2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)

CareNet: Building Regulation-compliant Home-based Healthcare Services with Software-defined Infrastructure

Peilong Li*, Chen Xu†, Yan Luo*
Department of Electrical and Computer Engineering
University of Massachusetts Lowell
Lowell, MA 01854
*{peilong_li, yan_luo}@uml.edu,
†{chen_xu}@student.uml.edu

Yu Cao
Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854
{yu_cao}@uml.edu,

Jomol Mathew‡, Yunsheng Ma§
‡Department of Information Technologies
§Department of Medicine
University of Massachusetts Medical School
Worcester, MA 01655
{jomol.mathew, yunsheng.ma}@umassmed.edu,

***Abstract*—Healthcare network and computing infrastructure is rapidly changing from closed environments to open environments that incorporate new devices and new application scenarios. Home-based healthcare is such an example of leveraging pervasive sensors and analyzing sensor data (often in real-time) to guide therapy or intervene. In this paper, we address the challenges in regulatory compliance when designing and deploying healthcare applications on a heterogeneous cloud environment. We propose the CareNet framework, consisting of a set of APIs and secure data transmission mechanisms, to facilitate the specification of home-based healthcare services running on the software-defined infrastructure (SDI). This work is a collaboration among computer scientists, medical researchers, healthcare IT and healthcare providers, and its goal is to reduce the gap between the availability of SDI and meeting regulatory compliance in healthcare applications. Our prototype demonstrates the feasibility of the framework and serves as testbed for novel experimental studies of emerging healthcare applications.**

***Index Terms*—Home-based Healthcare; HIPAA Compliance; Software-defined Infrastructure; XaaS; CORD;**

I. Introduction

The advances in information technology greatly accelerates the innovations in healthcare technology recently. In particular, home-based healthcare services such as rehabilitation, respiratory therapy, telemedicine, and so on are being realized due to the availability of low-cost sensors, effective data processing capabilities and advanced networking technologies. The prediction [1] reveals the demands of home-based healthcare market will keep increasing rapidly by at least 5% per year to the year of 2020, thanks to the cost savings and the comfort provided to patients and people in need.

The growing trend of home-based healthcare has introduced new challenges in data collection, transfer and sharing since the patients and their care providers are often geographically distributed. Existing healthcare infrastructures such as the traditional close-environment healthcare and the emerging cloud-based healthcare face vital obstacles as follows. Firstly, traditional healthcare infrastructure, which assumes a closed environment in a single or multiple fixed location, cannot efficiently analyze and share the patient data securely to multiple stakeholders. For example, the sensor data collected on patient from her residence have to be transferred to a remote analytics service or to the doctors' offices for diagnosis. Such data transfer over public networks requires both intensive computing resources and sufficient protection which are not readily available in traditional healthcare. Secondly, the reliability of cloud-based healthcare hinges on the data transmission performance between the end devices and the cloud. Many emerging home-based mission critical healthcare services that require real-time responses and decisions demand the network to be low-latency and high-bandwidth. However, real-world cloud latency ranges from hundreds of milliseconds to a few seconds because of the structure of the Internet. Therefore, cloud by itself is not a feasible solution to home-based healthcare. Thirdly, all patient data related diagnosis and analytics activities should be supported with an infrastructure that is regulation compliant. Patient information must be protected to comply with regulations such as Health Insurance Portability and Accountability Act (HIPAA) and Health Information Technology for Economic and Clinical Health (HITECH).

The emerging software-defined infrastructure (SDI) has shed light on the challenges in existing healthcare infrastructure. While computing resources on cloud platforms are flexible and cost-effective, there are new resources provisioned at the network edges for applications requiring high throughput and low latency. CORD [2] is such an example of edge based computing platform residing in the central office of a telco. Paradrop [3] contains programmable resources in a WiFi router deployed inside a patient's premises. These heterogeneous resources at every part of the network (end point, edge and core) bring both unprecedented opportunities for application design and challenges of performance and compliance verification.

There exists a gap between the availability of emerging SDI

978-1-5090-4722-2/17 $31.00 © 2017 IEEE
DOI 10.1109/CHASE.2017.121

and deploying regulation compliant healthcare services on top of that. Both technical and non-technical people desire tools to assist or even automate the design and verification process. In this paper, we are motivated to achieve the following goals. (1) We propose a healthcare framework called "CareNet" that enables the employment of a hybrid home-edge-core cloud to render high performance and real-time responsiveness for the home-based healthcare services. (2) We sought to propose a secure end-to-end data transmission mechanism and an advanced access control scheme so that every networking transaction on CareNet has to be compliant with the HIPAA technical safeguard. (3) We design a suite of high level Application Programming Interfaces (API) that exploit the underlying SDI resources to help various stakeholders to express their workflow and simplify the management of the healthcare resources without knowing the technical details. This work is intended to initiate the discussion among network researchers, medical researchers, healthcare ITs, patients and clinical staffs.

This paper is organized as follows. Section II provides the background of the work by describing in detail the hybrid cloud architecture and performance and regulatory requirements. Section III and Section IV present the design of CareNet system and the design of CareNet APIs respectively, followed by use case study in Section V. We evaluate the effectiveness of our preliminary CareNet testbed in Section VI and discuss the limitations of the work in Section VII, and conclude the paper in Section VIII.

II. Background

A. Software Defined Infrastructure for Healthcare

Healthcare organizations are gradually adopting cost-effective hybrid cloud services for both administrative tasks and clinical data [4]. For instance, the clinical data could be stored on a private cloud, whereas the equipment inventory data could be hosted on a public cloud. However, as the sensor-rich home-based care environment becomes particularly helpful for caring patients with chronic diseases, the sensor data originates from patients' premises and penetrates across the boundary of the conventional healthcare networks. The sensor data (e.g. video) is to be processed by medical researchers and then the analytical results are consumed by doctors, in real-time in most cases, to guide therapy or intervention. The transfer and processing of the data require much more networking and computing resources than simple medical history data. Such application scenarios are quite new to all the stakeholders, who are required to ensure the performance and compliance of the system hardware and software architecture.

Recently, the technology to exploit abundant computing and networking resources at telco central offices on the network edge, has enabled the feasible development of an interactive home-based healthcare model for care providers. As depicted in Figure 1, a home-based healthcare model consists of three major components: 1) **edge** - at homes, patients' sensor data are collected by various monitoring devices and the sensor data are transmitted and aggregated on a computing device (called "HomeNode") such as an enhanced WiFi router at the premise. At the network edge, telcos provide computing racks and white box switches [2] to support flexible data processing right after the data streams leave patients' homes. It is vital to keep computing resource at the edge of the network to support latency-sensitive applications and services. The edge nodes can also support intensive computation and stream mining, which process the data and reduce the data volume at a very early stage, thus cutting down delays and saving network bandwidth; 2) **hospital private cloud** - only hospital ITs and doctors can access private cloud. The private cloud serves as an enterprise scale data center and can be used to store and process patient medical records. But the private cloud is not a viable option for some care providers such as rural clinics; 3) **public cloud** - both patients and doctors can access the public cloud. The public cloud provides extra richer computation and storage for data analytics, and hosts REST service for mobile and web applications.

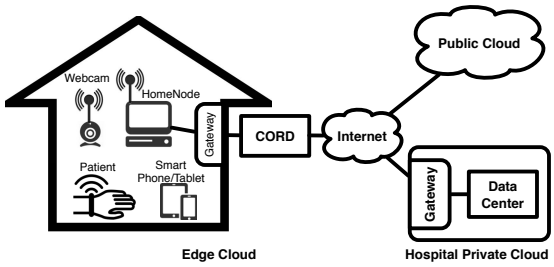

Fig. 1. Home-based Healthcare

The techniques of SDI is fundamental of the proposed home-based infrastructure, thanks to the programmability of network with software-defined networking (SDN) and the feasibility of resource management in cloud with OpenStack. Recently, the novel Everything-as-a-Service Operating System (XOS) emerges to provide a set of abstractions so that application builders can fully leverage the underlying programmable infrastructure. As depicted in Figure 2, XOS defines a coherent framework that consists of both OpenStack and OpenNetwork Operation System (ONOS) for combining SDN, network function virtualization, and cloud services, all running on commodity hardware, to build a cost effective and agile cloud infrastructure.

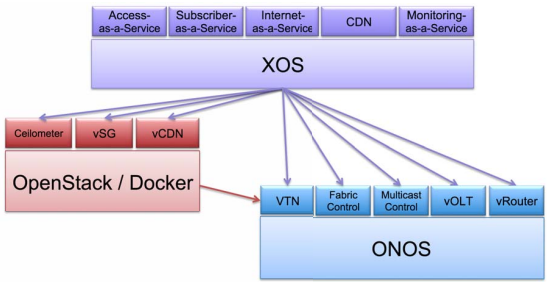

Fig. 2. Relationship between XOS, OpenStack and ONOS [5]

B. Regulatory Compliance Requirements

We hereby attempt to provide an overview of requirements on the healthcare regulatory compliance, with the understanding that healthcare regulations are very complex and broad. We consulted with medical researchers, healthcare ITs and doctors who work with patients on a daily basis, and try to understand the implications on networking technology. We learn that there are three major components in complying with the US HIPAA standards: Administrative, Physical, and Technical [6]. These guidelines stipulate that all medical practices must ensure that all necessary measures are in place while saving, accessing and sharing any electronic medical data to keep patient data secure. Lack of compliance to the HIPAA security standards could lead to large fines and in extreme cases even loss of medical licenses. While the administrative and physical safeguard guidelines pertain mostly for employees' security awareness and training, and facility related access control and security, the technical safeguard regulates the data storage and retrieval and the security of the network, which is addressable with computer techniques. We therefore only focus on the technical safeguard in this paper.

Technical safeguards are becoming increasingly important due to technology advancements in healthcare. Care providers are faced with the challenge of protecting electronic protected health information (ePHI), such as electronic health records, from various internal and external risks. To reduce risks to ePHI, covered entities must implement technical safeguards as good business practices. Table I lists a collection of Technical Safeguard standards and certain implementation specifications, which includes 4 regulation sections: access control, integrity, person/entity authentication, and transmission security. A covered entity may use appropriate security measures that allow it to reasonably implement the standards.

TABLE I
TECHNICAL SAFEGUARD REQUIREMENTS

Standards	Sections	Implementation Specifications
Access Control	§164.312(a)(1)	Unique User Identification (avoid disclosure of user information)
		Emergency Access Procedure: procedures for obtaining necessary ePHI during an emergency (privilege endorsement)
		Encryption and Decryption: a mechanism to encrypt and decrypt ePHI
Integrity	§164.312(c)(1)	Mechanism to Authenticate ePHI
Person/Entity Authentication	§164.312(d)	Implement procedures to verify that a person or entity seeking access to ePHI is the one claimed
Transmission Security	§164.312(e)(1)	Integrity Controls: the security measures to ensure that electronically transmitted ePHI is not improperly modified without detection until disposed of
		Encryption: a mechanism to encrypt ePHI whenever deemed appropriate.

C. Prior Work on Healthcare IT

Employing advanced computer techniques to facilitate the management of healthcare IT has been a hot and concerning topic across ITs, medical researchers, network professionals, care-providers, and the government [7], [8], [9]. Now with the dawning of the age of Cloud and XaaS, it's rather demanding to have collaborative cloud-based platforms and tools for easy, fast, efficient, and regulation-compliant management of healthcare IT.

ASTM International first proposes F2761-09 [7] as standard that defines the patient-centric Integrated Clinical Environment (ICE). F2761-09 intends to improve medical device error resistance and improved patient safety by providing capabilities such as comprehensive data acquisition for the electronic medical records, the integration of devices to enable real-time decision support, safety interlocks, and closed-loop control.

As an open extension to ICE, Plourde et al. [8], [9] recently design an open-source prototype called OpenICE to create a community implementation of an Integrated Clinical Environment. The design of OpenICE framework aims to integrate both healthcare devices and clinical applications to existing Healthcare IT ecosystems. For example, the OpenICE works as a distributed data aggregation and processing system that enables users to convert heterogeneous medical device data from supported devices into a predefined data model, and to exchange that data with demonstration clinical applications on a different machine (or machines).

However, none of these prior arts address three critical emerging issues in home-based healthcare. Firstly, existing arts such as OpenICE are designed for hospital-based clinical applications. However, home-based healthcare involves lots of more emerging issues such as remote monitoring, collaborative data sharing, and user access control, etc. Secondly, as traditional IT infrastructure is outdated by flexible and on-demand cloud services, how should a collaborative healthcare IT framework look like to take advantage of the ubiquitous cloud resources (edge, private, and public, etc.)? Thirdly, since healthcare IT requires regulatory compliance everywhere that involves with patients' data, how can a management framework enforce compliance for the system manager automatically?

With the aforementioned questions, we are motivated to explore the solutions with our proposed CareNet framework in the following sections.

III. SYSTEM DESIGN

To leverage the emerging SDI technologies in a regulation compliant manner, we propose in this section the CareNet, a heterogeneous computing and networking framework for providing effective healthcare to people living in a home setting equipped with advanced and versatile sensors. The goal of this section is to present the high level architecture of CareNet, and describe the key components in this framework. We also outline a comprehensive patient data processing/accessing/sharing mechanism that is part of the CareNet framework to enforce the regulation compliance.

A. The CareNet Framework

A high-level overview of CareNet framework is shown in Figure 3. The major distinctions between the conventional healthcare network and CareNet are the deployment of sensors at patients' premises and the presence of heterogeneous

edge/cloud computing resources at different segments of the system.

We explain the architecture of the system with the running flow of sensor data as follows. Firstly, in this human-centric framework, the healthcare activities are driven by sensor data generated around patients. Abundant body sensors and monitoring devices are installed on patient's body or at patient's home. Then, the heterogeneous sensor data stream is aggregated and preprocessed at an enhanced WiFi router or a small compute system called "HomeNode". The HomeNode runs a daemon process to associate each data stream from the sensors to an isolated application in a docker container for processing. Secondly, the container applications communicate with the CORD edge cloud at telco's central office that is equipped with rich software-defined compute, storage, and network resources. Leveraging such edge computing resources greatly reduces raw data volume that needs to be transferred, and highly reduces the response latency for some time-sensitive applications. The CareNet API, as a higher level abstraction of XOS running on CORD, renders the interface to manage the underlying edge resources. Every CareNet API call must be authenticated with the methods described in the next subsection to ensure regulatory compliance. Thirdly, the preprocessed data will be encrypted and then reach to the hybrid cloud domain via the Internet for more powerful and scalable computing and storage. The hybrid cloud also hosts RESTful service for all parties to access the information from their web or mobile applications. A prototype web-based CareNet application is hosted on our demonstration website [10].

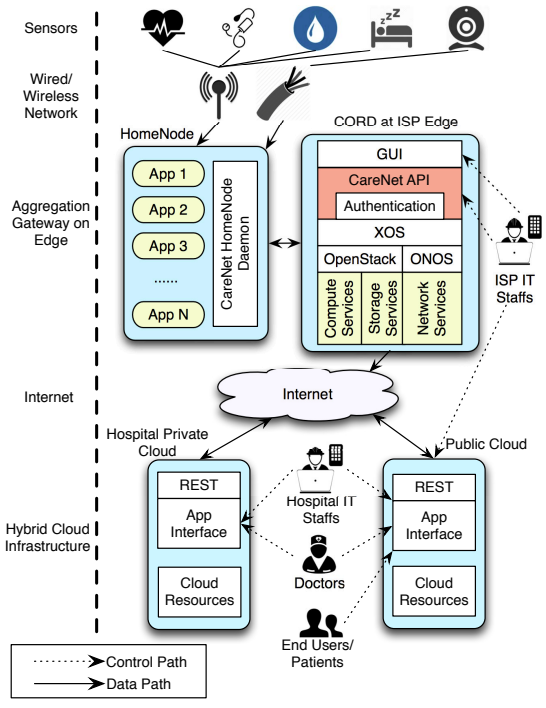

Fig. 3. Architecture of CareNet

B. Patient Data Security

The security of patient-data storage and outsourcing is a major concern of any cloud-based healthcare platform [11]. Existing works [12], [13], [14], [15] in this area have looked into two important data security issues in this area: 1) the security of distributed data storage for patient data across multiple segments of the network, and 2) the fine-grained access control for the collaborative sharing of private patient medical data. However, as regulatory compliance and heterogeneous computing resources are introduced into the proposed infrastructure, we have to consider more emerging issues on data security as follows.

Firstly, how to design a secure end-to-end network framework that consists of all the CareNet components - the HomeNode, the edge cloud and the core cloud, as regulated by HIPAA transmission security (164.312 (e))? We propose the secured networking architecture in Figure 4. From left to right, the solid arrows represent the data flow. Security-sensitive network flows that come from HomeNode will pass through an Internet Protocol Security (IPsec) protected network link to avoid wiretapping, while other flows from HomeNode such as entertainment streaming, gaming, etc. will stay in the non-protected path to reduce the encryption/decryption cost. Such network data filtering is controlled by the SDN controller residing at telco's central office. The layer 3 IPsec security method can potentially be replaced by layer 2 solutions such as Media Access Control Security (MACsec) for less latency overhead but higher CAPEX. Once data is processed after arriving at telco's CORD cloud, it will be encrypted before going into the untrusted network domain. For clients to access the processed data, they need to acquire a proper security key managed by the authentication system that comes with the CareNet API. It is worth noticing that data only flows from trusted domain to untrusted domain in one direction so that patient information won't be tampered by malicious information. From right to left on Figure 4, the dotted arrows elaborate the management flow. Commands and requests from clients' side have to pass through a double authentication scheme before entering the trusted domain. First of all, to establish the connection between clients and CareNet server at CORD over the Transport Layer Security (TSL) tunnel, clients need to acquire a proper certificate such as a SSH certificate. Second of all, requests made by calling the CareNet API will need to be authenticated. The authentication process will be elaborated in detail in the next point. The double authentication scheme ensures both network connection and access control are secure.

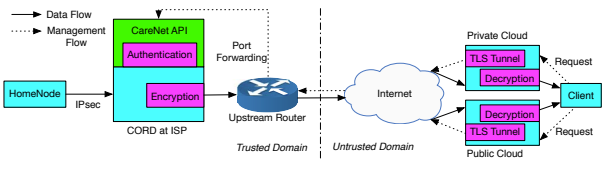

Fig. 4. The Transmission Security Compliant Network Framework

Secondly, HIPAA regulates another three important standards in technical safeguard: access control (164.312 (a)), integrity (164.312 (c)), and person or entity authentication (164.312 (d)), which require the integrity and protection of ePHI under the agreement with multiple parties. Specifically, patients have the right to delegate permissions to different data consumers and the permissions are subject to change in different situations such as time-out and security key revocation. Therefore, based on the idea of dividing the system into public/private domains [14], we propose the data access control system in Figure 5.

The system involves different parties as the participant: e.g., the data owner, who generates the data and controls the authorized user list and the associated attributes; authorized users, who access the health information based on access policy; and public users, who can access the patient data upon the attribute agreement. When a user registers with personal details and a password, the user is provided with a Global Identifier (GID), which uniquely identifies the user in the system. In the private domain, the users can then login using the GID and password combinations by which the users are authenticated. The authenticated users can send requests to data owner to request the access to the information. If the users are not granted with the access, they can only view the data based on the roles the user plays in the system. For the users come under public domain, they should firstly ① request the data. If ② the access privilege of the user satisfies the access policy, then the ④ Attribute Authority (AA) generates the key by ③ replacing the GID with tickets from the Ticket Generator. Thereafter, the generated private key and the encrypted data are shown to the user as they provide the key password. The data is then ⑤ decrypted using the private key and displayed.

There are 3 major advantages of the proposed data access control mechanism which differ from the traditional role-based access control (RBAC) [16]: 1) the authentication + ticket generation mechanism enhances the identity authentication process. The usage of ticket sessions instead of GID reduces the risks of GID leak, which protects patient information from being hacked. This mechanism enables the authentication system to be collusion resistant against users, attribute authorities and between user and authorities; 2) user revocation can be easily conducted by updating authorized user list and attribute storage in both private and public domain. Data owner therefore retains the use and disclosure rights of the protected PHI to other users as required by HIPAA privacy practice; 3) the proposed data protection model matches the CareNet API abstraction which is going to be discussed in Section IV, so that one can effectively express the data access requirements to authorized list and attribute storage by using the API.

IV. The CareNet API

To facilitate the usability of CareNet framework, we need well-defined and high-level APIs whereby both technical people like network operators and application developers, and non-technical people such as care providers and patients can express their requirements on data collection, sharing and processing. In this section, we first introduce the abstraction model in CareNet framework, and then present the APIs and explain their usage in details.

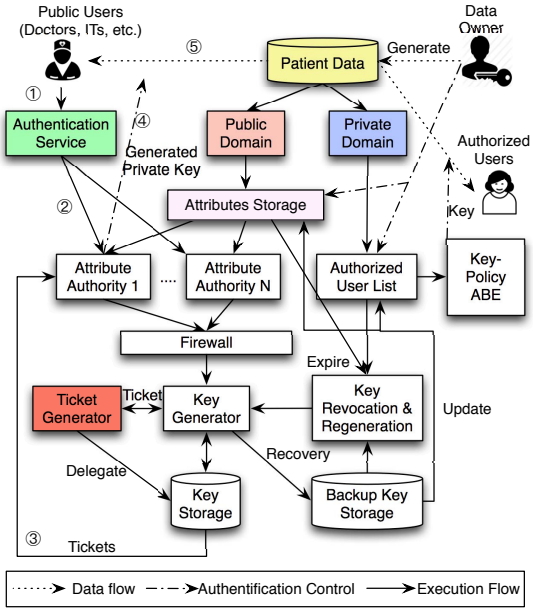

Fig. 5. Patient-Data Access Control

A. CareNet Abstraction

The high-level abstraction of CareNet framework aims to explain the major roles of objects and how they may interact with each other to express the workflow. The overall abstraction is illustrated in Figure 6, and we explain the meaning of each component as follows.

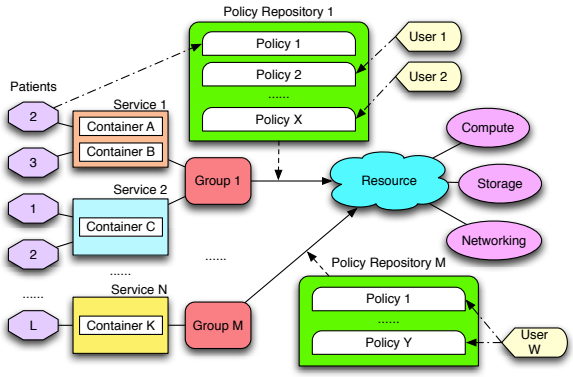

Fig. 6. API Abstraction

- **Abstraction 1 - Patients**: For a human-centric infrastructure, the fundamental elements in the abstraction are the patients that can benefit from various services. To map the many-to-many relationship between patients and care-providers, our proposed abstraction provides the flexibility that each patient can access different services,

and different patients (e.g. family members) can access the same service. Patients are the data sources in our model, and they have the ultimate authority to grant access policies to various data users.

- **Abstraction 2 - Services**: The "Services" presents an abstraction for the applications and the associated daemon processes that run across the CareNet framework. As the design of regulation compliant IoT applications requires data privacy preservation at endpoint, we employ Docker containerization technique to isolate IoT applications into a self-contained service. Each service consists of one or multiple running application(s), data exchange method among the applications, communication interfaces with the CareNet daemon, and the generated data stream. When initiated by the CareNet API, each service is given a unique service name and ID.
- **Abstraction 3 - Groups**: The API abstraction introduces the "Groups" concept that represents a collection of services with the associated patients and efficiently describes the service-service, patient-service interrelationship properties. Services in the same group have to follow the same policy to be processed and to access the resources. The "Group" abstraction renders a clean and effective generalization to map the high-level requirements to the underlying constructs in XOS framework, and facilitates the scalable design of patient-side applications.
- **Abstraction 4 - Resources**: Since the CareNet infrastructure consists of both edge and core cloud computing resources, we can treat CareNet as a resource-rich platform where each service group can maintain a configuration of its accessible resources. To be specific, resources in the CareNet infrastructure include micro applications and services that build upon the underlying virtualized hardware. For example, we can opt for compute pools with various CPU models/numbers and memory sizes, block (Cinder) or object (Swift) data storage, and high/low bandwidth high/low latency network paths with various service chains, etc.
- **Abstraction 5 - Users**: "Users" abstracts the parties that are involved in the CareNet infrastructure management. Users can be telco ITs, doctors, nurses, hospital ITs, patients, patient's relatives, etc., and they are in charge of designing various policies on demand, such as authentication policies, and resource assignment policies for services, etc.. It's worth noting that users have different levels of priority, which will determine the priorities of the policies written by users. For example, patients has the root priority of defining data access control by default and (s)he must assign the attribute authority as described in subsection .
- **Abstraction 6 - Policy**: "Policy" defines the CareNet infrastructure management behavior - it directs different service groups to access different resources by the policy defined configuration. Different users design their collection of policies for a service group to serve for their purposes. For example, patients has the right to specify the authorized user list and the attribute authority as required by HIPPA regulation. Application developers need to declare what services for the patient data require low network latency for effectiveness. ITs are responsible for defining the optimal network configuration for the overall system.

  Policies offer a clean method to describe how service groups utilize resources in non-technical terms. It's important to notice that policies are designed to be reusable. Once a user creates a policy for a certain group, (s)he can reuse it repetitively for other groups. More efficiently, users other than the patients can share their policies to others in the CareNet community, since only patients' policies involve HIPPA compliance concern. The reusability highly reduces the labor to update and rewrite policies, thus improves portability, accuracy, and agility.
- **Abstraction 7 - Policy Repository**: The design of "Policy Repository" contributes another charm in our API abstraction. Repository allows different policies to co-exist and work together based on different roles of the policy writer. If still using the same example in policy for instance, three parities - patients, doctors, and ITs can design policies from different aspects to work for the same service group. This abstraction enables a highly collaborative management fashion so that all parties in CareNet infrastructure can participate to enforce regulatory compliance and improve system efficiency.

B. CareNet APIs

We propose a list of APIs to facilitate the management of CareNet with the aforementioned abstractions. The APIs fall into three categories: 1) **service management** - in CareNet framework, services created for the IoT endpoint devices must be containerized to maintain service isolation. A set of APIs are used to create and terminate containerized services. Moreover, clinic or hospital trusted third party application can be installed/removed with the agreement between patients and the care-providers; 2) **performance related management** - the network flows from different services have different performance requirements determined with the knowledge from both doctors and technical ITs. Some example requirements are: health-critical traffic (low latency requirement) versus regular bio-sensing traffic (no latency requirement); block storage for databases, file-exchange, etc. versus object storage for video processing and data backup, etc.; and high versus low performance computing resources etc.; and 3) **policy enforcement**: services in the same group have to follow the same policies in the policy repository. For example, to write the access control policy for Service A, patients need to first define the authorized user list, and assign different attribute authorities to different users. Then doctors will specify what kind of requirement they need for this service in order to guarantee the timeliness and effectiveness of the service results. ITs then need to look at the technical requirements of the service so that the service runs smoothly in the system. All the policies will finally merge

together in the same repository to work collaboratively with different levels of priority.

The detailed API definition and description are depicted in Table II and we demonstrate a concrete use case by leveraging the APIs in Section V. The design of the APIs renders a unified interface for care providers to manage the proposed CareNet framework and promotes the flexibility of policy specification and compliance enforcement.

C. CORD Configuration with CareNet APIs

We design an automatic CareNet API conversion mechanism to help translate commands written with CareNet APIs into CORD hardware configurations. As demonstrated in Figure 7, the CareNet system allows users to first specify their requirements through the CareNet APIs or use web/app graphic user interfaces that are built upon the CareNet APIs. We call this step the requirement submission. Then the user requirement submission that are written with CareNet API will be fed into our designed API parser. The API parser applies regular expression (RegEx) technique to extract the keywords from requirement submission and find the argument domains within each API function to generate a JSON-formatted intermediate representation (IR). After obtaining the JSON IR, we use a translator to map the key-value pairs in IR to a Topology and Orchestration Specification for Cloud Applications (TOSCA) [17] formatted configuration file. Since TOSCA file is used as the interface configuration to the XOS system, our mapped TOSCA configuration can then eventually configure the CORD hardware.

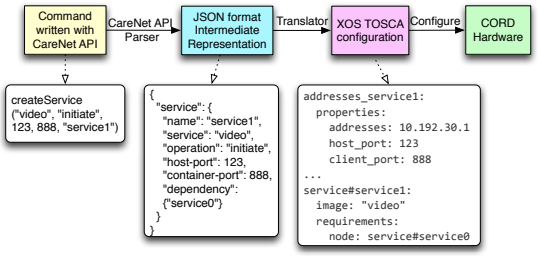

Fig. 7. Configure CORD Hardware with CareNet APIs

V. Case Study

The CareNet framework and the APIs can address several research problems of strong clinical importance and urgency. These problems arise from real-world scenarios where the patient risks can be identified and the timely intervention and interaction between care providers and patients will facilitate speedy recovery. We have identified an essential real-world home-based stroke recovery use case that can leverage the proposed work. We present for the use case the significance of employing CareNet framework, and the simplicity and agility of regulation compliant management with CareNet APIs.

Use Case: Kinect-based Real-time In-Home Stroke Rehabilitation

While it is a general consensus that post-stroke rehabilitation can substantially help people achieve the best possible long-term outcome, the economic cost associated with in-patient post-stroke rehabilitation could be devastating for many stroke victims. To help the stroke survivor to reduce the cost without sacrificing healthcare quality, home rehabilitation has emerged as a valuable supplement to high cost outpatient facilities and/or nursing facilities. In addition to cost saving, home rehabilitation offers great flexibility, which allows the patients to tailor their rehabilitation program and to follow their own schedules.

One of the promising solutions is to develop a new in-home stroke rehabilitation system using new gaming consoles, such as the Microsoft Xbox Kinect. By capturing raw data from the Xbox Kinect sensors, we can track skeletal positions of the patient as they attempt physical rehabilitation exercises, evaluate how the patient is accomplishing the exercises, and score them accordingly. However, there are two major concerns for the care providers to render HIPAA/HITECH compliant interactive healthcare. (1) **Security**: because of the limited computation power of a clinic/hospital, the patient's video stream need to be transmitted over a public network and then to a computing platform (e.g. private hospital cloud or public cloud) for processing. Therefore, patient's data must be encrypted at network edge. In addition, before reaching the computing facility, care providers have to ensure that electronically transmitted patient data is not improperly modified as required by the HIPPA "integrity control". (2) **Performance**: a typical Kinect user scenario has a data generation rate at 300 Mbps (640x480 32-bit RGB images are generated at a 30 frames per second (fps)). Therefore, significant bandwidth is required to transfer such a large volume of video stream. In practice, patients also have to wear body sensors, such as a smart watch to collect the accelerometer data and life characteristics (e.g., heartbeat, blood pressure). The data from body sensors are usually given low latency network priority over the Kinect video stream since bio-sensor data is often utilized to issue health-critical warning events, such as fall detection.

There are 3 major steps that leverage the proposed framework and APIs to express the use case workflow. The first step is the creation of services for different IoT sources. As suggested by regulatory compliance, "security at endpoint" is the key resort to build a secure IoT environment. Therefore, care providers need to create logically isolated services for each endpoint device. Besides, the service creation API permits the service dependency, which greatly facilitates the form of service chaining. The second step is to write different policies for different services, as the allocated resources for each service must meet the performance requirements by the care-providers. Thirdly, CareNet APIs offer the capability to validate the data integrity at any time of the data processing and issue early warnings for data discrepancy.

We demonstrate the pseudo code as follows.

```
/*
 * Psuedo Code Demo For Stroke Rehabilitation
 */
```

TABLE II
API SPECIFICATION

No.	API	Explanation
1	bool createService (char* serviceTemplate, char* cmd, unsigned int hostPort, unsigned int containerPort, char* serviceName, DependentService deplist)	Spawns a container to run a specific application from the CORD service template named "serviceTemplate". DependentService is a data structure that list all existing dependent services for the current service. The function returns true if the container was successfully created, otherwise the function returns false.
2	bool removeService (char* serviceName, unsigned int delay)	Removes the service with the given service name. Waits up to delay seconds before sending SIGKILL signal to container. The function returns true if the container was successfully removed, otherwise the function returns false.
3	unsigned int createPatient (PatientInfo patient ServiceList serlist)	Creates a patient with the provided patient information and the patient's associated services. PatientInfo is a JSON-formated key-value data structure. ServiceList is a list of service names. Returns the patients unique ID.
4	bool removePatient (unsigned int patID)	Removes a patient with the provided patient ID. Returns boolean value indication success or fail.
5	unsigned int createRes (ResourceList rlist)	Creates an accessible list of resources for certain service groups on the provided infrastructure. ResourceList is a JSON-formated key-value data structure. Returns the resource list ID.
6	bool removeRes (ResourceList rlist, unsigned int resID)	Removes a sub-list of resources rlist from the original resource list with ID=resID. Returns boolean indicator.
7	unsigned int createUser (UserInfo user)	Creates a user by using the provided user information. Returns a unique user ID.
8	bool removeUser (unsigned int uID)	Removes a user by using the provided user ID. Returns boolean indicator.
9	unsigned int createPolicy (Policy p)	Policy is a JSON-formatted data structure that contains multiple key-value fields. Some important key fields include "resource", "authentication_list", "attribute_authority", etc. This function creates a new policy with the field of data indicated in Policy "p". Returns the policy ID.
10	unsigned int createRepo (PolicyList polist)	Creates a new policy repository with a list of policy IDs. Returns the repository ID.
11	unsigned int createGroup (ServiceList serlist, unsigned int repoID)	Creates a service group with the provided service list and a policy repository ID.
12	void addServiceToGroup (char* serviceName, unsigned int gID)	Add an existing service to a service group with ID=gID.
13	void rmServiceFromGroup (char* serviceName, unsigned int gID)	Remove an existing service from a service group with ID=gID.
14	bool validateService (unsigned int gID, char* key)	Used for services that need to talk to public cloud. Validate the given service group ID to check if encrypted data has been modified. Returns true if data is safe.

```
// Step 1: Create the encryption, Kinect and watch service,
// add service dependency.
createService ("AES-Template", "initiate", hostPort1,
            containerPort1, "Encryption", NULL);
createService ("Kinect-Template", "initiate", hostPort2,
            containerPort2, "Kinect", "Encryption");
createService ("Watch-Template", "initiate", hostPort3,
            containerPort3, "Watch", "Encryption");

// Step 2: Allocate system resources to the services; create
// policy and policy repository; create service group.
resKinect = createRes (..., "max_delay_ms": "auto",
                    "bandwidth": 300);
resWatch = createRes (..., "max_delay_ms": 3,
                    "bandwidth": "auto");
resEncrypt = createRes (..., "max_delay_ms": "auto",
                    "bandwidth": "auto");

policyKinect = createPolicy ("resource": resKinect);
policyWatch = createPolicy ("resource": resWatch);
policyEncrypt = createPolicy ("resource": resEncrypt);
repoRehab = createRepo ({policyKinect, policyWatch,
                    policyEncrypt});

groupKinect = createGroup ({"Kinect", "Encryption"},
     userlist);
groupWatch = createGroup ({"Watch", "Encryption"},
     userlist);

// Step 3: Validate the integrity of data before
// reaching the computing facility.
if ( ! validateService (groupKinect, key) )
   issueWarning ("Kinect data discrepancy\n");
if ( ! validateService (groupWatch, key) )
   issueWarning ("Watch data discrepancy\n");
```

VI. EVALUATION

In this section, we demonstrate the implementation of the CareNet prototype and the effectiveness of the design with preliminary experiment results for the use case.

While working towards building a self-operated CORD testbed, we are currently testing our prototype on CloudLab [18], which offers transparent control and visibility of the cloud down to the bare metal. As demonstrated in Figure 8, we use virtual machines to emulate the HomeNode (for patients), provision the CORD (for networking and edge computing), and design the interconnection in between the HomeNode and CORD. The whole system is then connected to the Internet, so that the system can access the untrusted domain, i.e. Amazon AWS public cloud, for intensive computation. The specifications of hardware, virtual machines, and software versions are listed in Table III for reference.

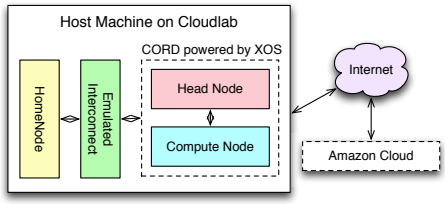

Fig. 8. Experiment Platform

TABLE III
TESTBED SPECIFICATIONS

Item	Specifications
Host CPU	2 x Intel E5-2630 v3 8-core CPUs at 2.40 GHz (Haswell w/ EM64T)
Host Memory	128GB ECC Memory (8x 16 GB DDR4 1866 MHz dual rank RDIMMs)
VM CPU	6-core QEMU Virtual CPU version 2.0.0 processors at 2.4 GHz
VM Memory	16 GB
VM Disk	256 GB
Host OS	Ubuntu 14.04.4
XOS version	branch "cord-1.0"
CORD version	branch "cord-1.0"
ONOS version	1.8.0

As described in the previous use case, we want to demonstrate how to configure the service chain with the two services

"Kinect"+"Encryption" on our CareNet framework in Figure 9. Users firstly specify their requirements by using CareNet APIs as shown in the use case. Then the requirement submission is fed into the conversion tool in Figure 7 to generate the TOSCA configuration file. The following code snippet shows a portion of the converted TOSCA configuration file that is used to configure the CORD hardware. After the service configuration, we then need to call the XOS APIs to create services, slices, and instances etc. as described in the XOS literature [5].

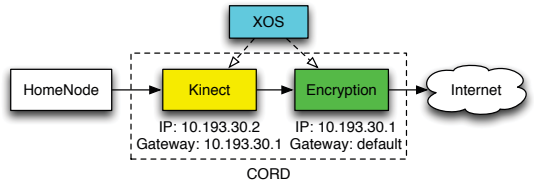

Fig. 9. An Example of SDI Framework

```
/*
 * Configuration file snippet for CORD Services
 */

// Part of s1.config
addresses_kinect:
  type: tosca.nodes.AddressPool
  properties:
    addresses: 10.193.30.1/25
    gateway_ip: 10.193.30.1
    gateway_mac: 02:42:0a:a8:00:01

// Create the Kinect service
// Add service dependency to Encryption service
service#kinect:
  ...
  requirements:
    node: service#encryption
    relationship: tosca.relationships.TenantOfService
  ...

// Part of s2.config
// Create the Encryption service
// Add service dependency to Kinect Service
service#service2:
  ...
  requirements:
    - addresses_kinect:
       node: addresses_kinect
       relationship: tosca.relationships.ProvidesAddresses
...
```

To test the service performance on CORD, we bring up another two services - Virtual Subscriber Gateway (vSG), and the official "Example Service," offered by CORD community. For all three services, we initiate user requests from HomeNode to CORD to generate the network traffic, and test their performance.

The vSG service functions as a Consumer Premises Equipment (CPE) that runs a bundle of subscriber-selected functions such as access restriction, access diagnosis, firewall and bandwidth metering, etc. The vSG service resembles L4-7 network service chaining that may be required by ITs who operate on both ISP edge and hospital private cloud. The Example Service works as a simple HTTP server - it responds with human-readable strings upon user requests. We employ Example Service to resemble the wearable watch service in the use case, because a doctor need to request for the real-time patient bio-metrics and life-critical warning signals through the HTTP protocol. The Video Processing Service is deployed to encrypt the video stream in use case using Advanced Encryption Standard (AES) [19]. We employ Video Processing service to protect the patient's identity especially when the information is transmitting on the Internet.

We measure three performance metrics of the 3 tested services - network round trip time (RTT), service network bandwidth allocation, and the service delay. RTT indicates the network link delay of the service environment; bandwidth allocation stands for the network bandwidth utilization of a service; and service delay denotes the computation time for the service application. As shown in Figure 10, Example Service only endures less than 2 ms of RTT, while retains 5 times more network bandwidth than the other two services since Example service is given low network latency links and we do not specify the bandwidth requirements. The vSG service observes the lowest service delay because it is less compute intensive. Since we do not specify the network speed and bandwidth for vSG, the system will automatically choose the most available resources for it. The Video Processing Service causes 32x more traffic delay than Example Service because of the computation complexity and has almost the same bandwidth as vSG. Although Video Processing Service has a relatively higher service delay comparing with the other two services, 7.81 ms is still considered as very low comparing with the popular cloud applications running with hundreds and thousands of milliseconds. The service delay of the chained Video Processing Service can be divided to two parts: 1) transmission/processing delay around 8.708 milliseconds; and 2) encryption delay around 0.0015 milliseconds. It's worth noting that the service startup time (the time for a service to get on-board) is about a few minutes, because a service needs download the required dependencies from the Internet and configures the VM images. Luckily, the service startup time is a one-time investment, therefore should not affect the user experience.

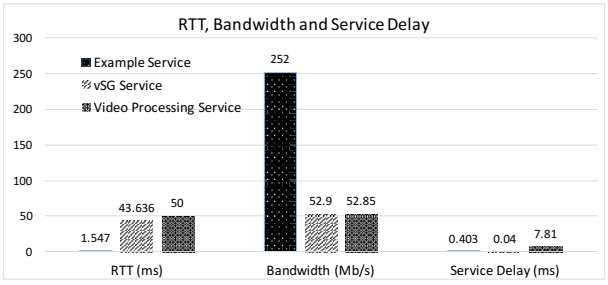

Fig. 10. RTT, Bandwidth, and Service Delay Measurement Results

VII. Discussion

We would like to point out some limitations of this work and a few future directions. First, the proposed abstraction

and APIs are not a comprehensive coverage of the regulatory requirements. As the healthcare related laws are complex, it is extremely difficult to express all requirements of a healthcare application especially when persons such as healthcare providers are not proficient with technologies in computing and networking. There should be a close collaboration among networking researchers, medical researchers and healthcare and clinic personnel, who together can refine the APIs and learn from using the APIs in real clinical settings. Second, the mapping of the APIs to underlying SDI is still an undergoing work. We are able to identify some critical API functions that are sophisticated enough to express the workflow in some known use cases. However, the list of APIs tend to grow larger while the research is on the way of addressing more comprehensive real-world clinical use cases. There exist a good number of related research projects on programming languages to express the performance and other attributes of a network and applications [20], [21], [22]. Yet, these prior works are not the best fit in describing the needs in the SDI environment. In contrast, we focus primarily on the abstraction of the resource and control requirements for the abundant home-edge-cloud resources. We see a good synergy between our effort and the prior works, and we plan to explore a more modularized compiler design in the near future.

VIII. Conclusion

In this work we aim to address the challenges in regulatory compliance of healthcare applications on emerging software defined infrastructure. As the network architecture and computing paradigm have changed significantly, we propose a holistic framework called CareNet for healthcare IT to support the specification of regulation-compliant data sharing/processing requirements, and further the mapping of such requirements to heterogeneous compute/network resources available at different segments of the data path spanning from homes to cloud. Our initial prototype and experimental results have demonstrated the feasibility and performance. Going beyond emulation experiments, our future work will focus on implementing the case study in a real home environment that is connected to an operational CORD deployment and cloud platforms.

References

[1] G. V. R. Inc., "Home healthcare market analysis by product (therapeutic, diagnostic equipment, mobility assist, diabetes monitor, intravenous pumps, holter monitors, heart rate meters, wheel chairs), by service (rehabilitation services, unskilled home care, respiratory therapy, infusion therapy, telemetry) and segment forecast to 2020," Sep. 2014. [Online]. Available: http://www.grandviewresearch.com/industry-analysis/home-healthcare-industry

[2] L. Peterson, "Cord: Central office re-architected as a datacenter," Nov. 2015. [Online]. Available: http://sdn.ieee.org/newsletter/november-2015/cord-central-office-re-architected-as-a-datacenter

[3] "Paradrop official website," 2015. [Online]. Available: https://www.paradrop.io/

[4] L. Columbus, "83% of healthcare organizations are using cloud-based apps today," Jul. 2014. [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2014/07/17/83-of-healthcare-organizations-are-using-cloud-based-apps-today

[5] "XOS: Service Orchestration for CORD," pp. 1–11, Jun. 2015.

[6] HHS, "Summary of the hipaa security rule," 2017. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/security/laws-regulations/

[7] ASTM-F2761-09(2013), "Medical devices and medical systems - essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ice) - part 1: General requirements and conceptual model," Sept 2009. [Online]. Available: http://www.astm.org/cgi-bin/resolver.cgi?F2761-09(2013)

[8] "Openice: An open-source integrated clinical environment," 2015. [Online]. Available: https://www.openice.info/

[9] J. Plourde, D. Arney, and J. M. Goldman, "Openice: An open, interoperable platform for medical cyber-physical systems," in *ICCPS*, 2014.

[10] C. Xu, "Carenet web-based demostration application," 2017. [Online]. Available: https://acanets.github.io/index.html

[11] H. A. K. Khattak, H. Abbass, A. Naeem, K. Saleem, and W. Iqbal, "Security concerns of cloud-based healthcare systems: A perspective of moving from single-cloud to a multi-cloud infrastructure," in *2015 17th International Conference on E-health Networking, Application Services (HealthCom)*, Oct 2015, pp. 61–67.

[12] D. B. Hoang and L. Chen, "Mobile cloud for assistive healthcare (mocash)," in *Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific*, Dec 2010, pp. 325–332.

[13] J. Wan, C. Zou, S. Ullah, C. F. Lai, M. Zhou, and X. Wang, "Cloud-enabled wireless body area networks for pervasive healthcare," *IEEE Network*, vol. 27, no. 5, pp. 56–61, September 2013.

[14] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 1, pp. 131–143, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2012.97

[15] A. Abbas and S. U. Khan, "A review on the state-of-the-art privacy-preserving approaches in the e-health clouds," *IEEE Journal of Biomedical and Health Informatics*, vol. 18, no. 4, pp. 1431–1441, July 2014.

[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, Feb 1996.

[17] O. A. open standards for the information society), "Tosca simple profile in yaml version 1.0," 2016. [Online]. Available: http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/cs01/TOSCA-Simple-Profile-YAML-v1.0-cs01.pdf

[18] "Cloudlab: Build your own cloud," 2014. [Online]. Available: https://www.cloudlab.us/

[19] "Advanced encryption standard." [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[20] A. D. Ferguson, A. Guha, J. Place, R. Fonseca, and S. Krishnamurthi, "Participatory networking," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, ser. Hot-ICE'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 2–2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2228283.2228286

[21] N. Foster, M. J. Freedman, R. Harrison, J. Rexford, M. L. Meola, and D. Walker, "Frenetic: A high-level language for openflow networks," in *Proceedings of the Workshop on Programmable Routers for Extensible Services of Tomorrow*, ser. PRESTO '10. New York, NY, USA: ACM, 2010, pp. 6:1–6:6. [Online]. Available: http://doi.acm.org/10.1145/1921151.1921160

[22] R. Soulé, S. Basu, P. J. Marandi, F. Pedone, R. Kleinberg, E. G. Sirer, and N. Foster, "Merlin: A language for provisioning network resources," in *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '14. New York, NY, USA: ACM, 2014, pp. 213–226. [Online]. Available: http://doi.acm.org/10.1145/2674005.2674989