A Comparative Study of LISP and Traditional Routing Protocols within an Autonomous System: Experimental Evaluation and Programmatic Management

Erika Abigail Katonová[1], Peter Feciľak[1], Rastislav Petija[1]

[1]Technical University of Košice, Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Letná 1/9, 040 01 Košice, Slovakia, erika.abigail.katonova@tuke.sk, peter.fecilak@cnl.sk, rastislav.petija@cnl.sk,

Abstract: This article aims to present the results of experimental verification of the suitability of creating network infrastructures using software-defined approaches. During the experiments, the attention was focused on the LISP routing protocol and the possibilities of managing the configuration of network devices through a programmatic approach using YANG models and the RESTCONF protocol. The results of the experiments confirmed that the LISP protocol is more efficient than traditional routing protocols in terms of bandwidth consumption, memory requirements and convergence speed. To demonstrate the possibility of managing and monitoring the network infrastructure using a programmatic approach, a controller was implemented in the form of a desktop application using the C# programming language. The implemented controller included functionalities for registration of managed devices, basic device configuration, basic and advanced management of LISP protocol configuration, monitoring of network devices by SNMP protocol, connectivity testing done by ICMP protocol, data classification by machine learning models, and management of OpenFlow protocol data flow configuration.

Keywords: Software-defined networks; LISP protocol; YANG models; RESTCONF protocol; OpenFlow protocol; network controller

1 Introduction

The constant increase in the amount of end devices and dynamic evolution of network infrastructures expose two interconnected problems that traditional networking approaches are increasingly unable to address. The first is the absence of centralized, programmatic management: traditional network device configuration is performed manually, is prone to human error, cannot integrate effectively with external services, and does not scale with the pace of change that

modern environments demand. The second problem is the growing number of routed networks itself, driven primarily by trends such as server virtualisation and micro-segmentation. As the number of networks within an autonomous system grows, traditional Interior Gateway Protocols (IGPs) such as OSPF, EIGRP, and IS-IS face compounding inefficiencies: topology changes trigger frequent and computationally expensive recalculations that must be executed on every router in the domain, large routing update messages consume significant bandwidth, and every router is required to maintain a complete routing table regardless of actual traffic patterns, placing a heavy and growing burden on memory. These two problems are related, both stem from the distributed, device-centric nature of traditional network architectures, and both call for a more centralised and programmable approach to network design.

Software-defined networking (SDN) offers a solution to both challenges. The separation of the control plane from the data plane enables centralized, automated management of network device configurations through modern programmatic interfaces, allowing prompt responses to changes. Additionally, the programmatic framework enables the deployment of new services, enhanced security of the environment, and improved integration with external services and systems [1][2]. For the routing problem, the LISP protocol, adopted within SDN environments as an overlay, pull-based routing solution, offloads routing intelligence to a centralised mapping system, eliminating the need for network-wide flooding and full routing table maintenance on every device. While the LISP protocol has been studied as an alternative routing solution, the overwhelming majority of existing research evaluates it exclusively in comparison with the Border Gateway Protocol (BGP), an Exterior Gateway Protocol (EGP) operating between autonomous systems. The behaviour of LISP within an IGP environment, specifically its convergence time, bandwidth consumption, and memory requirements relative to OSPF, EIGRP, and IS-IS, has not been systematically investigated. This paper addresses that gap by experimentally verifying the efficiency of the overlay-based, centralised routing approach offered by LISP in an IGP context, and by demonstrating programmatic network management through a purpose-built controller, confirming that LISP is a viable candidate for deployment within autonomous systems in large and dynamically changing network environments.

Several related works have explored SDN architecture and the LISP protocol in various contexts. The general SDN architecture and its control/data plane separation are described in [3][4][5], and surveys of SDN controllers are provided in [6][8]. Applications of LISP have been documented in aviation [15][16], IoT edge computing [18], 5G small cell networks [19], and data centre workload management [20], while RESTCONF/YANG-based configuration management is examined in [2]. It is proven that LISP has significantly influenced these industries, helping overcome key challenges and delivering effective solutions.

The remainder of this paper is structured as follows. Section 2 reviews the key architectures and protocols of software-defined networking, including the SDN

general architecture, the OpenFlow and LISP protocols, and a comparative analysis of traditional versus SDN routing approaches. Section 3 surveys applications of the LISP protocol across various industries to motivate its practical relevance. Section 4 presents the design of the experimental environment, including the proposed network topology, experimental scenarios, and the network controller. Section 5 describes the methodology and implementation details of both the network topology and the controller. Section 6 evaluates the experimental results for memory consumption, bandwidth usage, convergence time, and the benefits of the controller-based management approach. Section 7 provides a scalability analysis and theoretical extrapolation of the experimental findings to larger-scale topologies. Finally, the conclusions summarise the findings and directions for future research.

2 Architectures and protocols of SDN

The architecture of software-defined networks is based on the separation of the control plane from data plane of network devices, which enables the centralization of control across many devices [3]. By default, administrators interact with the control plane through the REST API. The network device itself can subsequently be managed from the control plane using the protocols such as RESTCONF, NETCONF, and OpenFlow protocols. For routing purposes, most solutions implement the LISP protocol.

2.1 SDN Architecture

There are numerous solutions to software-defined networks, however, all of them are based on the general architecture defined also by the authors of the publication [4]. The following Figure 1 illustrates this general architecture of SDN solutions.

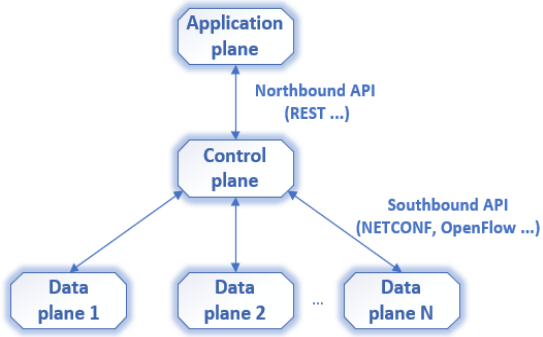

Figure 1

General architecture of SD solution [4]

The figure displays three interconnected planes [5]. The application plane represents external applications that can interconnect with the control. These are applications and tools for monitoring, collaboration, troubleshooting, GUIs, etc. The control plane is implemented by default in the form of a centralized controller that can evaluate requests from network devices, make decisions, and send control instructions back to the network device. The data plane represents network devices themselves and serves to ensure the forwarding and reception of transmitted data.

2.2 OpenFlow protocol

The OpenFlow protocol can be classified as a communication protocol that enables centralized control of network devices' behaviour from the controller. Based on the identification of data flows, the controller establishes and defines the rules for their processing [6]. It is also important to mention that this protocol does not distinguish between switching and routing. Both of these processes are operated by the controller, and devices receiving the data are generally referred to as OpenFlow switches. There are numerous OpenFlow controllers, most of them covered by the authors of this article [7], however, a few of the most popular being OpenDayLight, POX, NOX, Ryu, Floodlight, and ONOS [8].

2.3 LISP protocol

The LISP (Locator ID Separation Protocol) is a routing protocol that separates the roles of device identity and network location, assigning each endpoint two distinct addresses: an Endpoint Identifier (EID), which identifies the device, and a Routing Locator (RLOC), which identifies its current network attachment point. The formal definition of the protocol is specified in RFC 9301 [11]. LISP is notably adopted as the core routing mechanism in Cisco's SD-Access solution [9], and its effectiveness within that architecture, which is organised into eight functional modules, is further analysed in [10]. To illustrate the interaction between LISP components in the context of this study, Figure 2 presents an original diagram of a representative LISP network topology. The key components of the LISP architecture, as defined in RFC 9299 [12] and visible in the topology above, include:

- Mapping server: Device storing the global mapping database between EID-RLOC.
- ITR (Ingress Tunnel Router): Entry point for tunnel creation. It receives data from end devices and encapsulates them in LISP format. It transfers the data encapsulated in this way to the end ETR device of the tunnel. This device also transmits requests to obtain mappings between communicating end devices when necessary.

- ETR (Egress Tunnel Router): Exit point for tunnel creation. It receives LISP messages, which are decapsulated and delivered in their original form to the end device. ETR's purpose is also to register the mapping between EID-RLOC to the mapping server after connecting to the end network.
- xTR: Fulfills the role of both ITR and ETR.
- EID (Endpoint Identifier): IP address identifying the end device.
- RLOC (Routing Locators): One of the globally routable addresses of the ETR device. One RLOC can be assigned to multiple EID addresses.
- PETR (Proxy ETR): Sends data from the LISP domain to external networks. Removes the LISP header.
- PITR (Proxy ITR): Receives data from external networks, adds a LISP header, and sends it to the target device in the LISP domain.
- PxTR: Fullfils the role of both PETR and PITR.
- External device: A device outside the LISP domain.

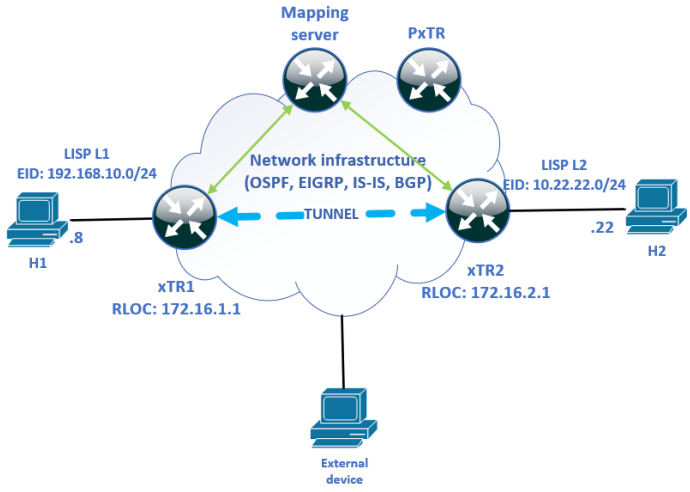

Figure 2

Custom experimental LISP topology

2.4 Comparison of traditional and SDN routing approaches

OSPF, IS-IS, EIGRP, and BGP are standard protocols used for routing within traditional network infrastructures [13]. With these protocols, both the control and data planes reside directly on the network device [14]. On the other hand, software-

defined network infrastructures use the LISP and OpenFlow protocols for data routing, which feature a separate control plane (located on the mapping server/controller) distinct from the data plane. The following Table1 compares traditional and modern approaches to data routing:

Table 1
Comparison of traditional and modern approaches to data routing

		Protocol		
		OSPF, IS-IS, EIGRP, BGP	OpenFlow	LISP
Feature	Maintain connection	YES	YES	NO
	Controller based	NO	YES	YES
	Possibility of automation	NO	YES	YES
	RESTCONF NETCONF	YES	YES	YES
	Processor requirements	HIGH	MEDIUM	LOW
	Memory requirements	HIGH	MEDIUM	LOW
	Speed of convergence	FAST	FAST	MEDIUM
	Tunnel based	NO	YES	YES

It should be noted that the lower processor requirements attributed to LISP in the table above need careful explanation since LISP performs per-packet encapsulation, which does introduce some additional processing at the data plane level. However, this overhead is constant, predictable, and in modern network hardware is typically handled by dedicated forwarding ASICs rather than the general-purpose CPU. The significant processor savings that LISP offers relative to traditional protocols are realised at the control plane level, where the difference in behaviour is fundamental. When a topology change occurs in a network running a traditional IGP such as OSPF, every router in the domain must independently execute a multi-step process: it first calculates which update messages need to be generated and sent to all neighbours, and then, upon receiving updates from its own neighbours, must rerun the full Shortest Path First (SPF) algorithm to recompute the entire forwarding tree and install new routes. This sequence of computation and flooding is replicated simultaneously on every single router in the autonomous system. In the case of IS-IS, the process is analogous, and EIGRP triggers the DUAL algorithm re-convergence across the affected portion of the topology. In contrast, when a network change occurs in a LISP domain, only the border xTR device that detected the change is involved in the control plane update: it sends a single registration message to the centralised Map Server, which updates the EID-to-RLOC mapping accordingly. No other device in the network performs any recalculation, no flooding occurs, and the rest of the infrastructure is entirely unaffected. The control plane processing is, therefore, localised to two devices (xTR and Map Server) regardless

of how large the network is. This is the reason that, when viewed globally across the entire network, LISP results in substantially lower aggregate CPU utilisation than traditional IGP protocols, despite the per-packet encapsulation overhead at the data plane.

3 LISP across various industries

The LISP Protocol has gained popularity in various industries as a flexible solution to address modern networking challenges. This chapter focuses on the different solutions and studies that incorporate this protocol.

In the article [15], the LISP protocol is used to implement distributed mobility in aviation networks. The research builds on a 2019 solution that suggested using Mobile IPv6 (MIPv6) Home Agents as regional mobility anchors in ICAO regions. However, the solution had limitations because commercial MIPv6 solutions lacked key features and couldn't operate as distributed anchors. To overcome these challenges, the authors suggest replacing MIPv6 with LISP-based regional mobility anchors. This approach keeps the original architecture intact while offering a more practical and efficient solution.

Another research in the aviation field [16] explores the application of the LISP protocol to improve aeronautical communications by addressing challenges like packet loss in multi-link environments. By integrating LISP and Random Linear Network Coding (RLNC) within SDN architecture, the researchers want to improve reliability and reduce delivery delays across satellite and terrestrial radio technologies.

Authors of the study [17] demonstrate a way in which the LISP protocol can solve key security issues in mobility concepts in the topic of access control lists (ACL). Traditional network frameworks often fail to transfer ACL rules when devices move between network segments. By proposing a dynamic method for transferring ACL rules in LISP environments, this research ensures that security protections remain synchronized regardless of endpoint movement. The study successfully encapsulates and transfers ACL rules without requiring additional processes. The conducted experiments have proven the hypothesis set by the authors, showing that synchronized security can be maintained in LISP networks.

The Internet of Things (IoT) sector also benefits from LISP, since authors in this study [18] highlight the effectiveness of using this protocol to optimize routing and manage mobility in mobile networks with edge computing. The proposed control system keeps user sessions uninterrupted by tracking service locations and connection details as users switch between different network access points. The analysis demonstrates that the system reduces costs and latency by minimizing message exchanges and delays compared to traditional methods. The authors

conclude that LISP is a potential solution for managing mobility and optimizing resources in dynamic network environments.

Researchers Fafolahan and Pierre [19] propose a different perspective to manage mobility in dense 5G Small Cell (SC) networks using LISP as a part of the solution. The solution addresses the challenges of heavy traffic, delays, and packet loss by dividing the network into localized service areas managed by dedicated anchors. The results demonstrate that the solution brings significant benefits. It reduces the processing load on the network by 90% compared to existing LISP-based mobile protocols.

Article [20] demonstrates that LISP can help manage how IP addresses are used in data centers, therefore it is more flexible and efficient. The authors also explain how important it is to distinguish between the "edge" (where devices connect to the network) and the "core" (the central part of the network). This separation helps improve how IP routing works and provides a way to support temporary or moving the workload in data centers, like VMs that may only be active for a short time or that may move frequently. And since LISP has a system for mapping EIDs to their corresponding RLOCs, it provides a new solution to manage network connections at the edge of the data center. This means that the network can dynamically handle VMs as they move around.

LISP's scalability has made it a great impact across the mentioned industries, addressing various challenges and providing solutions, mostly because of the function to separate the ID of the device from its locator ID. Therefore, the following work has meaning and purpose, and LISP can serve as a foundation for future advancements.

4 Proposal of the experimental environment

The experimental environment will contain a network topology suitable for LISP deployment, along with a management controller to manage the configuration of network devices. The configuration of the LISP protocol will be executed using the authoritative guidance provided in reference from Cisco [21].

4.1 Topology design for LISP protocol experiments

By analysing traditional routing protocols and the LISP protocol, it is possible to conclude that the LISP routing protocol may offer greater efficiency in terms of bandwidth conservation and memory requirements. The proposed topology has to be able to perform experiments to confirm or disprove the stated conclusions of the analysis. The topology must allow the deployment of both traditional routing protocols and the LISP protocol. In addition to the basic routing, the aim of the

experiments will be to confirm the functionality of other utilities of the LISP protocol, which include load balancing and communication with external networks.

From the perspective of the LISP protocol, for the purposes of basic functionality, it is necessary to deploy the Mapping Server devices and a pair of xTR devices, between which a tunnel will be created. To verify the possibility of load balancing, at least one xTR device has to be connected to the network using a pair of cables. Another necessary device is PxTR, which will enable communication with external networks. The last type of devices will be ordinary routers, configured with the traditional routing protocols, to ensure communication between routers that implement LISP functionality. Figure 3 on the next page, builds upon the conceptual framework depicted in Figure 2 and illustrates the detailed proposed experimental topology designed for this study. This configuration contains all necessary device types, where the devices marked R1-R6 represent traditional routers, which form the so-called Campus fabric.

4.2 The proposition of scenarios for experiments

Steps in the following list propose designed scenarios to verify the functionality and suitability of the LISP protocol:

1. Creating a topology and configuring addresses based on Figure 3 below.
2. Running traditional routing protocol on devices R1-R6. The traditional routing protocol will also run on edge LISP devices but will only function on the interfaces pointing to R1-R6. Completing this step will create a fully functional monitorable network.
3. Starting the process of capturing packets at the output of xTR devices, input to the Mapping server, and selected links between R1-R6.
4. Notification of one and later several LISP networks to the common routing protocol, with subsequent observation of processes related to the updating of routing information. In this way, it is possible to find out information about the number of transferred messages, their size, and bandwidth consumption. To further analyse the behaviour of traditional routing protocols, it is possible to save the data in the form of .pcap files.
5. Removing the configuration done in step 4, which will then allow the administrator to experiment with the LISP protocol.
6. Notification of the LISP networks to the LISP protocol. In this step, it is again possible to save captured messages in .pcap files for later analysis.
7. Evaluation of the achieved results and their mutual comparative analysis with a focus on the consumption of memory and bandwidth usage.

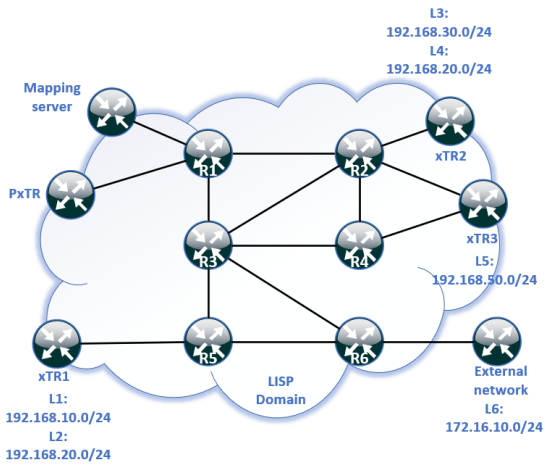

Figure 3

Proposed topology

This scenario for measuring the time required to achieve convergence involves using the Wireshark tool on end devices and generating ICMP messages. During data transmission, one of the links essential for the transfer is intentionally disabled. The time interval between receiving the last successful response and the first successful response after the outage represents the sought information. This scenario applies to traditional routing protocols. For LISP, the convergence time refers to the interval between sending the first transmission of the initial unsuccessful request and the receipt of a LISP message containing the desired EID-RLOC mapping.

4.3 The proposal of the network controller

To verify the suitability of the configuration management and monitoring of network infrastructures using a programmatic approach, it was necessary to design functionalities for the controller implemented in the next step. The following functionalities are considered significant:

1. Registration, display, and removal of managed devices.
2. Basic equipment configuration, including tasks such as reading and modifying device names using the RESTCONF protocol and handling data in JSON format.
3. Configuration management with a more complex structure. In this case, these are functions for managing the configuration of the LISP protocol. Within this protocol, these are functions for the basic configuration of the LISP protocol, which includes registration of IPv4 and IPv6 mappings

between EID and RLOC, implementing load balancing, enabling communication with external networks, and supporting mobility.

4. In order to demonstrate alternative methods for network management and monitoring, an additional functionality involves working with the SNMP protocol. This functionality enables the retrieval of statistical data about the processor utilization, which will be displayed in the form of a graph.
5. A function using the ICMP protocol is implemented to test the availability of end devices.
6. For the purpose of demonstrating advanced data analysis, a machine learning-based packet classification function is proposed.
7. The final functionality of the controller involves working with the OpenFlow protocol, enabling reading and modification of data flow records.

The 6th functionality is included as a proof-of-concept for future work, demonstrating that the controller architecture is extensible to support advanced network analytics. The primary contribution of this feature is to show that ML-based traffic classification can be practically integrated into a RESTCONF/YANG-oriented network management platform. The workflow implemented in this proof-of-concept is as follows: during the experiments, all network traffic was captured at the relevant interfaces and stored as .pcap files. These capture files were subsequently used as the source dataset for model training: packets were extracted, decoded, and labelled according to their protocol type (ICMP, TCP, UDP) based on header fields including protocol identifier, packet size, source and destination port numbers, and inter-arrival time. The labelled dataset was used to train a multiclass classification model using the ML.NET library. Once trained and integrated into the controller, the model is able to classify incoming packets in real time by their protocol type and traffic category. The inclusion of this feature demonstrates that future iterations of the controller could leverage traffic captured directly from managed network devices to continuously retrain classification models, enabling intelligent, data-driven network management decisions.

5 Methodology for design and implementation

The implementation of the experimental environment consists of the creation of the network infrastructure and the implementation of the network controller in the form of a desktop application.

5.1 Creation of the network topology

Design in Figure 3 was used as a base for creating a topology used for the purposes of this article. According to the design, the topology was created in an actual laboratory environment, where routers R1-R6 and the External network consisted of real Cisco routers. The devices specific to the LISP protocol (Mapping server, xTR1, xTR2, xTR3, PxTR) were implemented as CSR1000v virtual routers.

5.2 Implementation of the network controller

The controller was implemented in the form of a desktop application using the C# programming language. WPF technology was chosen for creating the user interface. The decision to implement a custom controller rather than adopting an existing SDN controller was deliberate and motivated by several concrete factors. Existing general-purpose SDN controllers such as OpenDayLight, ONOS, Ryu, and Floodlight [8] were evaluated and found to be unsuitable for this use case for the following reasons: some are excessively large and complex for the scope of this research, introducing unnecessary overhead; others are available only as enterprise editions and are, therefore, not freely accessible for academic research; and most critically, none of the evaluated controllers natively support the modern RESTCONF/YANG-based configuration management approach that is central to this research. By implementing a lightweight custom controller, it was possible to build a tool that contains precisely the functionality required: LISP configuration management, SNMP monitoring, ICMP testing, ML-based classification, and OpenFlow flow management, without any excess overhead. Equally important, the custom implementation serves as a proof-of-concept for the centralised management approach that may be intended to build upon in future research, and its modular architecture allows it to be extended as the research evolves.

6 Evaluation of LISP protocol experiments

Based on the results of the experiments, it has been demonstrated that the LISP routing protocol is more efficient than traditional routing protocols in terms of memory consumption, bandwidth utilization, and time required to achieve convergence. The benefits of programmatic configuration management using the controller were also highlighted. The obtained results are described in greater detail in the following subsections.

The experimental evaluation was designed to capture protocol behaviour across varying network conditions rather than a single fixed configuration. Two independent parameters were systematically varied: the number of advertised

networks (prefixes), evaluated across the range of 1 to 8 networks, and the number of inter-router links in the topology, evaluated across the range of 10 to 20 links. This two-dimensional variation allows the results to reflect the scaling behaviour characteristic of each protocol and enables the derivation of generalised closed-form formulas that describe how bandwidth consumption and memory requirements evolve as the network grows in either dimension. Bandwidth consumption and memory utilisation values were not obtained by direct runtime sampling, but were derived analytically using closed-form formulas constructed from the known structure of each protocol's control plane messages, as detailed in Section 6.2. The formulas capture how message sizes and routing table entries scale with the number of advertised networks and the number of links, respectively. Among the key findings that emerge from this analysis, discussed in full in Section 6.2, is that LISP bandwidth consumption is independent of the number of links in the topology. Since, LISP control plane communication is strictly localised between the xTR devices and the Map Server regardless of the size of the broader network. All analytically derived values were verified against packet captures obtained with Wireshark during live experiments, confirming that the message sizes and routing table entry counts predicted by the formulas matched those observed in the .pcap files. Because these metrics are fully determined by the protocol specifications and the topology configuration, they are deterministic and reproducible. Convergence time, by contrast, is a dynamic metric subject to real-time protocol behaviour, timer interactions, and hardware processing variability. Convergence times were, therefore, measured directly from the timestamps of packets recorded in .pcap captures during live experimental traffic, with a minimum of ten independent measurement runs per protocol and topology configuration. The arithmetic mean and standard deviation were computed from these observations. The observed standard deviations were consistently small relative to the mean values, confirming that the convergence behaviour of each protocol was stable and repeatable in the laboratory environment, in close agreement with the behaviour predicted by the respective protocol specifications.

6.1 Comparison of memory resource consumption

For all remote networks, dynamic routing protocols not only store the best routes in the routing table but also maintain information about all backup routes within their operational databases. The initial part of the experiments focused on these memory requirements. Based on theoretical knowledge, several formulas were estimated to calculate the number of records in relation to remote networks and possible paths.

Let's define the following three variables:

- *N_Networks:* The number of LISP networks represented as EID prefixes.
- *N_Routers:* The total number of routers making up the physical representation of the network.

- *N_Interfaces:* The number of interfaces of individual routers.

The total number of remote networks on one router can be represented by a formula:

$$N\_Networks\_Routers = N\_Networks \tag{1}$$

By generalizing the given statement for the entire network, it is possible to formulate the total number of networks learned by the routers in the entire topology by the following relation:

$$N\_Networks\_Topology = \sum_{i=0}^{n} N\_Networks\_Router_i \tag{2}$$

Traditional routing protocols keep all backup paths in addition to the best paths. Since the router should learn about each network from all connected neighbours through each interface, the total number of routes stored on each router is directly proportional to the number of interfaces, which can be formulated as:

$$N\_Paths\_Router = \sum_{i=0}^{n} N\_Interfaces_i \times N\_Networks \tag{3}$$

By generalizing this relationship for all routes, it is possible to calculate the total number of routes stored by the entire network infrastructure, which can be expressed by the formula:

$$N\_Paths\_Topology = N\_Routers \times \sum_{i=0}^{n} N\_Paths\_Router_i \tag{4}$$

Our topology contains four remote LISP networks and six routers R1-R6, serving purely the purpose of ensuring the reachability of end networks. To relax the problem, we consider only the links between these six routers. If one were to work with current end LISP routers, it would be necessary to consider that the router does not need to learn the directly connected network that it announces, which would bring inaccuracies into the calculations. At the same time, links to these end routers are also neglected. Table 2 presents the resource consumption related to the number of paths and networks, for individual routers and the entire topology.

Based on Table 2, even with a small topology containing 6 routers and 4 remote networks, the total number of stored networks in the sum for all routers is 24, and the total number of paths in the entire topology is 64. The advantage of the LISP protocol is that it does not work on R1-R6 devices and, nevertheless, guarantees the

delivery of data between LISP networks, which represents a significant saving in memory requirements.

Table 2
Memory resource consumption

Device name	Network numbers	Paths numbers
R1	4	8
R2	4	12
R3	4	20
R4, R5, R6	4	8
Topology	24	64

6.2 Bandwidth consumption comparison

Experiments have confirmed that traditional routing protocols exploit the transmission media to a greater extent than LISP. It was also shown that the data update process that occurs when there is a change in the network infrastructure in the case of OSPF and IS-IS protocols is dependent on the number of networks connected to the router that observed the change. The size of one update message of the OSPF protocol can be expressed by the formula:

$$1\_OSPF\_MSG = 72B + (12B \times N\_Networks) \tag{5}$$

The size of one update message of the IS-IS protocol can be expressed by a formula:

$$1\_IS - IS\_MSG = 67B + (12B \times N\_Networks) \tag{6}$$

The formulas listed contain the number of header check bytes without the L2 header, plus 12 bytes for each network included in the update. Since these are Link-State protocols, the update message must be delivered to each router at least once, with the fact that each message must be confirmed at the same time, which for the OSPF protocol represents the transfer of another fixed 64 bytes. By generalizing these assumptions, it is possible to estimate the following relations for calculating the bandwidth consumption of OSPF and IS-IS protocols:

$$BW\_OSPF = N\_Links \times (1\_OSPF\_MSG + 64B) \tag{7}$$

$$BW_{IS} - IS = N_{Links} \times 1\_IS - IS\_MSG \tag{8}$$

In the case of the EIGRP protocol, the update messages contain only a record of the network whose state has been changed, which represents savings compared to Link-

State protocols. Due to the above, the calculation of the bandwidth consumption is given by the sum of the size of the EIGRP packet (84B) and the confirmation message (40B), which is a total of 124B. The neighbouring device responds with the same sequence of data, and thus the total bandwidth consumption for the given line is 248B.

The LISP protocol achieved the most optimal results in this regard as well. To update the remote network, it requires the transmission of three messages between three devices, namely the Mapping Server, xTR1, and xTR2. No other device needs to be aware of the change. The first message is a LISP network registration request to the Mapping Server sent by the xTR2 device. The size of the message depends on the number of registered networks, and its size is given by the sum of 88B (LISP header) and 16B (one network specification). By generalization, the following formula can be derived:

$$Map\_Register = 88B + (16B \times N\_Networks) \tag{9}$$

When communicating to obtain a mapping record, a request of 120B must be sent to the mapping server, which in return responds with a 68B message.

The results of observations of the aspect of bandwidth consumption in single network announcement situations for the demonstration topology consisting of 14 lines from device xTR1 to device xTR2 are shown in the following Table 3:

Table 3
Bandwidth consumption

Protocol	1 msg size [B]	Total BW [B]
OSPF	84	2072
IS-IS	79	1106
EIGRP	84	3472
LISP	292	1528
LISP ProxyMS	292	1168

A clearer illustration of the increase in bandwidth consumption for each protocol can be observed when the number of networks being reported increases. Figure 4 presents the graph depicting the relationship between bandwidth consumption and the number of announced networks for all the aforementioned routing protocols. The increase in bandwidth consumption for all protocols relative to the number of announced networks is linear, while for the EIGRP protocol, there is no increase, since it always has a constant value. Additionally, it can be observed that the OSPF and IS-IS protocols have a steeper rate of increase compared to the LISP protocol. This further confirms that the LISP protocol optimizes bandwidth consumption, particularly in infrastructures consisting of numerous networks, which represents the current state of network infrastructures.

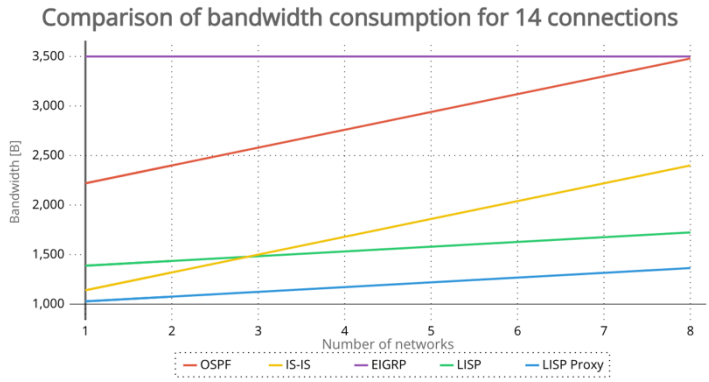

Figure 4

Bandwidth consumption for a variable number of networks

Another observation is the dependence of bandwidth consumption on the number of links present in the network topology. For simplicity, the number of advertised networks is maintained at 22 in this experiment. The variable component here is the number of links, which range from 10 to 20. Notably, the addition of links will not change the optimal path between xTR1 devices, MS, and xTR2, between which communications and primary routing information transfer takes place. Figure 7 illustrates the graph of the dependence of bandwidth consumption on the number of links between routers in the topology. The graph illustrates the fact that the bandwidth consumption increases in the case of traditional routing protocols linearly with the increasing number of links present in the network environment. Bandwidth consumption in the case of the LISP protocol is constant and does not directly depend on the number of connections in the network infrastructure.

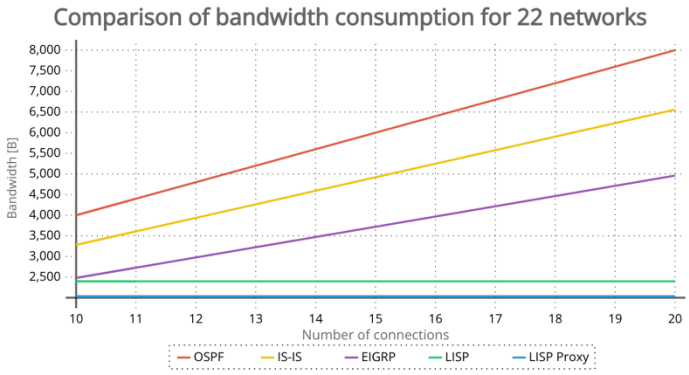

Figure 5

Bandwidth consumption for a variable number of connections

6.3 Comparison of the time interval for reaching convergence

Determining the time required to reach the state of convergence again, when all routers in the network have updated routing information, required the identification of the currently used path and its interruption during ongoing communication. The time between sending the last successful response request before the outage and the first successful request after the outage represented the time required to achieve convergence. The results obtained in the GNS3 topology are shown in Table 4.

Table 4
Time required to achieve convergence

Protocol	Time interval [sec]	Number of undelivered messages
OSPF	10.011	5
IS-IS	9.991	5
EIGRP	21.986	11
LISP	2.167	2

As can be observed, the LISP protocol achieved the state of convergence the fastest, in approximately 2 seconds, and also resulted in the smallest number of undelivered messages. This further validates the effectiveness of the LISP protocol.

6.4 Evaluation of the benefits of managing the network environment through the network controller

The implementation of the controller demonstrated the possibility of managing the configuration and monitoring the network infrastructure through a programmatic approach. This method allows for making configuration changes across multiple devices simultaneously, providing a scalable solution for networks of different sizes. Configuration in this way is also less prone to errors. Additionally, it enables universal management of device configurations from different manufacturers without requiring in-depth knowledge of each manufacturer's syntax.

7 Scalability and Theoretical Extrapolation

While the experimental results were obtained in a controlled laboratory environment with a finite number of nodes, the performance trends observed for LISP, OSPF, and EIGRP allow for theoretical extrapolation to larger-scale topologies. The following analysis projects the implications of the measured results

across three key dimensions: control plane bandwidth, memory consumption, and computational load with convergence time.

7.1 Control plane bandwidth scaling

In traditional IGPs (OSPF/IS-IS), bandwidth consumption for routing updates follows an exponential or high-linear growth pattern ($O(N^2)$ or O(NlogN) in dense mesh topologies) due to the flooding of Link State Advertisements (LSAs) across all nodes. In contrast, LISP's "pull-based" architecture scales linearly (O(N)) relative to the number of active flows because EID-to-RLOC mappings are resolved on demand and only the xTR devices directly involved in communication are updated. In a large-scale topology with thousands of endpoints, every OSPF router must process every topology change in the entire network, whereas a LISP xTR only processes mappings for the specific destinations it is communicating with. This suggests that the bandwidth savings observed in our experiments would become significantly more pronounced as the node count increases.

7.2 Memory consumption (RIB vs. Map-Cache)

The findings regarding memory efficiency are particularly significant in the context of scalability. In a large-scale deployment, a traditional router's Routing Information Base (RIB) must hold the complete set of advertised network prefixes, and memory consumption grows proportionally to the number of routes in the domain regardless of actual traffic patterns. LISP's Map-Cache, by contrast, is a demand-driven, transient structure that stores only mappings for destinations that have been actively queried. In scenarios characterised by sparse communication patterns, such as a branch office that communicates exclusively with a small number of data centre prefixes out of a globally large address space, LISP would maintain a memory footprint estimated to be 80–90% smaller than a traditional router required to hold the full routing table.

7.3 Computational load and convergence

In larger topologies, OSPF's SPF calculation time increases significantly with the complexity of the Link State Database (LSDB). LISP avoids this global recalculation entirely: only the relevant xTR sends a Map-Register message to the Map Server, and the rest of the network infrastructure is unaffected. Extrapolating the convergence data obtained in our experiments suggest that in multi-area or autonomous-system-scale deployments, LISP would provide significantly faster endpoint re-homing and route update propagation compared to the incremental LSA flooding and multi-area SPF recalculation required by OSPF. This advantage is expected to be especially pronounced during periods of high network dynamism,

such as mass endpoint mobility events or simultaneous link failures in multiple areas, where traditional IGP convergence times accumulate across the domain while LISP convergence remains bounded by a constant number of Map-Register and Map-Reply message exchanges.

Conclusions

The focus of this research was on comparing traditional routing protocols with the LISP protocol. The experiments demonstrated that the LISP protocol is more efficient in terms of memory resource consumption, bandwidth utilization, and reaching a state of convergence. Consequently, it can be argued that the LISP protocol is better suited for deployment in larger, dynamically changing networks. Research has also highlighted that a programmatic approach to network configuration management and monitoring offers several benefits, including faster change implementation, reduced error rates, and enhanced versatility. From a scientific standpoint, this work contributes to the body of knowledge by providing systematic experimental evidence of LISP efficiency specifically within an IGP context, compared against OSPF, EIGRP, and IS-IS under controlled conditions. The confirmed advantages of the overlay and centralised-mapping approach suggest that the theoretical scalability benefits of LISP, derived from the separation of location and identity and the elimination of network-wide flooding, translate into measurable gains even in relatively small topologies of the type studied here. These gains are expected to grow superlinearly as network size increases because traditional IGP update complexity scales with the number of routers and links ($O(n^2)$ flooding in dense topologies), while LISP control-plane messages remain localised to the xTR-to-MapServer pair regardless of network size, the relative advantage of LISP is theoretically amplified in larger networks. While verifying this scaling behaviour at a larger scale (hundreds of routers) would require dedicated simulation infrastructure beyond the scope of this study, the mathematical models derived in Section 6 for update message sizes and routing table entries provide a theoretical basis for extrapolating the observed trends to larger topologies. Simulation-based validation at scale, as well as deployment in virtualised environments representative of modern data centres and cloud platforms, is identified as a primary direction for future research.

Future work regarding this topic of network management and routing protocols could dive into exploring the integration of machine learning techniques to further enhance the efficiency and adaptability of network infrastructures. By leveraging ML algorithms, future systems could analyse traffic patterns, predict potential network congestion, and dynamically optimize routing decisions based on real-time conditions. This integration could facilitate a more intelligent network and improve fault tolerance by proactively identifying issues before they impact network performance. As mentioned in the introduction, when networks continue to grow in complexity and scale, incorporating machine learning could lead to more intelligent solutions in network management.

References

[1] Katonová, E. A.; Petija, R.; Jakab, F.; Kainz, O.; Michalko, M.; Dzubak, J.: An innovative multidisciplinary approach to the teaching computer networks and cybersecurity. In: 2022 20th International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE, 2022. DOI: 10.1109/iceta57911.2022.9974880.

[2] Katonová, E. A.; Petija, R.; Jakab, F.; Feciľak, P.; Michalko, M.: Integration of the programmability and automation concepts into the teaching of computer networks. In: 2021 19th International Conference on Emerging eLearning Technologies and Applications (ICETA). IEEE, 2021. DOI: 10.1109/iceta54173.2021.9726621.

[3] Correa Chica, J. C.; Imbachi, J. C.; Botero Vega, J. F.: Security in SDN: A comprehensive survey, In: Journal of Network and Computer Applications, 2020. DOI: 10.1016/j.jnca.2020.102595.

[4] Kamarudin, I. E.; Ameedeen, M. A.; Mohd, F.; Zabidi, A.: Integrating Edge Computing and Software Defined Networking in Internet of Things: A Systematic Review. Iraqi Journal for Computer Science and Mathematics College of Education - Aliraqia University, 2023. DOI: 10.52866/ijcsm.2023.04.04.011.

[5] Thirupathi, V.; Sandeep, CH.; Kumar, N.; Kumar, P. P.: A comprehensive review on sdn architecture, applications and major benifits of SDN. In: International Journal of Advanced Science and Technology, 2019.

[6] Paliwal, M.; Shrimankar, D.; Tembhurne, O.: Controllers in SDN: A Review Report. In: IEEE Access, 2018, DOI: 10.1109/access.2018.2846236.

[7] Nisar, K.; Jimson, E. R.; Hijazi, M. H. A.; Welch, I.; Hassan, R.; Aman, A. H. M.; Sodhro, A. H.; Pirbhulal, S.; Khan, S.: A survey on the architecture, application, and security of software defined networking: Challenges and open issues. In: Internet of Things, 2020. DOI: 10.1016/j.iot.2020.100289.

[8] Mamushiane, L.; Lysko, A.; Dlamini, S.: A comparative evaluation of the performance of popular SDN controllers. In: 2018 Wireless 41 Literature Days (WD). IEEE, 2018, s. 54–59. DOI: 10.1109/wd.2018.8361694.

[9] Vemula, S.; Gooley, J.; Hasan, R.: Cisco Software-Defined Access. Pearson Education (US), Limited, 2020. ISBN: 9780136448389.

[10] Paillisse, J.; Portoles, M.; Lopez, A.; Rodriguez-Natal, A.; Iacobacci, D.; Leong, J.; Moreno, V.; Cabellos, A.; Maino, F.; Hooda, S.: SD-access: practical experiences in designing and deploying software defined enterprise networks. In: Proceedings of the 16th International Conference on emerging

Networking EXperiments and Technologies (CoNEXT), 2020. DOI: 10.1145/3386367.3431288.

[11] Farinacci, D.; Maino, F.; Fuller, V.; Cabellos-Aparicio, A.: Locator/ID Separation Protocol (LISP) Control Plane [RFC 9301]. RFC Editor, 2022. Request for Comments, No. 9301. DOI: 10.17487/RFC9301.

[12] Cabellos-Aparicio, A.; Saucez, D.: An Architectural Introduction to the Locator/ID Separation Protocol (LISP) [RFC 9299]. RFC EDITOR, 2022. Request for Comments, No. 9299. DOI: 10.17487/RFC9299.

[13] Manzoor, A,; Hussain, M.; Mehrban, S.: Performance Analysis and Route Optimization: Redistribution between EIGRP, OSPF & BGP Routing Protocols. In: Computer Standards & Interfaces, 2020. DOI: 10.1016/j.csi.2019.103391.

[14] Haji, S. H.; Zeebaree, S. R. M.; Saeed, R. H.; Ameen, S. Y.; Shukur, H. M.; Omar, N. Sadeeq, M. A. M.; Ageed, Z. S.; Ibrahim, I. M.; Yasin, H. M.: Comparison of Software Defined Networking with Traditional Networking. In: Asian Journal of Research in Computer Science, 2021. DOI: 10.9734/ajrcos/2021/v9i230216

[15] McParland, T.; Niraula, M.: Distributed Mobility Anchoring Using LISP Mobile Node, In: 2020 Integrated Communications Navigation and Surveillance Conference (ICNS). IEEE, 2020. DOI: 10.1109/ICNS50378.2020.9222937.

[16] Luong, D. K.; Hu, Y.-F.; Li, J.-P.; Ali, M.; Benamrane, F.; Abdo, K.: Seamless handover in SDN-Based Future Avionics Networks with Network Coding and LISP Mobility Protocol, In: AIAA 38th Digital Avionics Systems Conference (DASC). IEEE, 2019. DOI: 10.1109/DASC43569.2019.9081615

[17] Lu, T.-T.: Novel method for transferring access control list rules to synchronize security protection in a locator/identifier separation protocol environment with cross-segment host mobility. In: International Journal of Communication Systems, 2018. DOI: 10.1002/dac.3868

[18] Sun, K.; Kim, Y.: Enhanced LISP Mapping System for Optimizing Service Path in Edge Computing Environment. In: IEEE Access, 2020. DOI: 10.1109/access.2020.3031915

[19] Fafolahan, E. M. O.; Pierre, S.: A Seamless Mobility Management Protocol in 5G Locator Identificator Split Dense Small Cells. In: IEEE Transactions on Mobile Computing. IEEE, 2020. DOI: 10.1109/TMC.2019.2915071

[20] Babakian, A.; Monclus, P.; Braun, R.; Lipman, J.: A Retrospective on Workload Identifiers: From Data Center to Cloud-Native Networks. In: IEEE Access, 2022. DOI: 10.1109/ACCESS.2022.3211293.

[21] Shakil, T.; Jain, V.; Louis, Y.: LISP Network Deployment and Troubleshooting: The Complete Guide to LISP Implementation on IOS-XE, IOS-XR, and NX-OS, In: Cisco Press, 2019. ISBN: 9780134783130