xDIoT: Leveraging Reliable Cross-domain Communication Across IoT Networks

Kounteya Sarkar, Sudip Misra and Mohammad S. Obaidat

***Abstract*—We propose xDIoT, a paradigm for reliable cross-domain communication across separated IoT network domains over some public network such as the Internet. Depending on use-case scenarios, several individual IoT domains, each consisting of heterogeneous end-devices (sensors and actuators) are deployed across geographical regions. When these domains need to communicate with one another they can use an intermediate public network like the Internet. To this end, a standard paradigm is required for such inter-domain communication over public networks to reduce latencies and prevent inconsistencies, which is absent in current deployments. With xDIoT we address this issue to provide a uniform communication paradigm. In xDIoT, each individual domain has an associated gateway access point (AP) acting as the bridge between the intra-domain IoT network and the external public network. These APs perform Domain Information Exchange through JSON format to gain knowledge about each other and use a generalized packet header structure to encapsulate all data flowing between these APs. Through the use of JSON data exchange and the proposed packet header, the APs can perform seamless inter-domain communication. Implementation and analysis show that xDIoT achieves about 10% improvement in total communication latency with 80% improvement in packet processing time at individual gateway APs. The proposed mechanism allows not only distributed IoT networks, but heterogeneous non-IoT frameworks such as campus LANs, P2P clusters and overlay networks to attain homogeneity and agreement in inter-communication. xDIoT uses open and readily available data formats and interfaces that makes it easily integrable and interoperable.**

I. Introduction

In this paper we propose xDIoT, a reliable cross-domain (or inter-domain) communication paradigm for heterogeneous Internet of Things (IoT) network domains spread across a public network like the Internet. Different IoT domains are able to communicate with each other seamlessly with low latencies through xDIoT, even though they are geographically far apart. Each of these domains has a number of IoT end-devices such as wireless sensor nodes and actuators connected with each other in a mesh architecture. A central access point (AP) acts as the communication gateway for all the individual IoT devices of one domain to connect to end-devices of other domains. For each IoT domain, the AP is connected to the public network on one side and the IoT intra-network on the other side. It thus forms the bridge between the IoT domain and the public network. xDIoT enables these domain APs situated across the public network to exchange information with each other reliably. This significantly reduces the complexity of inter-domain messaging achieving a uniform communication paradigm across public networks like the Internet.

IoT deployment uses several heterogeneous devices over a geographical area connected through various protocols and technologies [1]. The deployment regions may be logically similar, but physically separated. Each of these deployment regions form individual domains which may come under a common administration. For example, domains of similar IoT architecture may be deployed across smart grids that are physically separated from each other. When these domains are required to communicate with each other, the most logical option would be to use the public Internet as the intermediary carrier network between them. Not only similar domains, but dissimilar ones also can communicate with each other. On detecting a sudden power surge, a smart grid deployment may want to immediately message distant smart home deployments that come under the same management to switch off all electrical appliances. The types of devices, architecture, and protocols used in smart grids and smart homes are different according to their own use-cases. Therefore, a seamless end-to-end communication paradigm between physically separated and heterogeneous domains of IoT networks across an intermediary network is beneficial. This forms our primary motivation behind proposing xDIoT which can be employed by, say, IoT service providers for communication across their various physical deployment domains. Inter-domain communication is always an important and challenging aspect for various kinds of networks and forms a general class of network problems [2].

In xDIoT, we propose to offer a two-way approach towards providing reliable inter-domain communication. We assume that every individual IoT end-device is uniquely addressable within its own defined address space and that these devices are capable of communicating with each other across domains. As an example, all IoT nodes belonging to a service provider share a common address space. An access point (AP) is associated with each domain that is aware of the address space of each IoT nodes within its domain and serves as a gateway between the domain and the public network. Since these APs are also a part of the public network like the Internet, they are IP addressable (assuming the TCP/IP protocol stack). To this end, xDIoT provides reliable communication through the following approaches,

- **Domain information exchange.** The APs exchange basic information about their individual domains via JSON file format. This helps each of the gateway access points to have knowledge of other domains and their address spaces, as well as send out its own domain information to others.
- **Generalized IoT based packet header.** All the application layer data sent from an end-device in one domain to another end-device in another domain is encapsulated in a generalized packet header designed by us specifically

suited for IoT data. The encapsulation is done at the gateway APs and then they are sent as application layer data over the usual TCP/IP Internet to the destination domain's AP. The latter then de-encapsulates the header and sends it to the final IoT destination device.

In this work, we concentrate on the communication between the gateway APs and the format of messages they exchange. As individual domains are orchestrated differently according to their individual use-cases, the AP of each domain internally handles the heterogeneity and routing within its own domain. By providing an uniform agreement between APs regarding their internal composition, xDIoT makes cross-domain exchange significantly easier. The generalized packet header furthermore enables the intermediary carrier network devices (routers, firewalls etc) to identify IoT specific traffic as a separate application class.

II. Related Works

Heterogeneity is an inherent feature of IoT networks comprising of a myriad of different devices. Yildirim *et al.* [3] gives a generic overview of the different types of heterogeneity in IoT architecture. Communication between such heterogeneous domains present challenges. Villa *et. al* [4] presents Inter-Domain Messaging (IDM) to facilitate inter-domain communication across IoT and other networks. IDM bypasses cloud-based solutions by providing P2P inter-domain communication, but it requires an IDM router to be installed on the intermediary devices. xDIoT has no such requirement of separate installation of modules. Anand *et. al* [5] focuses on secure inter-domain IoT communication. Gyrard *et. al* proposes semantic ontology based M3 framework for cross-domain IoT applications, especially with respest to M2M communication [6]. In [7], Desai *et. al* provides IoT interoperability through semantic gateways in domains and a multi-protocol proxy architecture. Soursos *et. al* [8] further discusses on the interoperability issues in vertical IoT domains. The authors in [9] provide yet another secure architecture for federated IoT domains for handling trust management.

III. xDIoT:Cross-domain communication

xDIoT facilitates seamless cross-domain communication among physically separated heterogeneous IoT domains over a public network (we consider the Internet henceforth). Fig. 1 depicts the architecture of the scenario that we consider. 'A', 'B' and 'C' are individual IoT domains separated from each other geographically. Each of these domains have their own set of IoT devices configured in some architecture as the use-case may be, along with a gateway AP acting as a bridge between the internal and the external network (Internet). The APs run the full TCP/IP protocol stack to enable it to communicate over the Internet on one side. They also have the requisite functional blocks to communicate with individual IoT end-devices on the other side within its domain of control. In order for the APs to act as bridges between the Internet and IoT networks, they mainly perform the following two functions.

- **Packet Demultiplexing.** The APs act as a de-multiplexer for all the packets coming from the Internet destined towards various individual IoT nodes within its domain. Based on the final destination address the AP forwards the packet to the respective device through a route which it knows. This route is local to that domain and is known only by the AP of that domain.
- **Packet Multiplexing.** Opposite to the de-multiplexing, the APs also perform packet multiplexing for all traffic destined towards the Internet and originating from within the domain. As IoT packets reach the AP from the end-nodes of its own domain whose final destination is another IoT device in another domain, the AP encapsulates the packet in a specific header depending on the destination domain. It further encapsulates the packet in a general IP (IPv4 or IPv6) packet and sends it out on the Internet towards the destination domain.

To achieve the two functionalities, all the APs maintain a local database which maps individual IoT end-devices present within its domain to their respective address spaces, along with the local route information from the AP to that device. This database is created initially when the domain is set up updated continuously as and when new devices are placed or existing ones are removed. Due to the heterogeneity of the IoT network, there can be many types of devices with their own configurations and addresses. The device-database is required to capture all these heterogeneity details as well so that the AP can perform the multiplexing efficiently. Our main consideration in this paper being inter-domain communication, we do not delve into the creation and working of this database, which captures the intra-domain actions.

A. Domain Information Exchange

The first step towards cross-domain communication is for the APs to exchange information about their own domains with each other. We propose to use the JSON file format, which is widely supported by any computing device. In this phase, the APs share the information required for communication across domains. Fig. 2 shows a sample JSON exchange between two gateway APs.

The domain information exchange contains primarily two major information, data about one's own domain and information about the presence of other domains in the network that it knows about. As shown in Fig. 2, the "self_domain" object contains information about the former while the "other_domains" object contains data about the latter. In xDIoT, each distinct IoT domain across the Internet is identified uniquely by the IPv4 (or IPv6) address of the domain gateway AP. This IPv4 address enables the domain to be accessed across the Internet by other domains.

Regarding information about one's own domain, the JSON file contains a "domain_id" which is an unique numeric identification of the domain, "domain_type" indicating the functional nature of the IoT domain (sensing, actuation or mixed), the IP address of the gateway AP, and the number of individual IoT devices within the domain. It includes the information

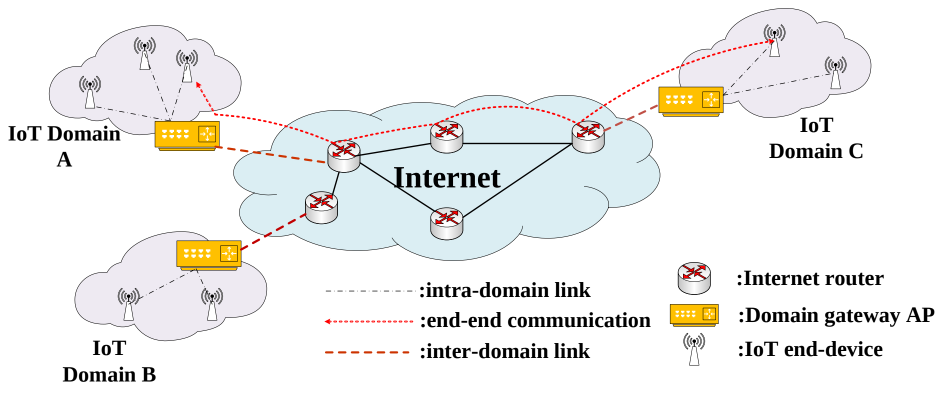


Fig. 1: Typical network architecture for IoT cross-domain communication over the Internet

```
{"self_domain": {
  "domain_id": "number",
  "domain_type": "sensor",
  "domain_AP_address": "192.168.1.2",
  "number_of_devices": "2",
  "end_devices": [
      {
      "type": "temperature",
      "status":"active",
      "local_address":"AAAAAA"
      },
      {
      "type": "pressure",
      "status":"active",
      "local_address":"AAAAAB"
      }]
  },
  "other_domains":{
  "number_of_known_domains":"2",
  "domain_information":[
  {
  "domain_AP_address": "192.168.1.3",
  "domain_id": "88",
  },
  {
  "domain_AP_address": "192.168.1.4",
  "domain_id": "90",
  }]
 }}
```

Fig. 2: Sample domain information exchange in JSON

about the IoT nodes present within the domains, namely their type (temperature or pressure type sensor), its activation status (active, sleeping, deactivated) and the ”local_address” of each end-device, which is the unique local intra-domain address given to every end-device by the domain AP. In xDIoT, every IoT end-device can be uniquely identified globally across the network by the concatenation of the domain identifier with the node’s own local_address within that domain. In the example given in Fig. 2, three attributes are shown corresponding to every end-device.

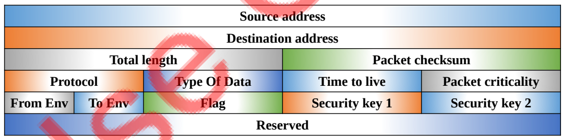


Fig. 3: Generalized Packet Header

B. Generalized IoT packet header

In xDIoT, we propose a generalized packet header for IoT data for transport across the Internet. Fig. 3 shows the structure of the header, which we have designed taking motivation from structures of existing protocols. It enables the APs to understand the type and nature of the application layer IoT data that it encapsulates, allowing them to take appropriate subsequent actions. Whenever cross-domain data exchange is required between end-devices, the source domain AP collects the data from devices within its domain and encapsulates them with the proposed header. It sets the various header fields based on the type of data and the source and destination node address. The source AP then sends out the packets with this header over the TCP/IP stack towards the destination domain (multiplexing). On receiving the packets over the Internet, the destination AP de-encapsulates the header to understand the packet contents to take subsequent actions with the data, like forwarding it to the ultimate destination IoT device in its domain (de-multiplexing). Since our proposed header lies in the application layer over the base IoT data, it can be used independently irrespective of any bottom layer network protocols being used (TCP, UDP, IPv4, IPv6 among others). It ensures hassle free delivery of the cross-domain data across any public network.

Table I gives the description of each of the header fields as shown in Fig. 3. The **source** and **destination addresses** refer to the unique addresses of the source and the destination IoT end-devices. Since end-devices from one domain have to be addressable from other domains, the complete address format comprises of two parts, a leading part denoting the domain followed by the

TABLE I: Summary of different header fields

Header field	Field length (in bits)	Field explanation
Source address	32	Represents the source of the packet
Destination address	32	Represents the destination of the packets
Total length	16	Represents the total length of the packet
Packet checksum	16	Represents packet content plus header checksum
Protocol	8	Represents the underlying IoT protocol
Type of data	8	Represents the type/nature of packet data
Time to live	8	Represents the maximum number of hops the packet is expected to live
Packet criticality	8	Represents the packet data criticality
From Environment	4	Represents whether packet data originated from environment
To Environment	4	Represents whether packet data is directed towards environment
Flag	8	Flag bits, left open for implementation
Security key 1	8	Represents the encryption/decryption key
Security key 2	8	Represents the encryption/decryption key
Reserved	32	Reserved for any future use

part denoting individual devices within that domain. Specifically the complete address is of the form *AA:BBBBBB*, where each 'A' and 'B represent a nibble of 4 bits. The one byte long 'AA' is the unique domain specific address and the three bytes long 'BBBBBB' represent the individual addresses of each device within a domain. The complete address uniquely identifies both the domain as well the individual devices in it. The addresses are kept 32 bits long in total length, allowing 2^8 different possible domains across a public network and 2^{24} possible end IoT devices within each domain.

The **total length** represents the length of the header and the data together. **Packet checksum** is the standard security checksum as is present in other headers also. **Protocol** signifies any underlying IoT specific protocol that the source domain is using, for example MQTT and CoAP. **Type of Data** signifies the type of data that the packet carries, for example sensor value, topology information, management information, actuation command and others. It is useful in logging activities for future audit and maintenance. **Time to Live** denotes the number of devices in a multi-hop routing that the data can traverse at most before it becomes obsolete. It helps the carrier network to fix the routing path accordingly. **Packet criticality** determines the criticality of the data that the packet contains. IoT data often are real-time in nature carrying critical sensor information. The **From** and **To Environment** fields denotes whether the packet data are taken from the physical environment (sensor values) or are destined towards it (actuator values) along with the specific sub-type of the environment being considered (16 types of physical environments can be considered with 4 bits). The two **Security Keys** are for the end IoT devices to perform initial key exchange for encryption and decryption along with the cryptographic algorithm being used. IoT devices are mostly low-power computation devices, hence they cannot sustain heavy cryptography with large key sizes. Thus we have kept smaller key sizes, but have given the option to have two separate keys to make the encryption stronger. Finally, the **Reserved** field is kept blank for further future use.

Algorithm 1: Behavior of each domain AP with respect to domain information exchange

```
1  for Each new domain that joins do
2      Manually configure the domain AP with knowledge of one existing AP;
3      if new AP is the only one domain present then
4          Do Nothing ;
5      else
6          New AP connects with a known domain;
7          It learns of other domains ;
8          Initiates exchange with all other domains ;
9      Update local information base ;
10 for Each existing domain that leaves the network do
11     The domain AP informs all other domains ;
12 for All existing domains present do
13     if AP learns of newly joined domains OR AP learns about removal of domains OR domain architecture of AP changes then
14         Initiate exchange with all other APs ;
15         Update local information base ;
```

C. Gateway AP Behavior.

When a new domain joins the network, its AP is initially given manually the address of the gateway AP of at-least one other domain already present across the Internet. If it is the first domain to be present, then it does nothing (since there is no cross-domain communication). If not, the former AP then performs the domain information exchange with the latter one with JSON. Through this, it learns the existence and addresses of other APs that are also present across the network. It performs domain information exchange in the same way with all these other domains as well. Whenever any AP learns the existence of other new domains, or removal of other domains, or due to major architectural modification of its own domain, it performs information exchange with all other APs that it knows of. This ensures that all the domains eventually converge with time on the consistent and relaible knowledge of each other, just like knowledge of route updates in BGP in the Internet. Algorithm 1 shows the behavior of the APs with respect to domain information exchange.

Once the AP has necessary information about other domains, it is ready to perform message exchange. It encapsulates any IoT data that it receives from within its own domain with the proposed packet header and sends it out to the Internet. The source AP knows both the unique domain address and the IP address of the destination domain AP from the information exchange. It also consults the local device database that it has and accordingly sets the header fields accordingly. The destination AP de-encapsulates the header, validates the required fields and chooses the appropriate routing path to send the data to the final end-device within its domain. Algorithm 2 shows the action of the APs with respect to the generalized headers (multiplexing and de-multiplexing).

Algorithm 2: Multiplexing and de-multiplexing functionalities of the APs

```
1  for Each gateway AP belonging to a domain do
2      if new IoT message arrives from within the domain
         then
3          Check contents  final destination of the message ;
4          Consult local database for destination address ;
5          Encapsulate the data in the generalized header ;
6          Send it out over the Internet ;
7      else if New packet arrives from the Internet then
8          De-encapsulate  analyze the generalized header;
9          Verify destination IoT device to be within the
             domain ;
10         Forward the data to the ultimate IoT end-device ;
11     Repeat process ;
```

IV. Performance Analysis

By providing a standard approach towards cross-domain communication, xDIoT achieves reliability, reduction in processing time at gateway APs and reduction in the total communication latency across the domains. Since IoT networks are heterogeneous, in the absence of any general agreement, the domains spend additional computational resources and time whenever any inter-domain communication is required. This problem is accentuated when number of domains increases. The total latency of communication during a cross-domain transfer is given by equation 1, where T_α represents the total cross-domain latency, t_{domain} is the latency due to intra-domain routing, t_β is the processing latency at the AP due to heterogeneous architecture and $t_{Internet}$ is the Internet transfer latency from source AP to destination AP. We implement xDIoT on a lab scale architecture sending messages from an IoT node in one domain to another domain through intermediary laboratory IP-based LAN (Fig. 4). Our task in xDIoT is to bring down t_β since the other two terms (t_{domain} and $t_{Internet}$) depend heavily on the architecture of IoT domains and standard Internet best-effort delivery. In our test setup (Fig. 4), we keep the t_{domain} and $t_{Internet}$ as fixed values and perform the analysis only on t_β and T_α. We note that t_{domain} being the intra-domian latency between end IoT node and its domain AP will remain same with or without xDIoT, and $t_{Internet}$ is simply dependant on the carrier network over which we have no control. So it suffices to have the analysis on t_β and T_α only.

$$T_\alpha = t_{domain} + t_\beta + t_{Internet} \quad (1)$$

Fig. 5 shows the total cumulative end-end latency in a cross domain communication for different number of domains considered, with each domain performing information exchanges

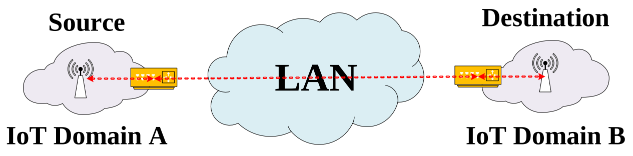
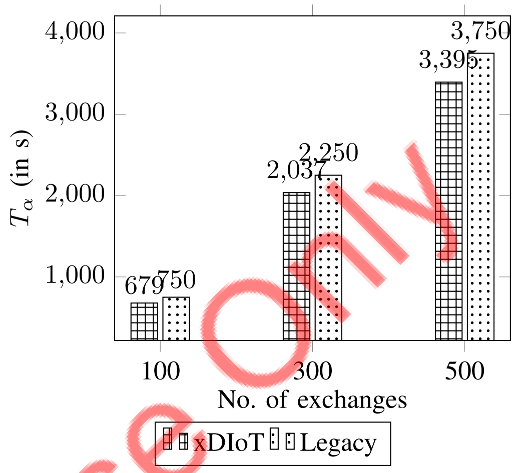
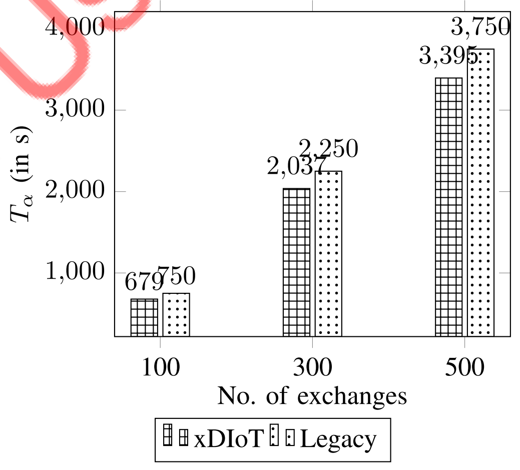


Fig. 4: Laboratory test architecture



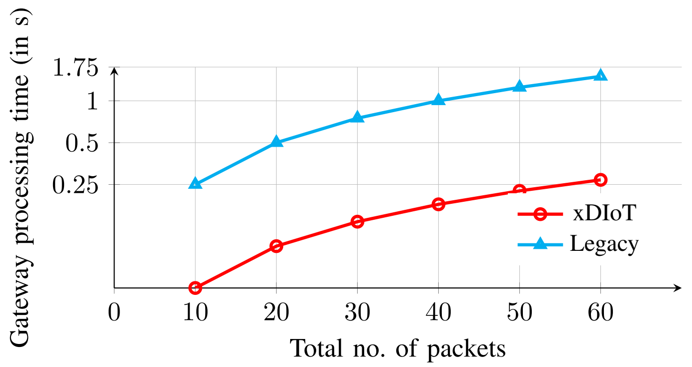
(a) No. of domains = 10



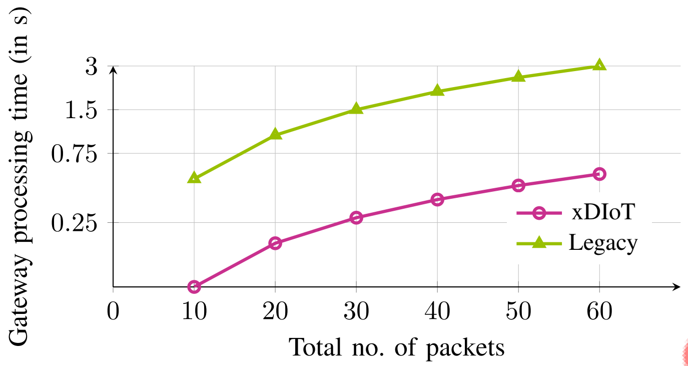
(b) No. of domains = 20

Fig. 5: Total time (T_α) comparison with no. of cross-domain exchanges

multiple times. Building upon the test architecture of Fig. 4, we have simulated the effect of increased number of domains through equivalent software modules. Fig. 6 shows the cumulative gateway processing time t_β for the similar experiment with different number of domains and multiple information exchanges. Comparison was done between our proposed xDIoT architecture and a legacy cross-domain communication lacking any generalized paradigm. In the case of legacy communication, we have considered that each pair of domains uses its own pre-mediated mechanism to share information with one another. For our simulation, we have orchestrated this pre-mediated scheme as 'broadcast-and-learn' whereby a domain has no knowledge about about other domains and sends a broadcast request over the network to learn every time a new packet transfer is required. Both Fig. 5 and Fig. 6 show that xDIoT achieves about 10% reduction in total



(a) Number of domains=5



(b) Number of domains=10

Fig. 6: Cross-domain information processing time at gateway APs (t_β)

V. Conclusion

This paper proposes xDIoT, a paradigm for uniform and reliable cross-domain communication for IoT networks across intermediary public networks. The gateway APs serve as the bridge between the internal IoT network and the external public network. Through the Domain Information Exchange using JSON and the generalized packet header, the APs perform seamless data exchange with each other. Furthermore, through their multiplexing and de-multiplexing functionalities, the APs are able to send and receive packets to and from IoT end-devices within their domains. The overall effect is that an end-device from one domain can interact reliably with the same in another domain through their respective gateway APs. xDIoT achieves a reduction of about 10% in total communication latency and as much as 80% reduction in individual gateway AP processing time. The immediate future plan of work involves the security analysis of xDIoT to make the inter-domain exchanges robust, secure and fail-proof against malicious attacks.

Through this work we aim to solve a crucial issue for seamless communication among distributed IoT networks by bringing in a homogeneous scheme agreeable to every domains. It enables convenient interoperability between different stakeholders, for example different businesses, each of which can deploy domains and networks with vendor-specific proprietary protocols and settings, but require a suitable inter-communication mechanism between the domains for data exchange. This work can be further extended not only for inter-operability and communication among distant IoT domains but non-IoT architecture such as campus LANs, MPLS networks and distributed P2P clusters. Even overlay networks can take inspiration from our proposed idea to formulate and implement communication schemes among heterogeneous network clusters.

References

[1] M. I. Hussain, “Internet of Things: challenges and research opportunities,” *CSI transactions on ICT*, vol. 5, no. 1, pp. 87–95, 2017.

[2] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, “Internet inter-domain traffic,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 75–86, 2010.

[3] G. Yıldırım and Y. Tatar, “On wsn heterogeneity in iot and cpss,” in *2017 International Conference on Computer Science and Engineering (UBMK)*, pp. 1020–1024, IEEE, 2017.

[4] D. Villa, O. Acena, F. J. Villanueva, M. J. Santofimia, S. Escolar, X. del Toro Garca, and J. C. Lopez, “Idm: an inter-domain messaging protocol for iot,” in *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pp. 8355–8360, IEEE, 2017.

[5] A. Anand, A. Galletta, A. Celesti, M. Fazio, and M. Villari, “A secure inter-domain communication for iot devices,” in *2019 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 235–240, IEEE, 2019.

[6] A. Gyrard, S. K. Datta, C. Bonnet, and K. Boudaoud, “Standardizing generic cross-domain applications in internet of things,” in *2014 IEEE Globecom Workshops (GC Wkshps)*, pp. 589–594, IEEE, 2014.

[7] P. Desai, A. Sheth, and P. Anantharam, “Semantic gateway as a service architecture for iot interoperability,” in *2015 IEEE International Conference on Mobile Services*, pp. 313–319, IEEE, 2015.

[8] S. Soursos, I. P. Žarko, P. Zwickl, I. Gojmerac, G. Bianchi, and G. Carrozzo, “Towards the cross-domain interoperability of iot platforms,” in *2016 European conference on networks and communications (EuCNC)*, pp. 398–402, IEEE, 2016.

[9] B. Anggorojati and R. Prasad, “Securing communication in inter domains internet of things using identity-based cryptography,” in *2017 International Workshop on Big Data and Information Security (IWBIS)*, pp. 137–142, IEEE, 2017.