Article

Named Data Networking Based Disaster Response Support System over Edge Computing Infrastructure †

Minh-Ngoc Tran and Younghan Kim *

School of Electronic Engineering, Soongsil University, Seoul 06978, Korea; mipearlska1307@dcn.ssu.ac.kr
* Correspondence: younghak@ssu.ac.kr; Tel.: +82-02-820-0841

† This paper is the extended version of the conference paper: Tran, M.N.; Kim, Y. NDN-based Emergency Communication over Edge Computing Infrastructure. In Proceedings of the 11th International Conference on ICT Convergence, Jeju Island, Korea, 21–23 October 2020.

Abstract: After a disaster happens, effective communication and information sharing between emergency response team members play a crucial role in a successful disaster response phase. With dedicated roles and missions are assigned to responders, role-based communication is a pivotal feature that an emergency communication network needs to support. Previous works have shown that Named Data Networking (NDN) has many advantages over traditional IP-based networks in providing this feature. However, these studies are only simulation-based. To apply NDN in disaster scenarios, real implementation of a deployment architecture over existing infrastructure during the disaster should be considered. Not only should it ensure efficient emergency communication, but the architecture should deal with other disaster-related challenges such as responder mobility, intermittent network, and replacement possibility due to disaster damage. In this paper, we designed and implemented an NDN-based disaster response support system over Edge Computing infrastructure with KubeEdge as the chosen edge platform to solve the above issues. Our proof-of-concept system performance shows that the architecture achieved efficient role-based communication support, fast mobility handover duration, quick network convergence time in case of node replacement, and loss-free information exchange between responders and the management center on the cloud.

Keywords: named data networking; edge computing; disaster response; emergency communication; KubeEdge

check for updates

Citation: Tran, M.-N.; Kim, Y. Named Data Networking Based Disaster Response Support System over Edge Computing Infrastructure. *Electronics* **2021**, *10*, 335. https://doi.org/10.3390/electronics10030335

Academic Editor: Khaled Elleithy
Received: 17 December 2020
Accepted: 27 January 2021
Published: 1 February 2021

1. Introduction

Disaster Response is the second phase in the four-phase process of Disaster Management (Preparedness, Response, Rehabilitation and Reconstruction, Mitigation)—a process that government and authorities follow to reduce potential damage from hazards, assure prompt and appropriate assistance to victims during a disaster, and recover after that according to WHO training package document [1]. Disaster Response happens immediately aftermath of a disaster and aims to minimize the damage by conducting assistance services (searching and rescuing), distributing supplies, and medical care. In this phase, emergency responders are normally divided into teams with different missions. Reliable and timely information exchange between these responders and their commanders is the key to make an effective disaster response phase.

However, several challenges make it difficult to communicate efficiently in a disaster scenario. First, infrastructure is usually damaged and broken after a disaster happens. With the messaging server is down, most normal centralized message delivery applications cannot work. Second, communication using the current IP-based network show limitations. Considering responders' dynamic roles and high mobility when participating in the disaster response phase, it is difficult to know each individual IP address to contact. This causes a significant delay in emergency information delivery during the disaster. As a result, inappropriate resource allocation, late assistance when rescuing people happen and these issues can lead to unwanted and serious consequences.

Because of these challenges, it is necessary to find a network solution that has mobility built-in support and enable location-independent, role-based group communication between responders. In recent years, several studies [2–4] have shown that Information-Centric Networking (including NDN—one of its architectures) can improve IP-based network weaknesses in providing these mentioned features in disaster. Specifically, NDN uses the content name as the main entity inside the network to route the packet instead of IP address–content location in IP-based network. It allows NDN consumers to fetch data from anywhere in the network storing the copy of the data that has the same name as the interest packet (the NDN request packet). This location-independent feature enables group role-based message delivery in NDN. People in the same messaging group can fetch messages by using the group's unique name prefix. Moreover, NDN location-independent feature enables native mobility support. When a user moves to a new location, data can be requested using its name instead of having to know the new location address like in an IP-based network.

Although several studies [3–12] have leveraged NDN in dealing with disaster management issues, real implementation of an NDN-based architecture over existing infrastructure during a disaster is lacking. Most works' results are limited to simulation analysis. Moreover, when deploying an NDN network in a disaster scenario, apart from role-based communication, additional challenges should be considered. First, infrastructure is normally damaged during a disaster, quick replacement time for network nodes is required. Second, despite the NDN mobility built-in support feature, only the consumer mobility issue is natively supported. Due to emergency responder high mobility, producer mobility problem is also required to be addressed. NDN needs to be enhanced so that network can automatically build routing paths to the producer's new location after moving. Another challenge is the intermittent network caused by disaster hazards. To effectively manage tasks during the disaster response phase, reported information from responders to the management center and commands from commanders to responders should ensure to be delivered without any loss. Missing information can seriously affect searching and rescue efforts.

In this paper, we designed and implemented an NDN-based disaster response support system over Edge Computing infrastructure with KubeEdge as the chosen implementation edge platform to solve the above issues. In this proposed architecture, we design a deployment strategy to establish an NDN network at both one region level and multiple regions level. NDN network functions are deployed as containers from cloud to edge equipment to provide emergency communication. Each cluster contains several edge nodes to provide an NDN network for one region. Master nodes from each cluster are tunneled to connect the NDN network between two regions. In case of damaged edge nodes caused by the disaster, NDN containerized function will be deployed from the cloud to replaced nodes. With Named-data link-state protocol (NLSR) [13] enabled, the NDN network can converge quickly after replacement. We also enhance NDN with a protocol between user devices and border edge nodes to solve the mobility challenge. Finally, we utilize KubeEdge to provide reliable information sharing between cloud and edge in discontinuous network conditions. The architecture is implemented using KubeEdge on multiple servers and NUCs. For end-users, we design an exclusive NDN disaster application that provides message delivery and information exchange with the management center function. Our proof-of-concept system performance shows that the architecture obtains the following achievements:

- A deployment architecture to provide emergency communication over NDN network and NDN device management for disaster information exchange
- Faster network convergence time after replacement in case of damaged network node caused by disaster compared to IP-based network
- Faster mobility handover duration compared to Mobile IP [14] and rendezvous mobility solution for NDN [15]

- Low information exchange transmission overhead between cloud and responder devices
- Ensuring loss-free information exchange between responders and management center at cloud compared to normal NDN method without using edge platform.

Compared with the preliminary version [16], this paper provides more details about the architecture design, presents implementation results and analysis.

The remaining sections are organized as follows: Section 2 presents the related works. Subsequently, we present our proposed architecture and system design, including NDN network deployment architecture over edge computing, NDN mobility protocol, and information exchange mechanism between cloud and responders through KubeEdge in Section 3. Section 4 shows our detailed implementation and Section 5 shows evaluation results.

2. Related Works

2.1. Named Data Networking Advantages over IP-Based Network in Disaster Response

The limitation of IP-based networks compared with NDN in disaster response has been discussed in several studies [2–4]. There are three main advantages of NDN over IP architecture. Firstly, location-independent routing capability allows NDN to perform more timely information dissemination than IP-based networks. Group communication between responders can be easily created by using group name prefixes in NDN. Meanwhile, IP addresses need to be distributed first before messages can be exchanged. Moreover, the NDN in-network caching feature allows it to provide lower latency data delivery than the IP network. Secondly, NDN can handle the responder mobility challenge better than IP architecture. NDN provides mobility built-in support whereas IP need to use patch like mobile-IP. Lastly, since bandwidth is limited inside the network in disaster scenarios due to damaged network infrastructure, NDN is more convenient. By using the name as the main network entity, NDN packet is smaller than IP counterpart which includes the address inside. Moreover, the interest aggregation feature of NDN also helps it to utilize network bandwidth.

2.2. Previous Research Efforts

With these clear advantages of NDN over IP-based networks in disaster management, the research community has contributed several disaster-related solutions using this network architecture. Although state-of-the-art NDN has been applied in message delivery [9,10], social networking service [11], and vehicular communication [12] in disaster scenarios, most works extend and improve NDN features to enhance its performance in different aspects. Koki et al. [8] improve the capability of naming disaster information in NDN by using natural language processing. Abdul et al. [3] added lightweight push support to the original NDN for an NDN-IoT-based system for disaster management. This extension helped reduce system latency and increase its throughput. Mohammad et al. [5] proposed a disaster NDN information dissemination with complex graph-based namespaces, automatic name-based load-splitting support on a recipient-based publish/subscribe architecture. Their system improved state-of-the-art NDN in terms of information delivery time and load sharing. Another kind of approach in leveraging NDN in disaster management solutions is combining NDN with other architecture. Yang et al. [7] adopted NDN in Mobile Ad hoc networks (MANET). The authors proposed a proactive routing protocol that extends NDN's Forwarding Information Base (FIB) and leverages NDN multipath forwarding support. Their system improved transmission efficiency in MANET.

However, it is complex to evaluate these systems in a real disaster scenario since most previous works are simulated using NS-3 network simulator ndnSim [17]. Only a few works have real implementations of their systems. The mentioned NDN-based MANET [7] is demonstrated by a platform consisting of several Raspberry Pi smart cars. NDN engine with the proposed routing protocol is installed in these Raspberry Pis. Another work that has real demonstration is the graph-based namespaces NDN information dissemination system [5]. The demonstration [6] consists of several responder devices, a coordination center, and a middle device moving between them to update responder devices' namespaces. Although their results are reliable, these architectures are only applicable to device-to-device (D2D) network. User devices can run out of energy anytime because power supplies might not be available after the disaster happens. Moreover, the D2D network is not reliable in large-scale disaster scenarios. Therefore, it is necessary to find a way to deploy an NDN-based system on the remaining infrastructure.

Our previous works [18] has already presented a real implementation of a name-based emergency communication system by virtualizing NDN network functions over Kubernetes. However, we find it necessary to enhance our system to deal with other disaster-related challenges so that it can work effectively in real scenarios. In this paper, our proposed architecture focus on the following issues. First, producer mobility should be supported because responders frequently move during their disaster response missions. Second, since network infrastructure can be damaged by hazards during the disaster, the network architecture should be able to re-converge fast in case of replacement. Lastly, the system should be able to prevent information loss caused by intermittent network conditions. In this work, we design a mobility protocol between responder devices and border edge and utilize KubeEdge—a lightweight edge computing platform to address these mentioned problems.

2.3. KubeEdge

KubeEdge [19] is an open-source edge computing platform that is designed to extend Kubernetes containerized orchestration capabilities to hosts at Edge. It supports networking, application deployment, and metadata synchronization between cloud and edge devices. KubeEdge architecture consists of two parts: Cloudcore and Edgecore as shown in Figure 1.

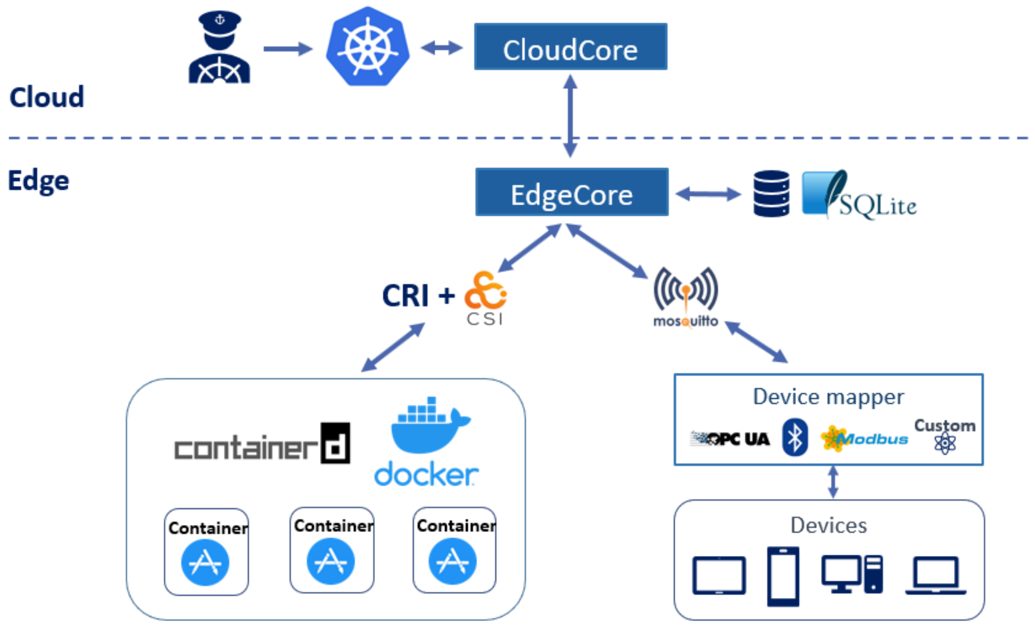


Figure 1. KubeEdge architecture.

Compared with Kubernetes, orchestration capabilities are separated from edge nodes to the cloud side. Cloudcore is the extended Kubernetes controller that controls edge nodes and pods metadata so that data can be targeted to specific edge nodes. Meanwhile, computing functionalities—pod deployments are handled by Edgecore.

Several KubeEdge advantages make us choose it as the edge platform to deploy NDN network functions in disaster scenarios. First, Edgecore has a lightweight size. It is a node agent equivalent to kubelet [20] in Kubernetes. However, several features of kubelet are stripped out and SQLite [21] is used instead of etcd [22] to make Edgecore's binary size small and consume much less memory than kubelet. This allows this edge node agent to be installed even in resources-constraint devices when normal network infrastructure is heavily damaged by the disaster. Second, Edgecore supports device management with multiple protocols such as Bluetooth, Modbus [23], OPC-UA [24]. A custom protocol can also be defined by KubeEdge users. This feature allows us to use KubeEdge to manage NDN responder devices. Information can be exchanged between the disaster management commander at cloud and responders by utilizing KubeEdge's Mosquitto publish/subscribe framework [25]. Finally, KubeEdge supports transmission reliability between cloud and edge, which is a very important feature, especially in disaster intermittent network conditions. The mechanism that enables this feature is called At-Least-Once Delivery [26] and the architecture design between Cloudcore and Edgecore that enables this feature is shown in Figure 2.

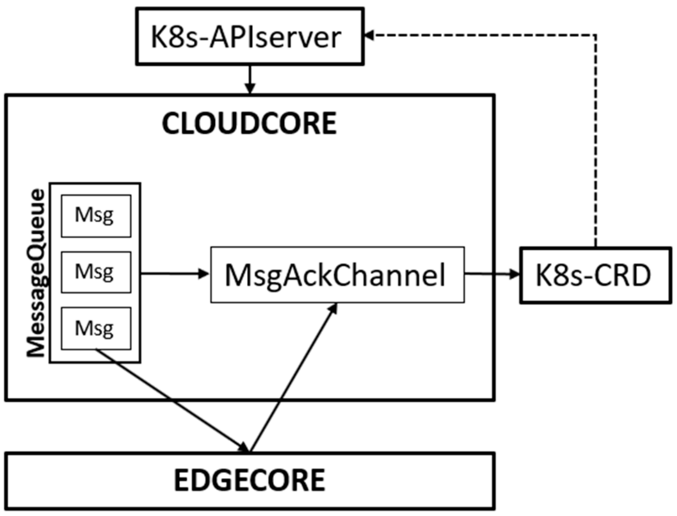


Figure 2. KubeEdge Cloud-Edge transmission reliability design.

The Kubernetes Custom Resources Definition (K8s-CRD) is used to store the resource version of the messages which are sent from cloud to edge. The message is marked as completed only when Cloudcore receiving the acknowledgment through the Message Acknowledgement Channel (MsgAckChannel) from Edgecore after successful message receipt here. In case of disconnection between cloud and edge happens, unacknowledged messages will be kept in the messages queue for resending after the connection is recovered. With this mechanism, data loss communication between cloud and edge can be prevented and recovered when the network is unstable.

3. System Design

The general architecture of deploying NDN network over edge computing infrastructure has two separated parts: cloud and edge. The Cloud side is responsible for edge nodes management, NDN containerized network functions deployment over managed edge nodes. The edge side contains all the edge nodes located around disaster areas. Each edge node is responsible for running one containerized NDN router. Among them, nodes which provide Wi-Fi access point are border edge nodes. They act as NDN gateway for responders to connect and exchange information through the NDN network.

When deploying this architecture in disaster scenarios, a suitable edge platform that can work well in disaster conditions is required. Based on the characteristics that we discussed in the previous part, we choose KubeEdge as the platform to deploy the NDN network over.

3.1. NDN Deployment over KubeEdge Architecture

We first demonstrate the general network architecture for a single region where one KubeEdge cluster is used to deploy NDN network. The architecture is shown in Figure 3.

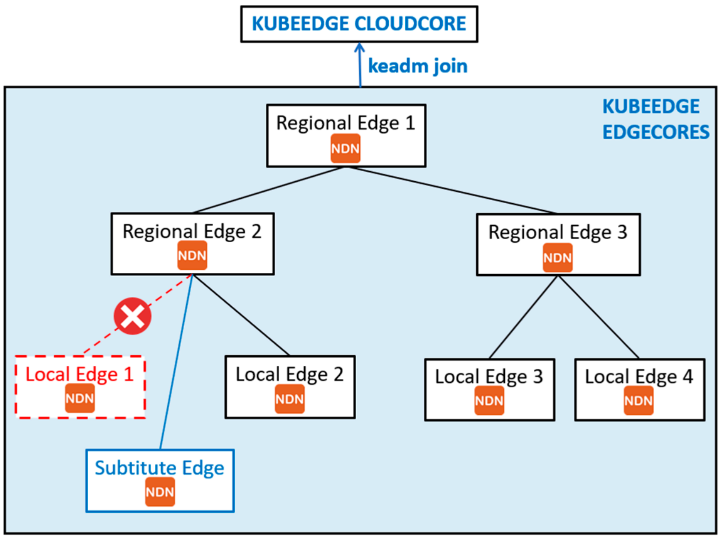


Figure 3. Named Data Networking (NDN) over KubeEdge general network architecture in single region.

KubeEdge Cloudcore is the sole component on the cloud side. Cloudcore manages all edge nodes in the cluster and responder devices' information that connect to them. It is also responsible for deploying containerized NDN routers at the correct edge nodes based on their pre-defined yaml files.

The edge side of the architecture contains edge nodes with KubeEdge Edgecore installed inside them. Each edge node runs one instance of NDN containerized router deployed from CloudCore. Border edge nodes have Wi-Fi access points to provide an NDN gateway for responders to connect to the NDN network. In case an edge node is damaged during the disaster, Edgecore binary can easily be installed into any network equipment (even resources constraint once) thanks to the lightweight size of the Edgecore. After that, given that Cloudcore information is pre-installed inside the replaced edge node by an emergency repair team member, this node can re-join the KubeEdge cluster and deploy the corresponding NDN routers based on the command from Cloudcore. The NDN network is recovered at this moment.

For large scale disaster area which spreads over multiple regions, multiple KubeEdge clusters are needed to provide NDN network. Figure 4 shows our designed architecture for deploying the NDN network using KubeEdge over multiple regions.

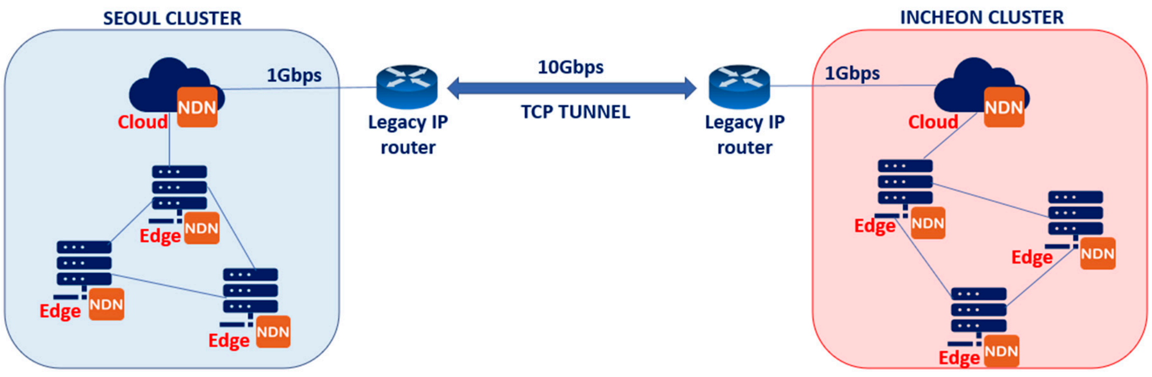
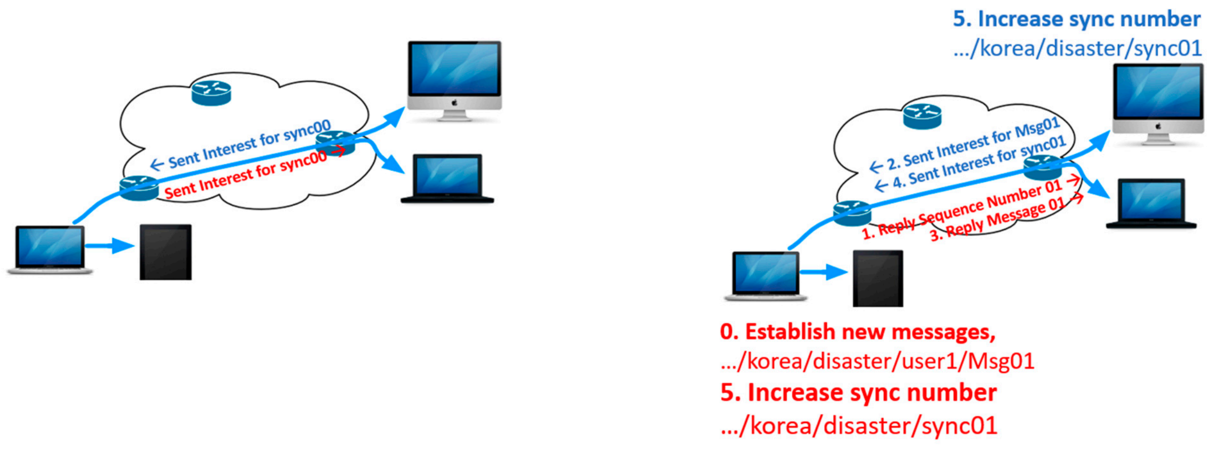


Figure 4. NDN over KubeEdge general network architecture in two regions.

The NDN network in each region will connect to an NDN router deployed at Cloud-core node. This router will act as an NDN gateway for each region and connect to an IP gateway router. To connect the network between two regions, a high-speed, high-bandwidth TCP tunnel will be created between two IP gateway routers. This design is referred to as a real NDN deployment presented in [27].

3.2. NDN Emergency Communication Design

During the disaster response phase, emergency responders are normally split into several groups. Each group includes several members that have the same role, and do the same mission. Moreover, each responder may have multiple roles. Hence, we design an emergency communication namespace that allows each responder to participate in multiple group message rooms based on their roles. Figure 5 shows how messages are delivered inside the NDN network using our designed namespaces.


← Sent Interest for sync00
Sent Interest for sync00 →
5. Increase sync number
.../korea/disaster/sync01
← 2. Sent Interest for Msg01
← 4. Sent Interest for sync01
1. Reply Sequence Number 01 →
3. Reply Message 01 →
0. Establish new messages,
.../korea/disaster/user1/Msg01
5. Increase sync number
.../korea/disaster/sync01

(**a**) (**b**)

Figure 5. NDN emergency group message delivery: (**a**) Balance state; (**b**) New message state.

The distributed NDN message delivery mechanism is based on ChronoSync [28]. Each group has two kinds of interest packets: group interest and user message interest. They are named following the role of responders in the same group and their format is given in Table 1. The group interest is used for fetching the sequence number of the message from other users. The user message interest is used for fetching the content of the message. At a balanced state, when there is no new message from any responder inside the group, everyone will send a group interest with the same sync number. When one responder establishes a new message, his device will reply with a data packet containing the new message sequence number to other responder devices. Then, they will send the message interest with that received sequence number to fetch the new message. After that, every responder device in the group will send the new sync interest with the sync number increased by one, and the balance state is achieved again.

Table 1. Emergency group communication namespace.

Interest Kind	Format
Group	responderRole/syncNumber
User Message	responderRole/userID/messageSequenceNumber

With this namespace design, we can provide flexible and location-independent emergency communication for responders. Figure 6 shows how we applied our namespace design for message delivery in a Seoul disaster scenario. Four responders participate in three messaging channels in this scenario. "/disaster01/seoul/dongjak/emergency" channel is for all emergency responders (including hospital, fire and police responders) who is doing their mission in Dongjak. "/disaster01/seoul/hospital" channel is for all hospital responders in Seoul. "/disaster01/seoul/general" channel is the general channel for everyone. Each responder can join multiple channels based on their roles. When responders connect to a border edge node and join a channel, route to responder devices will be automatically advertised to the NDN network by the edge node. Hence, no matter where the responders are, they can send and receive messages from their channels using the group name prefixes.

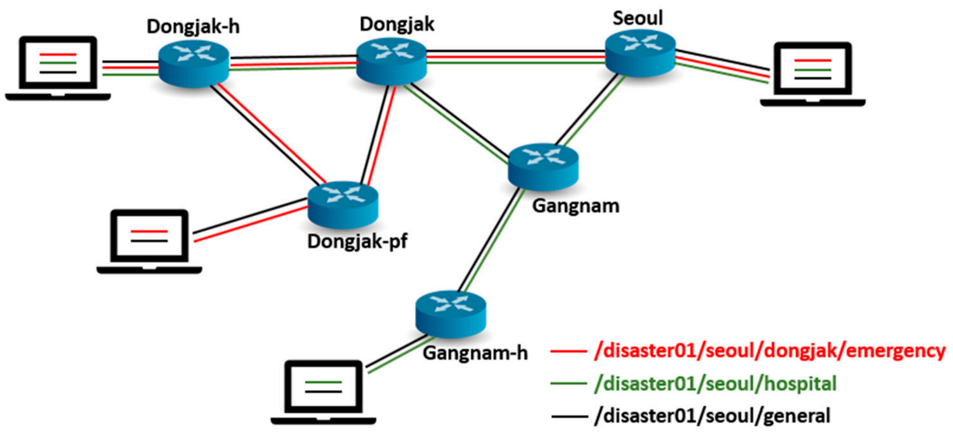

Dongjak-h
Dongjak
Seoul
Gangnam
Dongjak-pf
Gangnam-h
/disaster01/seoul/dongjak/emergency
/disaster01/seoul/hospital
/disaster01/seoul/general

Figure 6. Multiple concurrent NDN group messaging channels in Seoul disaster scenario.

3.3. NDN Mobility Support Design

As responders need to move a lot during their missions in the disaster response phase, mobility support is an essential feature that our network architecture should provide. Our proposed design is shown in Figure 7.

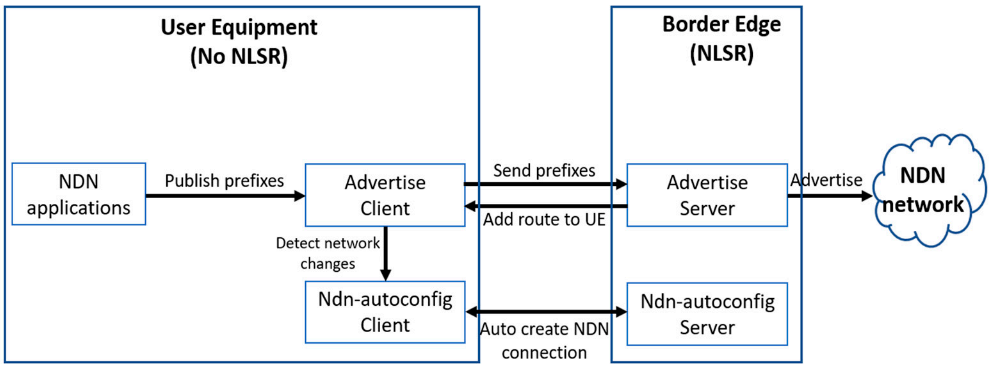


Figure 7. NDN mobility support design.

After moving, NDN name prefixes from responders' devices need to be advertised to the NDN network to continue message delivery. To minimize the handover time, we add two pairs of client and server at user equipment and border edge so that this advertisement process can be performed immediately at the node that the responder device connects to. Normally, when responders join a group channel, the group prefix and the message prefix will be sent from the advertising client to the advertising server at the edge node. They will be advertised to the NDN network by the NLSR engine here. These prefixes will also be saved in the local database of the advertising client. When moving to a new location and connecting to a new border edge node, the advertising client will detect the network change and inform the ndn-autoconfig [29] client. Ndn-autoconfig client will automatically create an NDN connection with the corresponding server running at the edge node. After the NDN connection is established, those saved prefixes at the advertising client will be sent to the server for advertising to the NDN network. At the same time, routes back to user equipment for these prefixes will also be created. At this moment, the responder can send and receive NDN messages normally.

3.4. NDN Device Management Using KubeEdge as Edge Computing Platform

Effective disaster response task management requires the commander at the management center to have an overview picture of what is happening around disaster areas. Hence, we utilize KubeEdge device management feature to create an NDN responder device management system for the centralized commander on the cloud. Through this system, the commander can receive the reported mission status of all responders (current event, requirement, location) from their devices. Based on this information, the commander can directly update responders' missions, roles from cloud to responder devices. Moreover, we take advantage of the reliable cloud-edge transmission feature of KubeEdge so that no information (reports from devices and updates from cloud) between cloud and devices is dropped in intermittent network conditions.

The information exchange mechanism between cloud and responder devices is demonstrated in Figure 8. Update information from the commander at cloud will be sent to Edgecore and publish to Mosquitto topics here. Subscribed devices will receive updates. Meanwhile, reported information from responders will be processed by the device-mapper and published to Mosquitto topic. Edgecore gets reported data from subscriptions and sends it to the cloud. Commander can get data from the Kubernetes API Server.

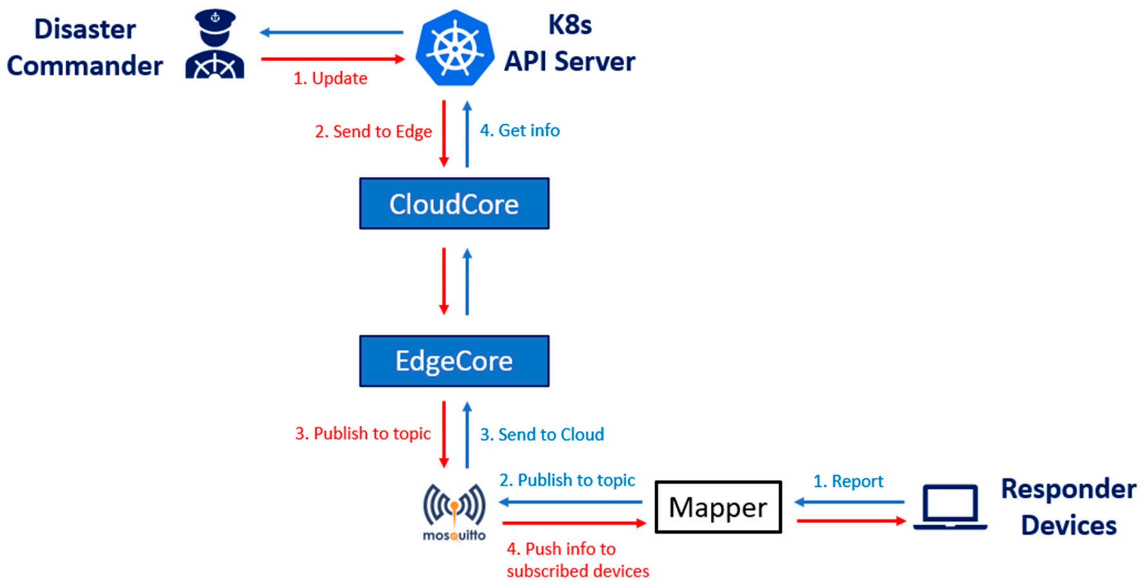


Figure 8. Device Management using KubeEdge.

4. Implementation

4.1. System Topology and Configuration

Figure 9 shows the cross-regional NDN network topology in our implementation. We used physical servers for Cloudcore and normal edge nodes. D34010WYK NUCs are used for border edge nodes and replacement nodes. KubeEdge version 1.4 is installed in every node. NDN-cxx [30], NFD [31] version 0.6.5, and NLSR version 0.5 are used for NDN network. IBRDTN [32] version 1.0.1 is used for delay-tolerant networking support. For end-users, we use Lenovo ideapad D330 notebooks. An exclusive disaster response assistance application is also designed for responders. It provides two features: NDN group communication and NDN device management (updating missions from cloud to device and reporting event, requirement, location information from devices to the cloud). The detailed configuration of our implementation is shown in Table 2.

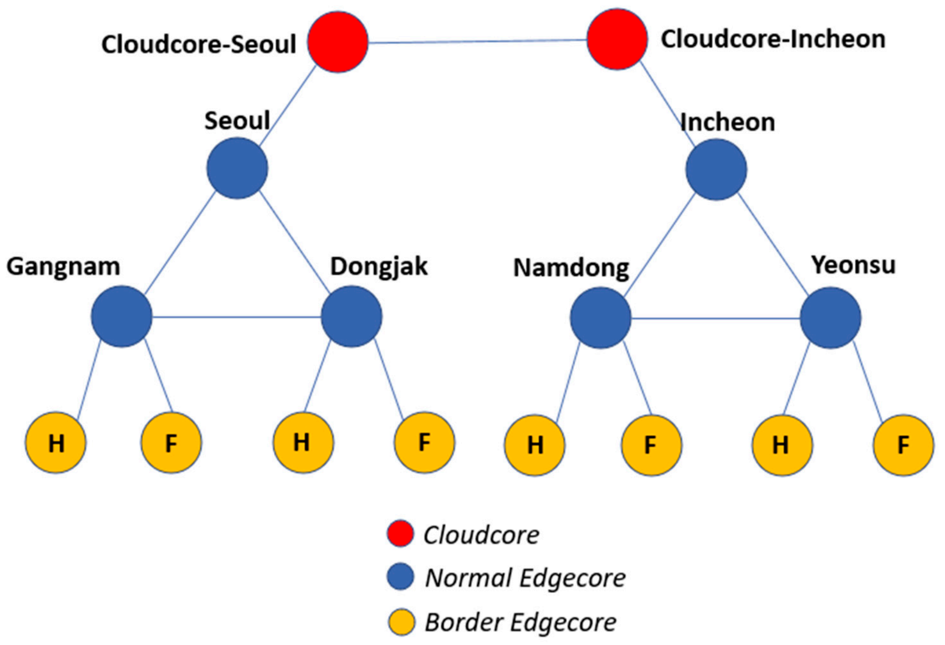


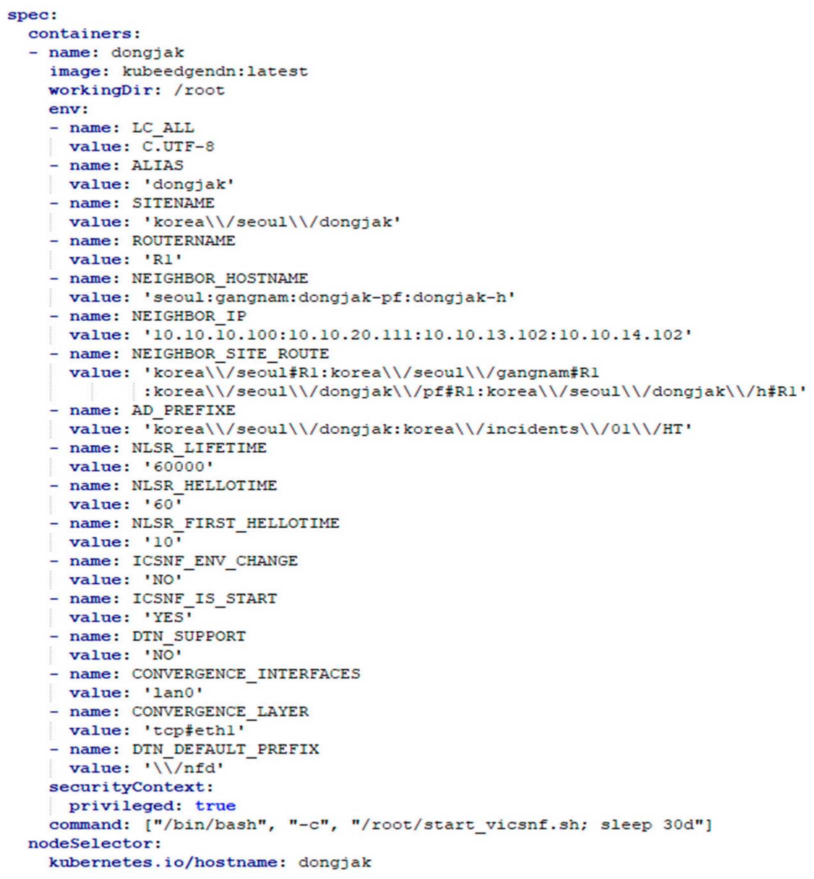
Figure 9. Cross-regional NDN network topology in the implementation.

Table 2. Detailed implementation configuration.

Entity	Hardware	Software
Cloudcore	CPU: Intel® Xeon 2.4 GHz, RAM: 256 GB, HDD: 1.8 TB	Ubuntu server 18.04 KubeEdge Cloudcore v1.4 NDN, NFD v0.6.5 NLSR v0.5
Normal Edgecore	CPU: Intel® Xeon 1.5 GHz, RAM: 64 GB, HDD: 440 GB	Ubuntu server 18.04 KubeEdge Edgecore v1.4 NDNcxx, NFD v0.6.5 NLSR v0.5 Ibrdtn v1.0.1
Border Edgecore	CPU: Intel® Core i3 1.7 GHz, RAM: 8 GB, HDD: 120 GB	Ubuntu 16.04 KubeEdge Edgecore v1.4 NDNcxx, NFD v0.6.5 NLSR v0.5 Ibrdtn v1.0.1
End-user	CPU: Intel® Dual Core, RAM: 8 GB, HDD: 128 GB	Ubuntu 16.04 NDNcxx, NFD v0.6.5 NLSR v0.5 Ibrdtn v1.0.1

4.2. NDN Network Deployment

NDN routers are deployed using pre-configured yaml files. This information helps NDN routers can automatically create NDN connections with other NDN nodes in the network. An example of an NDN router's yaml file is shown in Figure 10. In this file, information about the deployed NDN router's name (ALIAS, SITENAME), neighbor nodes (NEIGHBOR_IP, NEIGHBOR_HOSTNAME, NEIGHBOR_SITE_ROUTE), pre-advertised prefixes (AD_PREFIXE), NLSR configuration (NLSR_LIFETIME, NLSR_HELLOTIME, NLSR_FIRST_HELLOTIME), Delay-Tolerant Networking [33] (DTN) configuration (DTN_SUPPORT, DTN_DEFAULT_PREFIX) and targeted edge node (kubernete.io/hostname) is defined in environment variables. Based on these variables, once being deployed, NDN routers can automatically start up and link with their neighbor nodes to create NDN network between them. Pre-advertised prefixes will be advertised by NLSR engine so that information can be delivered immediately through the deployed routers. Moreover, NDN mobility handling and edge device management software are pre-installed inside the image.



```
spec:
  containers:
  - name: dongjak
    image: kubeedgendn:latest
    workingDir: /root
    env:
    - name: LC_ALL
      value: C.UTF-8
    - name: ALIAS
      value: 'dongjak'
    - name: SITENAME
      value: 'korea\\/seoul\\/dongjak'
    - name: ROUTERNAME
      value: 'R1'
    - name: NEIGHBOR_HOSTNAME
      value: 'seoul:gangnam:dongjak-pf:dongjak-h'
    - name: NEIGHBOR_IP
      value: '10.10.10.100:10.10.20.111:10.10.13.102:10.10.14.102'
    - name: NEIGHBOR_SITE_ROUTE
      value: 'korea\\/seoul#R1:korea\\/seoul\\/gangnam#R1
              :korea\\/seoul\\/dongjak\\/pf#R1:korea\\/seoul\\/dongjak\\/h#R1'
    - name: AD_PREFIXE
      value: 'korea\\/seoul\\/dongjak:korea\\/incidents\\/01\\/HT'
    - name: NLSR_LIFETIME
      value: '60000'
    - name: NLSR_HELLOTIME
      value: '60'
    - name: NLSR_FIRST_HELLOTIME
      value: '10'
    - name: ICSNF_ENV_CHANGE
      value: 'NO'
    - name: ICSNF_IS_START
      value: 'YES'
    - name: DTN_SUPPORT
      value: 'NO'
    - name: CONVERGENCE_INTERFACES
      value: 'lan0'
    - name: CONVERGENCE_LAYER
      value: 'tcp#eth1'
    - name: DTN_DEFAULT_PREFIX
      value: '\\/nfd'
    securityContext:
      privileged: true
    command: ["/bin/bash", "-c", "/root/start_vicsnf.sh; sleep 30d"]
  nodeSelector:
    kubernetes.io/hostname: dongjak
```

Figure 10. NDN containerized router deployment file example.

4.3. *KubeEdge Extension for NDN Device Management*

We extend KubeEdge Edgecore with an NDN Edge Management client. This client communicates with the ndn-autoconfig server running inside each edge node. Whenever a responder connects to a border edge node, if it is the first time he joins the emergency communication network, the edge client will ask the Cloudcore to register the device to the system as a Kubernetes custom resource. Otherwise, the edge client will tell the Cloudcore to enable or disable the device based on the responder connection status to the edge. Moreover, we design an NDN mapper for responder devices. After the device is registered at Cloudcore, it can use the mosquitto publish/subscribe framework to get updates from the cloud or push information to the central management center. Commander can monitor and update responder devices information by interacting with the KubeEdge control plane at cloud node by using commands such as "kubectl get device" and "kubectl edit device". The extended KubeEdge architecture design is shown in Figure 11.

Figure 12 shows a sequence diagram of how a new responder NDN device is registered to the management system, subscribes to get mission updates, and publishes event status to the cloud.

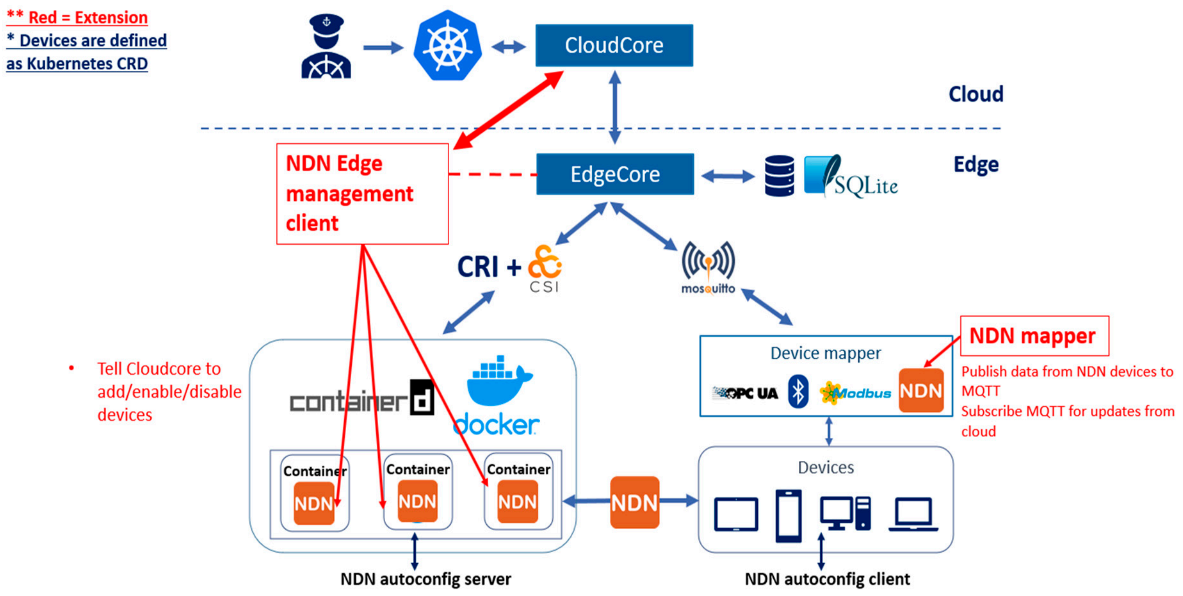


Figure 11. KubeEdge extension to support NDN device management.

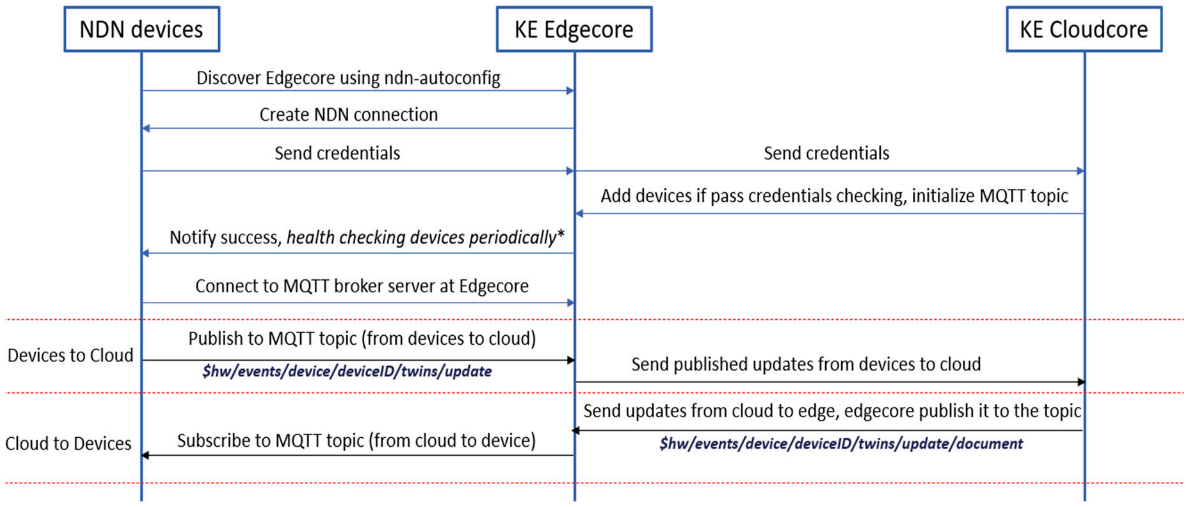

NDN devices
KE Edgecore
KE Cloudcore
Discover Edgecore using ndn-autoconfig
Create NDN connection
Send credentials
Send credentials
Add devices if pass credentials checking, initialize MQTT topic
Notify success, health checking devices periodically*
Connect to MQTT broker server at Edgecore
Devices to Cloud
Publish to MQTT topic (from devices to cloud)
$hw/events/device/deviceID/twins/update
Send published updates from devices to cloud
Cloud to Devices
Send updates from cloud to edge, edgecore publish it to the topic
Subscribe to MQTT topic (from cloud to device)
$hw/events/device/deviceID/twins/update/document

Figure 12. New NDN device management sequence diagram.

5. Evaluation

5.1. Network Convergence Time in Case of Node Replacement

During a disaster, network nodes are usually damaged and need to be replaced. Network convergence time after replacement should be as fast as possible to re-enable emergency communication through the network. Hence, we evaluated our system network convergence time by comparing between replacing NDN edge nodes and IP edge nodes. Both NDN routers and IP routers are deployed as containers over KubeEdge. We chose the EIGRP routing protocol [34] for IP networks because it has been proven to be the fastest convergence IP routing protocol by several studies [35,36]. Network convergence time is calculated from the moment when the replaced node joins the KubeEdge cluster until the routing advertisement process is completed. The equation for calculation is:

$$T_{NC} = T_{CJ} + T_{RD} + T_A \tag{1}$$

where T_{NC} is network convergence time, T_{CJ} is cluster joining time, T_{RD} is containerized router deployment time and T_A is advertisement time. We gradually increased the amount of replaced nodes at the same time to evaluate the performance. The network convergence time of our system is shown in Figure 13 and the comparison between it and the IP system is shown in Figure 14.

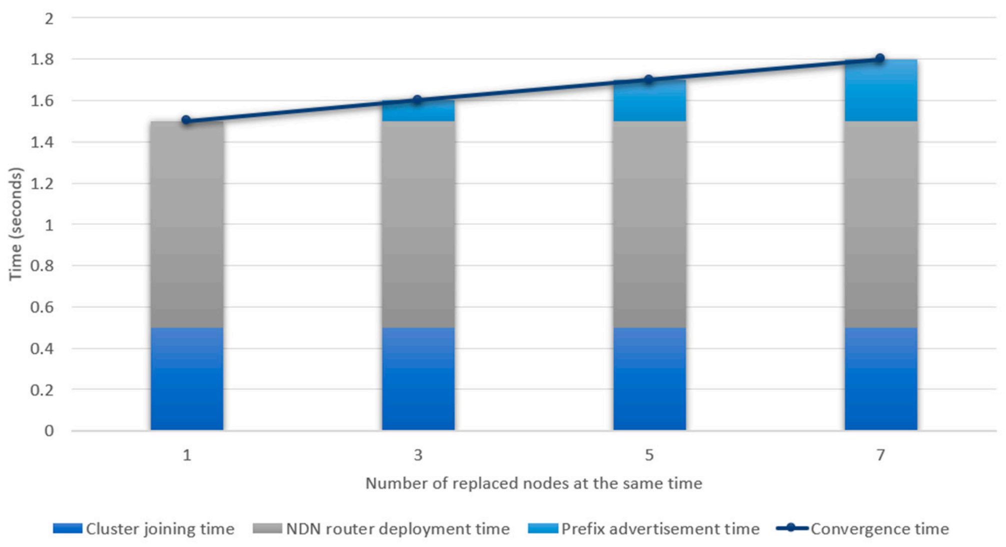


Figure 13. Our system network convergence time.

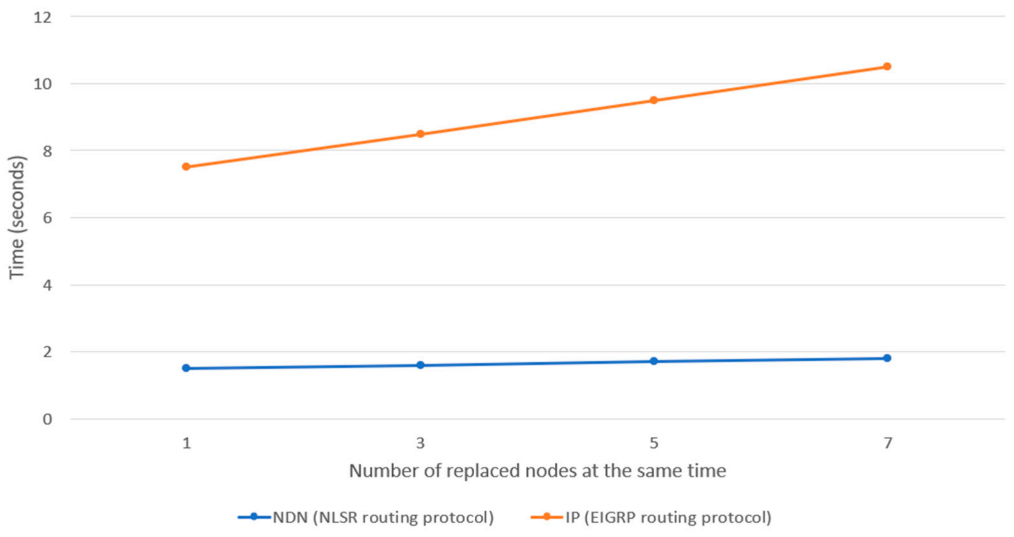


Figure 14. The comparison of network convergence time between NDN and IP network.

Figure 13 illustrates that our system achieved a very fast convergence time even when multiple nodes needed to be replaced simultaneously. The network can recover to a normal state after replacement in less than 2 s and the impact of increasing the amount of replaced nodes simultaneously is trivial (only 0.1 s per node). Compared with the IP network, Figure 14 shows that the NDN network with NLSR routing protocol converges much faster than the IP network with EIGRP routing protocol. The NDN network only requires 1.8 s to converge while the IP network requires 10.5 s when the amount of replaced nodes is 7. Moreover, when the number of replaced nodes at the same time increases, the convergence time for the NDN network increases slightly while it increases significantly for the IP network. The difference is caused by NLSR faster advertisement time thanks to its multipath routing calculation feature [37].

5.2. Mobility Handover Duration

We compared our mobility support method with rendezvous NDN mobility support proposed in [15] and Mobile-IP [14]. We consider two levels of network topology to evaluate. One topology is for a single region and the other one is for cross regions. The scenario is group communication between two responders connecting to two border edge nodes in the same district (Gangnam in our scenario). Then, one responder will move to another district. The topologies and scenarios are shown in Figure 15.

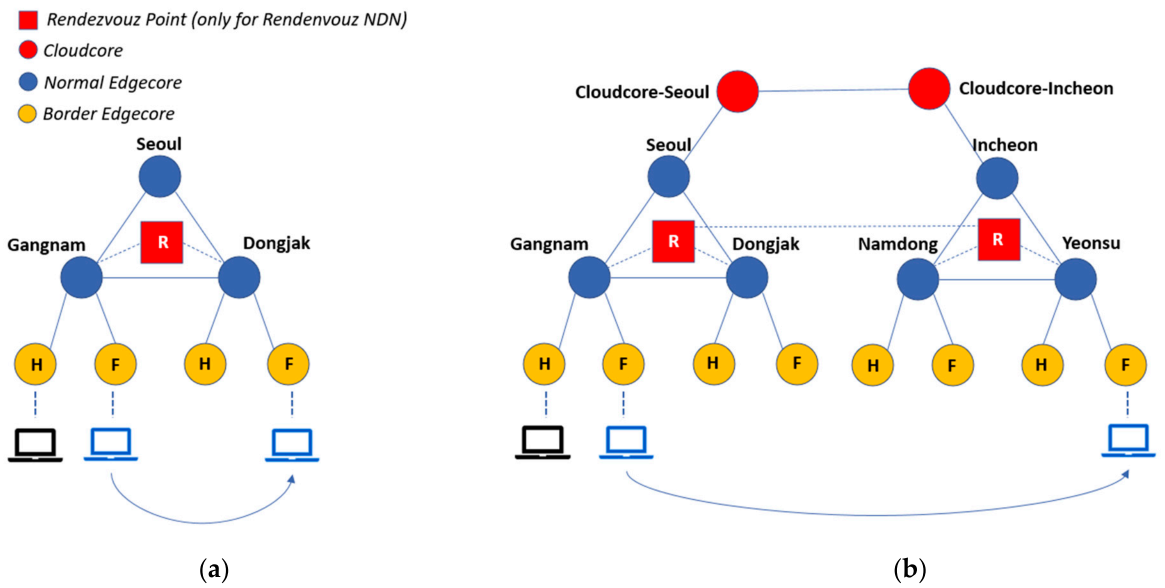


Figure 15. Network topologies and scenario for mobility handover measurement (**a**) Single region topology; (**b**) Cross-region topology.

For the rendezvous NDN mobility method, we deploy the rendezvous server at the center of the topology for each region. Rendezvous servers between regions are interconnected. The rendezvous server functionality is kept the same as [15]. When receiving mobility handling request (routes advertisement request from a user device when it connects to a new edge node), this server will update intermediate nodes between it and the device with new routes. Additionally, any interest that does not has the route to the newly moved device will be forwarded to the rendezvous server. After that, the interest will be forwarded to the device using new updated routes.

We measured mobility handover duration based on the amount of network hops that the mobility handling request from the device needs to go through. Figure 16 shows the comparison between our mobility method, rendezvous NDN method, and Mobile-IP.

The comparison shows that our proposed mobility support method has the lowest mobility handover duration. In our method, because the mobility handling process is conducted right at the edge node that the moving device connects to, the number of hops needed in both topologies is only 1. In the rendezvous NDN method, the mobility handling request needs to be forwarded from the edge node to the rendezvous server. Hence, the number of hops is 2 for single region topology and 3 for cross-region topology (one more hop between two rendezvous servers at two regions). In the Mobile-IP method, an announcement packet will be sent from the newly connected edge node to the previously connected one so that the number of hops is 4 and 7 for single region topology and cross-region topology respectively.

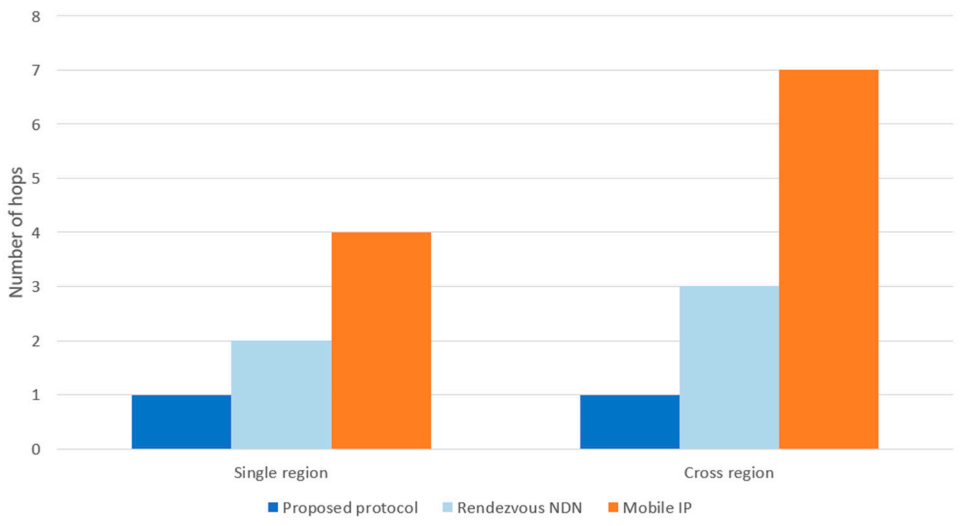


Figure 16. The comparison of mobility handover duration.

5.3. Transmission Overhead When Exchanging Information between Cloud and Device

We evaluated the benefit of using KubeEdge to manage NDN devices by comparing it with two other disaster management systems that do not use Edge computing platforms: the normal publish/subscribe NDN management system [38] and the state-of-the-art NDN-DM system [39]. In our system, KubeEdge uses Mosquitto as the publish/subscribe framework for information exchange between cloud and devices. Meanwhile, the NDN publish/subscribe communication framework and NDN push-based mechanism are used in the other two systems respectively. We created two NDN device management systems that followed these works' design to make a comparison with our system. As mentioned in previous parts, responders can upload an event, a requirement, or his location to the cloud, and the commander on the cloud can update responder missions. We considered one event/requirement/location/mission as a status. We monitored the number of transmitted packets between cloud and device when the amount of exchange status increases. The result is shown in Figure 17.

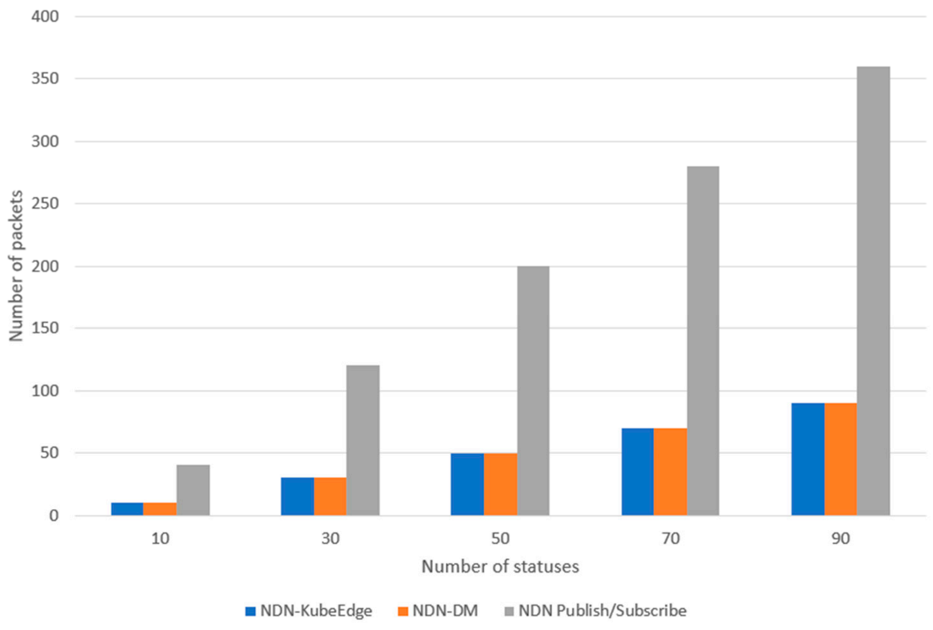


Figure 17. Number of transmitted packets to synchronize status between cloud and devices comparison between our system versus NDN Publish/Subscribe framework and NDN-DM.

The result shows that the number of transmitted packets when using NDN Publish/Subscribe increases proportional with the number of statuses, while they are equal when using KubeEdge and NDN push mechanism in NDN-DM. Specifically, the number of exchange packets between cloud and devices when using the NDN Publish/Subscribe framework is 4 times higher than our system and NDN-DM. The reason is that the NDN Publish/Subscribe needs to exchange four interest/data packets between them to synchronize one status [38], while the cloud/device only needs to push one packet containing the status to the Mosquitto framework in KubeEdge. The same reason is applied for NDN-DM as only one packet is needed to push a status to the destination. With lower transmission overhead when exchange status between cloud and devices, our system can avoid congestion and packet losses in high network traffic conditions in disaster scenarios. Our system is as efficient as NDN-DM in terms of reducing information transmission overhead. The advantage of it over NDN-DM is presented in the next part.

5.4. Packet Recovery Capability in Intermittent Network

During the disaster response phase, the network is not reliable. Packet recovery capability is a key feature to avoid valuable information loss. We kept comparing our system with the NDN Publish/Subscribe system and NDN-DM to show the advantage of using KubeEdge to recover packet loss in an intermittent network. In the cross-regional topology, we disconnected several network links, performed 50 packets exchange between cloud and devices. Then, we monitored the successful status exchange ratio between them after reconnecting the network links (the amount of received status over the amount of sent status). The result is shown in Figure 18.

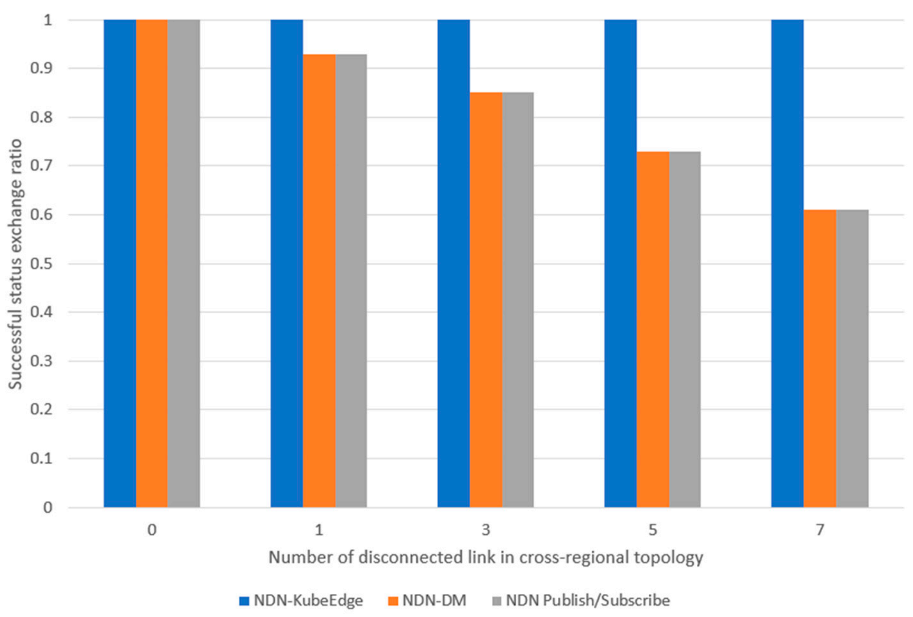


Figure 18. Packet recovery capability comparison between our system versus NDN Publish/Subscribe and NDN-DM.

The figure shows that KubeEdge can successfully recover every dropped packet caused by network disconnection. Meanwhile, for the other two systems, the successful percentage of information exchange gradually drops down when the number of disconnected links increases. This result shows the effectiveness of the cloud-edge transmission reliability design of KubeEdge which can resend packets after network links are recovered. Information exchange through the other two NDN networks suffers from packet loss because there is no recovery support.

5.5. System Performance Discussion

In this part, we aggregate our system performance evaluation presented in four previous parts in Table 3 to highlight our system contribution based on comparison with other relevant systems.

Table 3. Our system aggregated results.

Compared Aspects	Our System Features	Compared Features	Results
Network convergence time after replacing damaged nodes	NDN's NLSR routing protocol	IP's EIGRP routing protocol [34]	Faster
Mobility handover time	Our NDN mobility support design	Mobile-IP [14] NDN rendezvous mobility mechanism [15]	Faster than both
Information transmission overhead	NDN devices' information management over Edge platform (KubeEdge)	NDN Publish/Subscribe [38] NDN-DM's push mechanism [39]	Equal to NDN-DM, lower than NDN Pub/Sub (4 times)
Information recovery capability	NDN devices' information management over Edge platform (KubeEdge)	NDN Publish/Subscribe [38] NDN-DM's push mechanism [39]	Ensure recovery while NDN Pub/Sub and NDN-DM can not

To summarize, in this work, our system shows the advantages of integrating NDN and Edge Computing infrastructure in disaster management, especially in the disaster response phase. Firstly, we demonstrated the benefit of using NDN over IP to achieve fast network convergence time in case of damaged network node replacement. This aspect has not been studied before in any works according to our research. Secondly, we designed a better mobility support mechanism for NDN compared with previously proposed solutions. Thirdly, we show edge computing effectiveness in NDN device information management. Compared with other recent proposed NDN management systems, the usage of edge computing platform not only reduces transmission overhead but also ensures recovery ability in case of intermittent network conditions. Finally, we deploy our proof-of-concept system on real platforms with KubeEdge as the chosen edge platform while most related works are only simulation studies.

6. Conclusions

This paper presented a deployment architecture of the NDN network over Edge Computing infrastructure to provide support for the disaster response phase. We showed a proof-of-concept system by implementing the architecture using the KubeEdge edge computing platform. Our system assists the disaster response phase by enabling emergency group communication and disaster information exchange through NDN device management. Moreover, the experimental and analytical results showed that our proposed architecture deals well with disaster challenges. It achieves faster network convergence time in case of node replacement over IP-based network, faster mobility handover time compared with MobileIP, and rendezvous NDN mobility method. Moreover, KubeEdge features lower information exchange transmission overhead and enables packet recovery capability in intermittent network conditions.

For future works, our system can be improved by bringing enhanced NDN features that have been proven by simulation in previous studies into real implementation. Moreover, an edge computing platform can also be utilized to provide support for NDN IoT devices since these devices can greatly help to collect a huge amount of information in disaster areas.

Author Contributions: All authors contributed to the study and wrote the article. M.-N.T. proposed the idea, designs, and performs the evaluation. Y.K. suggests directions for the detailed designs and

evaluation, as well as coordinating the work. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Government of Korea (MSIT) (No.2020-0-00946, Development of Fast and Automatic Service recovery and Transition software in Hybrid Cloud Environment) and under the ITRC (Information Technology Research Center) support program (IITP-2020-2017-0-01633) supervised by IITP.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. WHO/EHA. *Disasters and Emergencies Definitions Training Package*; Panafrican Emergency Training Centre: Addis Ababa, Ethiopia, 2002.
2. Chen, J.; Arumaithurai, M.; Fu, X.; Ramakrishnan, K.K. CNS: Content-oriented Notification Service for Managing Disasters. In Proceedings of the 3rd ACM Conference on Information-Centric Networking (ICN '16), Kyoto, Japan, 26–28 September 2016; pp. 122–131. [CrossRef]
3. Hannan, A.; Arshad, S.; Azam, M.A.; Loob, J.; Ahmed, S.H.; Majeed, M.F.; Shah, S.C. Disaster Management System Aided by Named Data Network of Things: Architecture, Design, and Analysis. *Sensors* **2018**, *18*, 2431. [CrossRef] [PubMed]
4. Shvartzshnaider, Y.; Ott, M. Design for change: Information-centric architecture to support agile disaster response. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 4025–4029. [CrossRef]
5. Jahanian, M.; Chen, J.; Ramakrishnan, K.K. Graph-based Namespaces and Load Sharing for Efficient Information Dis-semination in Disasters. In Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP), Chicago, IL, USA, 8–10 October 2019; pp. 1–12. [CrossRef]
6. Chen, J.; Xing, Y.; Ramakrishnan, K.K.; Jahanian, M.; Seferoglu, H.; Yuksel, M. ReDiCom: Resilient Communication for First Responders in Disaster Management. In Proceedings of the 2019 IEEE 27th International Conference on Network Protocols (ICNP), Chicago, IL, USA, USA, 8–10 October 2019; pp. 1–2. [CrossRef]
7. Jin, Y.; Tan, X.; Feng, W.; Lv, J.; Tuerxun, A.; Wang, K. MANET for Disaster Relief based on NDN. In Proceedings of the 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN), Shenzhen, China, 15–17 August 2018; pp. 147–153. [CrossRef]
8. Okamoto, K.; Mochida, T.; Nozaki, D.; Wen, Z.; Qi, X.; Sato, T. Content-Oriented Surveillance System Based on ICN in Disaster Scenarios. In Proceedings of the 2018 21st International Symposium on Wireless Personal Multimedia Communications (WPMC), Chiang Rai, Thailand, 25–28 November 2018; pp. 484–489. [CrossRef]
9. Koizumi, Y.; Yamamoto, Y.; Hasegawa, T. Emergency Message Delivery in NDN Networks with Source Location Verifi-cation. In Proceedings of the 2019 IEEE Globecom Workshops (GC Wkshps), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
10. Kim, S.; Urata, Y.; Koizumi, Y.; Hasegawa, T. Power-saving NDN-based message delivery based on collaborative com-munication in disasters. In Proceedings of the 21st IEEE International Workshop on Local and Metropolitan Area Networks, Beijing, China, 22–24 April 2015; pp. 1–6. [CrossRef]
11. Ogawara, T.; Kawahara, Y.; Asami, T. Information dissemination performance of a disaster-tolerant NDN-based distrib-uted application in disrupted cellular networks. In Proceedings of the IEEE P2P 2013, Trento, Italy, 9–11 September 2013; pp. 1–5. [CrossRef]
12. Deshmukh, R.; Zink, M. An information centric networking approach for sensor to vehicular network communication in disasters. In Proceedings of the 2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Rome, Italy, 9–11 October 2017; pp. 227–234. [CrossRef]
13. NLSR—Named Data Link State Routing Protocol. Available online: https://named-data.net/doc/NLSR/current/ (accessed on 24 November 2020).
14. Perkin, C.E.; Johnson, D.B.; Arkko, J. *Mobility Support in IPv6: RFC 6275*; IETF: Fremont, CA, USA, 2011.
15. Zhang, Y.; Xiz, Z.; Mastorakis, S.; Zhang, L. KITE: Producer mobility support in named data networking. In Proceedings of the 5th ACM Conference on Information-Centric Networking (ICN '18), Boston, MA, USA, 21–23 September 2018; pp. 125–136. [CrossRef]
16. Tran, M.-N.; Kim, Y. NDN-based Emergency Communication over Edge Computing Infrastructure. In Proceedings of the 2020 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Korea, 21–23 October 2020; pp. 353–358. [CrossRef]
17. NdnSim—NS-3 Based Named Data Networking (NDN) Simulator. Available online: https://ndnsim.net/current/ (accessed on 24 November 2020).
18. Dinh, N.-T.; Tran, M.-N.; Park, Y.; Kim, Y. An Information-centric NFV-based System Implementation for Disaster Management Services. In Proceedings of the 2020 International Conference on Information Networking (ICOIN), Barcelona, Spain, 7–10 January 2020; pp. 807–810. [CrossRef]

19. KubeEdge—An Open Platform to Enable Edge Computing. Available online: https://kubeedge.io/en/ (accessed on 24 November 2020).
20. Kubelet. Available online: https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/ (accessed on 24 November 2020).
21. SQLite. Available online: https://www.sqlite.org/index.html (accessed on 24 November 2020).
22. Etcd—A Distributed, Reliable Key-Value Store for the Most Critical Data of a Distributed System. Available online: https://etcd.io/ (accessed on 24 November 2020).
23. Modbus. Available online: https://modbus.org/ (accessed on 24 November 2020).
24. OPC Foundation—Unified Architecture. Available online: https://opcfoundation.org/about/opc-technologies/opc-ua/ (accessed on 24 November 2020).
25. Eclipse Mosquitto—An Open Source MQTT Broker. Available online: https://mosquitto.org/ (accessed on 24 November 2020).
26. KubeEdge Reliable Message Delivery. Available online: https://github.com/kubeedge/kubeedge/blob/master/docs/proposals/reliable-message-delivery.md (accessed on 24 November 2020).
27. Lim, H.; Ni, A.; Kim, D.; Ko, Y.-B.; Shannigrahi, S.; Papadopoulos, C. NDN Construction for Big Science: Lessons Learned from Establishing a Testbed. *IEEE Netw.* **2018**, *32*, 124–136. [CrossRef]
28. Zhu, Z.; Afanasyev, A. Let's ChronoSync: Decentralized dataset state synchronization in Named Data Networking. In Proceedings of the 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 7–10 October 2013; pp. 1–10. [CrossRef]
29. Ndn-autoconfig. Available online: https://named-data.net/doc/NFD/current/manpages/ndn-autoconfig.html (accessed on 24 November 2020).
30. Ndn-cxx: NDN C++ Library with eXperimental eXtensions. Available online: https://named-data.net/doc/ndn-cxx/current/ (accessed on 24 November 2020).
31. Named Data Networking Forwarding Daemon (NFD). Available online: https://named-data.net/doc/NFD/current/ (accessed on 24 November 2020).
32. IBR-DTN—A Modular and Lightweight Implementation of the Bundle Protocol. Available online: https://github.com/ibrdtn/ibrdtn (accessed on 24 November 2020).
33. Cerf, V.; Burleigh, S.; Hooke, A.; Torgerson, L.; Durst, R.; Scott, K.; Fall, K.; Weiss, H. Delay-Tolerant Networking Architecture. Available online: https://tools.ietf.org/html/rfc4838 (accessed on 24 November 2020).
34. Enhanced Interior Gateway Routing Protocol. Available online: https://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html (accessed on 24 November 2020).
35. Asabere, E.D.; Panford, J.K.; Hayfron-Acquah, J.B. Comparative Analysis Of Convergence Times Between OSPF, EIGRP, IS-IS and BGP Routing Protocols in a Network. *Int. J. Comput. Sci. Inf. Secur.* **2017**, *15*, 225–227.
36. Vukotic, I.T.; Scepanovic, S. Measurements of convergence time for RIP and EIGRP protocols. *Scr. Scientiarum Naturalium* **2011**, *2*, 71–83.
37. Wang, L.; Lehman, V.; Hoque, A.K.M.M.; Zhang, B.; Yu, Y.; Zhang, L. A Secure Link State Routing Protocol for NDN. *IEEE Access* **2018**, *6*, 10470–10482. [CrossRef]
38. Shang, W.; Gawande, A.; Zhang, M.; Afanasyev, A.; Burke, J.; Wang, L.; Zhang, L. Publish-Subscribe Communication in Building Management Systems over Named Data Networking. In Proceedings of the 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, 29 July–1 August 2019; pp. 1–10. [CrossRef]
39. Ali, Z.; Shah, M.A.; Almogren, A.; Din, I.U.; Maple, C.; Khattak, H.A. Named Data Networking for Efficient IoT-based Disaster Management in a Smart Campus. *Sustainability* **2020**, *12*, 3088. [CrossRef]