUSION

In contrast to current healthcare data sharing frameworks, we propose a dynamic programmable infrastructure that can bridge functionality gaps. Subsequently, the framework enables effective communication and interoperability when sharing data between heterogeneous infrastructures. The framework manages an overlay of attributes services (BFC) to align the data sharing policies with the heterogeneous technical attributes of healthcare domains. The framework assigns nodes into areas, initialises channels, considers collaboration rules, and sets up the bridging functions such that it can control the information flow in a sub-infrastructure.

We defined the logic model and provided the mathematical notations to follow. We simulated the algorithm and designed a prototype to use it. We want to stress that our framework, built to support healthcare use cases, is nonetheless capable of supporting general data sharing use cases. Our work defined the framework component needed to support this.

The measurement formulas of the infrastructure aggregation and success rate are used to quantify the infrastructural properties and compute an estimated failure and cost. We evaluated the performance of the algorithm and estimated that it scales with $O(n^3)$ as an upper limit, which is a one time/application setup delay.

The clustering of nodes into areas is done according to the node's attributes. We will extend on the possible list of characterising attributes and the bridgeablity of each. The attributes are classified into: low-level communication attributes (*e.g.* IP version), security attributes (*e.g.* level of encryption), service attributes (*e.g.* ML model privacy level), storage attributes, and computing attributes. After that, we will append more requirements statements to the logic model to accommodate to customised requirements (*e.g.* $n_g \Rightarrow n_h$, *iff* $A_g \subseteq A_h$ AND $n_g \Rightarrow n_h$, *iff* $a_1 \in A_h$).

The paper proposes the EPI framework and lays out the framework's logic model, and we plan to extend on that with real testbeds experiments. Moreover, this paper explains how the framework and its components will be deployed in a typical infrastructure utilising NFVI and virtual networks. We see potential in the use of overlay networks and network virtual functions as technologies to provide the missing bridges (refer to Section VII and VIII), and we intend on emulating a real EPI infrastructure to evaluate its performance.

REFERENCES

[1] V. Moorthy, A. M. H. Restrepo, and M. P. Preziosi. (2020). *Data Sharing for Novel Coronavirus COVID-19*. [Online]. Available: https://www.who.int/

[2] (2013). *Summary of the HIPAA Security Rule*. HHS.gov. [Online]. Available: https://www.hhs.gov/.html

[3] (2019). *General Data Protection Regulation (GDPR)—Official Legal Text*. General Data Protection Regulation (GDPR). Accessed: Sep. 9, 2020. [Online]. Available: https://gdpr-info.eu/

[4] V. Patel, W. Barker, and E. Siminerio, *Trends in Consumer Access and Use of Electronic Health Information*. Washington, DC, USA: Office of the National Coordinator for Health Information Technology, 2015.

[5] K. Sartipi, K. Kuriakose, and W. Ma, "An infrastructure for secure sharing of medical images between PACS and EHR systems," in *Proc. Conf. Center Adv. Stud. Collaborative Res.*, Toronto, ON, Canada, 2013, pp. 245–259. [Online]. Available: http://dl.acm.org/citation.cfm?id=2555523.2555549

[6] V. Patel, "A framework for secure and decentralized sharing of medical imaging data via blockchain consensus," *Health Informat. J.*, vol. 25, no. 4, pp. 1398–1411, Dec. 2019, doi: 10.1177/1460458218769699.

[7] E. Roelofs, A. Dekker, E. Meldolesi, R. G. P. M. van Stiphout, V. Valentini, and P. Lambin, "International data-sharing for radiotherapy research: An open-source based infrastructure for multicentric clinical data mining," *Radiotherapy Oncol.*, vol. 110, no. 2, pp. 370–374, Feb. 2014, doi: 10.1016/j.radonc.2013.11.001.

[8] D. Thilakanathan, S. Chen, S. Nepal, R. Calvo, and L. Alem, "A platform for secure monitoring and sharing of generic health data in the cloud," *Future Gener. Comput. Syst.*, vol. 35, pp. 102–113, Jun. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13001908

[9] K. N. Griggs, O. Ossipova, C. P. Kohlios, A. N. Baccarini, E. A. Howson, and T. Hayajneh, "Healthcare blockchain system using smart contracts for secure automated remote patient monitoring," *J. Med. Syst.*, vol. 42, no. 7, pp. 1–7, Jul. 2018, doi: 10.1007/s10916-018-0982-x.

[10] Q. Xia, E. B. Sifah, K. O. Asamoah, J. Gao, X. Du, and M. Guizani, "MeDShare: Trust-less medical data sharing among cloud service providers via blockchain," *IEEE Access*, vol. 5, pp. 14757–14767, 2017, doi: 10.1109/access.2017.2730843.

[11] K. Fan, S. Wang, Y. Ren, H. Li, and Y. Yang, "MedBlock: Efficient and secure medical data sharing via blockchain," *J. Med. Syst.*, vol. 42, no. 8, p. 136, Aug. 2018, doi: 10.1007/s10916-018-0993-7.

[12] N. A. Anoop and T. K. Parani, "A real time efficient & secure patient monitoring based on wireless body area networks," in *Proc. Online Int. Conf. Green Eng. Technol. (IC-GET)*, Nov. 2016, pp. 1–5, doi: 10.1109/get.2016.7916814.

[13] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Kumar, R. Sundarasekar, and C. Thota, "A new architecture of Internet of Things and big data ecosystem for secured smart healthcare monitoring and alerting system," *Future Gener. Comput. Syst.*, vol. 82, pp. 375–387, May 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17305149

[14] S. Shakeri, V. Maccatrozzo, L. Veen, R. Bakhshi, L. Gommans, C. de Laat, and P. Grosso, "Modeling and matching digital data marketplace policies," in *Proc. 15th Int. Conf. eSci. (eSci.)*, Sep. 2019, pp. 570–577.

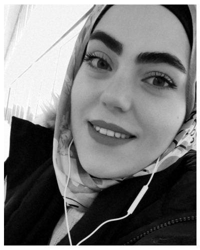
[15] L. Zhang, R. Cushing, L. Gommans, C. De Laat, and P. Grosso, "Modeling of collaboration archetypes in digital market places," *IEEE Access*, vol. 7, pp. 102689–102700, 2019, doi: 10.1109/access.2019.2931762.

[16] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 236–262, 1st Quart., 2016, doi: 10.1109/comst.2015.2477041.

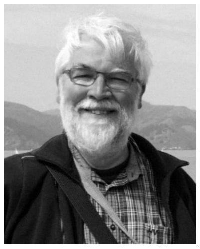
[17] B. Yi, X. Wang, K. Li, S. K. Das, and M. Huang, "A comprehensive survey of network function virtualization," *Comput. Netw.*, vol. 133, pp. 212–262, Mar. 2018.

[18] A. AbdelSalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, "Implementation of virtual network function chaining through segment routing in a Linux-based NFV infrastructure," in *Proc. IEEE Conf. Netw. Softwarization (NetSoft)*, Jul. 2017, pp. 1–5.

[19] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, "NFV: State of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Netw.*, vol. 28, no. 6, pp. 18–26, Nov. 2014, doi: 10.1109/mnet.2014.6963800.



JAMILA ALSAYED KASSEM received the B.Eng. degree in computer engineering from Beirut Arab University, Debbieh, in 2017, and the M.Sc. degree in information and network security from the University of the West of Scotland, U.K., in 2018. She is currently pursuing the Ph.D. degree with the MultiScale Networked Systems, University of Amsterdam. Her research interests include novel network infrastructure, programmable infrastructure, health data sharing, and information security.



CEES DE LAAT (Member, IEEE) is currently the Chair of the System and Network Engineering (SNE) Laboratory, Complex Cyber Infrastructure (CCI) Research Group, Faculty of Science, Informatics Institute, University of Amsterdam. He serves with the Lawrence Berkeley Laboratory Policy Board for ESnet. He is a Cofounder of the Global Lambda Integrated Facility (GLIF), the Founder of GRIDforum.nl, and a Founding Member of CineGrid.org. His group has been a part of A.O. EU projects GN4-2, SWITCH, CYCLONE, ENVRIplus and ENVRI, Geysers, NOVI, NEXTGRID, and EGEE, and national projects DL4LD, SARNET, COMMIT, GIGAport, VL-e, and EPI. He is also a member of the Advisory Board of Internet Society Netherlands and Scientific Technical Advisory Board of SURF Netherlands.

ARIE TAAL received the Ph.D. degree in nuclear science from the University of Delft, in 1989. He is currently a Part-Time Researcher in advanced Internet research with the University of Amsterdam. His research interest includes network's algorithms and architecture.

PAOLA GROSSO (Member, IEEE) is currently an Associate Professor with the University of Amsterdam where she leads the Multiscale Networked Systems Research Group. Her work focuses on the creation of sustainable and secure e-infrastructures, and relying on the provisioning and design of programmable networks. She has an extensive list of publications on the topic and contributes to several national and international projects.