algorithms

MDPI

Article

Anchor-Based Method for Inter-Domain Mobility Management in Software-Defined Networking

Akichy Adon Jean Rodrigue Kanda [1], Amanvon Ferdinand Atta [2,*], Zacrada Françoise Odile Trey [3], Michel Babri [1] and Ahmed Dooguy Kora [4]

1 Laboratoire de Recherche en Informatique et Telecommunication, Institut National Polytechnique Felix Houphouët-Boigny, Yamoussoukro BP 1093, Côte d'Ivoire; akichy.kanda18@inphb.ci (A.A.J.R.K.); michel.babri@inphb.ci (M.B.)
2 Unité de Recherche et d'Expertise Numerique, Universite Virtuelle de Côte d'Ivoire, Abidjan 28 BP 536, Côte d'Ivoire
3 Laboratoire de Mécanique et Informatique, Université Félix Houphouët Boigny, Abidjan 01 BP V34, Côte d'Ivoire; trey.odile@ufhb.edu.ci
4 Ecole Superieure Polytechnique, Université Cheikh Anta Diop, Dakar BP 5085, Senegal; ahmed.kora@esmt.sn
* Correspondence: amanvon.atta@uvci.edu.ci

Abstract: Recently, there has been an explosive growth in wireless devices capable of connecting to the Internet and utilizing various services anytime, anywhere, often while on the move. In the realm of the Internet, such devices are called mobile nodes. When these devices are in motion or traverse different domains while communicating, effective mobility management becomes essential to ensure the continuity of their services. Software-defined networking (SDN), a new paradigm in networking, offers numerous possibilities for addressing the challenges of mobility management. By decoupling the control and data planes, SDN enables greater flexibility and adaptability, making them a powerful framework for solving mobility-related issues. However, communication can still be momentarily disrupted due to frequent changes in IP addresses, a drop in radio signals, or configuration issues associated with gateways. Therefore, this paper introduces Routage Inter-domains in SDN (RI-SDN), a novel anchor-based routing method designed for inter-domain mobility in SDN architectures. The method identifies a suitable anchor domain, a critical intermediary domain that contributes to reducing delays during data transfer because it is the closest domain (i.e., node) to the destination. Once the anchor domain is identified, the best routing path is determined as the route with the smallest metric, incorporating elements such as bandwidth, flow operations, and the number of domain hops. Simulation results demonstrate significant improvements in data transfer delay and handover latency compared to existing methods. By leveraging SDN's potential, RI-SDN presents a robust and innovative solution for real-world scenarios requiring reliable mobility management.

Keywords: IP mobile; software-defined networking (SDN); mobility management; handover; Proxy Mobile IPv6 (PMIPv6); mobility anchor

check for updates

Citation: Kanda, A.A.J.R.; Atta, A.F.; Trey, Z.F.O.; Babri, M.; Kora, A.D. Anchor-Based Method for Inter-Domain Mobility Management in Software-Defined Networking. *Algorithms* **2024**, *17*, 566. https://doi.org/10.3390/a17120566

Received: 23 October 2024
Revised: 4 December 2024
Accepted: 9 December 2024
Published: 11 December 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Today's Internet is predominantly driven by wireless technology and its extensions. When coupled with the Internet, wireless technology provides access to a myriad of services available at all times, even while on the move [1]. However, this extensive use poses a problem when considering the design of the Internet. The architecture of the Internet is primarily based on logical connections between a clearly identified source, which includes its IP address and other parameters, and one or more identified recipients. Hence, when even one of these parameters is modified during movement, the connection is reset, potentially causing intolerable delays for certain real-time service applications. Ensuring service continuity during the movement of a mobile node is a genuine challenge as some applications and services are less tolerant of disconnection delays. Hence, the problem addressed in our study can be stated as follows: Given a multi-domain network, and a mobile node in the network, how can we find a path to reduce data transfer delay when the mobile node moves between two different domains. The challenge of managing mobility in multi-domain networks, especially in ensuring uninterrupted IP traffic flow, has been a persistent challenge that has been tackled through different methods throughout the years. These solutions, such as Mobile IP, Proxy Mobile IP, and Distributed Mobility Management, have unique benefits and drawbacks. Lately, software-defined networking (SDN) has become a promising solution for managing networks effectively. In fact, SDN is a network architecture that aims to be dynamic, manageable, economical, and flexible in order to be appropriate for the high-bandwidth, dynamic applications of today [2]. Its fundamental idea of dividing the control plane from the data plane has made a revolution in networks possible in a number of ways [3–5]. SDN controllers make up the control plane, whereas switches, or forwarding elements, make up the data plane. A number of benefits are provided by SDN, including bespoke application-driven network management, flexible programmable control, and dynamic routing [6,7]. By integrating SDN with mobility management, an improvement of route selection is achieved when a mobile node (MN) moves within an SDN domain [8]. In the context of SDN, a domain typically consists of network devices (e.g., switches, routers) managed by a single SDN controller [9]. Moreover, a domain refers to a logically or physically bounded part of a network that operates under a specific administrative control. The role of SDN in a multi-domain architecture is crucial, as it enables centralized control over heterogeneous networks while ensuring interoperability between different domains. This capability is particularly relevant in scenarios involving mobility management, where seamless communication across domains is essential for reducing latency and ensuring service continuity.

To address this issue, we propose a routing method based on SDN and the mobility anchor principle for inter-domain mobility management. The domain used as an anchor is a domain responsible for retrieving data from a mobile node on the move and creating a tunnel between it and its correspondent in order to facilitate communication and reduce the data transfer delay. To sum up, our main contributions are threefold:

1. We model the problem of inter-domain mobility management in SDN-based architectures as a problem of selecting an efficient route belonging to a set of routes. The chosen route reduces a metric that incorporates bandwidth availability, the number of hops between the mobile node and the anchor domain, and the number of hops (i.e., domains) between the anchor domain and data destination node or the corresponding node.
2. We propose an algorithm to select an anchor domain that is closest to the data destination node. This strategic placement contributes to reducing the overall data transfer delay.
3. We develop an algorithm specifically designed for route selection within the SDN framework. This algorithm prioritizes routes that traverse the chosen anchor domain while considering parameters like bandwidth capacity and the number of domains involved in the data transfer path.

The remainder of this paper is organized as follows: Section 2 summarizes the issues with existing mobility management methods. Section 3 gives a formal description of the addressed problem. Section 4 presents our method in more detail. Section 5 explains the simulation setup and the results, and Section 6 gives the conclusions of our study.

2. Related Work

Several authors have explored SDN-based approaches for mobility management. Idri et al. [10] utilizes routing headers to sustain communication without IP tunneling mechanisms. However, their work does not delve into route selection, leaving a significant gap that requires a central controller to manage the network, which incurs additional costs. Wang et al. [11] proposed FastSplit. This mechanism reuses previous routes to reduce signaling load, whereas new routes remain nearly optimal. FastSplit uses path length and signaling overhead to determine new routes but does not consider real-time information. In [8], the authors introduced mobility management based on Mobile IP, capable of handling inter-domain transfers. However, this approach relies on Mobile IPv6 and its route optimization function, which is not universally supported by all MNs.

In [12], an OpenFlow-compatible Mobile IPv6 proxy (OF-PMIPv6) is proposed, which separates the control path from the data path. Specifically, the mobility control function resides in the controller while data paths between the Mobile Access Gateway (MAG) and Local Mobility Anchor (LMA) are maintained as an IP data tunnel. The MAG and the LMA are key components of the PMIPv6 (Proxy Mobile IPv6) protocol, designed to manage node mobility in IP networks. The MAG, located at the network edge, detects mobile node movements and establishes tunnels to the LMA, which is responsible for mobility anchoring and centralized traffic routing. The LMA maintains a database of active sessions and ensures communication continuity during handovers when nodes change their point of access. This collaboration ensures that nodes retain the same IP address while moving, providing seamless connectivity. However, this architecture can be limited by increased latency and potential failures at the LMA level. In [13], an S-DMM (Software-Defined Mobility Management) scheme was introduced based on the SDN architecture. S-DMM removes mobility logic from access routers, providing flow-based mobility support. S-DMM achieves a transfer performance similar to existing DMM but simplifies mobility management. In [14], the authors propose an OpenFlow-based Mobile IPv6 proxy (OPMIPv6) utilizing the OpenFlow architecture advantages. In OPMIPv6, LMA functions reside in an SDN controller, whereas the MAG function is implemented in an access switch or the controller. It alleviates IP-in-IP tunneling overhead by utilizing OpenFlow flow-table-based routing. In [15], an SDN-based DMM solution (SDN-DMM) was proposed. In SDN-DMM, mobility control was implemented as an application within the controller, leveraging the benefits of the SDN architecture. Lastly, Ref. [16] introduces a distributed IP mobility management scheme by adopting SDN technology. This scheme uses a flat mobile network architecture that eliminates a single point of failure. A flat mobile network architecture is a network design where no centralized hierarchy exists. In this type of network, entities responsible for management functions (such as access points, routers, or controllers) communicate directly with each other without relying on a single central point of control or routing. Furthermore, it utilizes multiple controllers to synchronize distributed mobility management mechanisms and alleviate the load on a centralized controller. The aforementioned SDN solutions can facilitate mobility management by leveraging SDN advantages. In [17], a PMIPv6-based solution, SDN introduces additional control signaling into OF-PMIPv6. The reactive and proactive transfer schemes of this solution enable improvements in transfer latency and packet loss compared to standard PMIPv6. In [18], a NO-PMIPv6 method was proposed, reducing the mobile support delay during inter-domain movement. This method draws inspiration from the strategic positioning of edge nodes and the PMIPv6 protocol to anticipate the mobile's next destination. However, this method is limited to reducing inter-domain transfer delay and does not address optimal data routing after a domain change.

To address the aforementioned issues, we propose a method (RI-SDN) that reduces inter-domain transfer delay by optimizing routes after inter-domain movement. This method is inspired by the SDN principle. First, the proposed method selects an anchor domain (i.e., a domain that determines the position of the mobile node and data flows between it and its correspondent) such that the anchor is closest to the correspondent node. Then, we determine paths based on the anchor, and parameters such as bandwith and the number of traversed domains.

3. Problem Formulation

Given a multi-domain network and a mobile node in the network, how can 110 paths be found to reduce data transfer delay when this mobile node moves between two different domains? Let the following be the given parameters of the problem:

- $G = (V, E)$: Multi-domain network graph representing nodes and links, where V is the set of nodes in the network and E is the set of links (edges) connecting the nodes.
- $MN \in V$: A mobile node.
- D: Set of domains within the network.
- D_0 and D_f: Respectively, the initial domain where MN is located before its motion and the final domain where MN is located after its motion.
- $R = \{r_1, \dots, |R|\}$: Set of available routes for the mobile node.
- B_e: The value of the bandwidth on link $e \in E$.
- T_e: The delay on link e.

Our objective is to identify a route $r_k \in R$ with minimal data transfer delay cost M_k. According to [19], Equation (1) provides the expression for M_k:

$$M_k = MF_k + MD_k + MB_k \tag{1}$$

And

$$T_k = \sum_{e \in E_k} T_e \tag{2}$$

With
M_k, the total metric of route r_k;
MF_k, the total metric for flow operations;
MD_k , the total metric for the number of domain hops;
MB_k, the total metric for the available bandwidth of the route;
T_k, the total metric for delay of the route;
E_k is the set of links that constituted the route r_k.

$$MD_k = \frac{D_k}{D_{max}}, k \in \{1, 2, \dots, |R|\} \tag{3}$$

$$MF_k = \frac{F_k}{F_{max}}, k \in \{1, 2, \dots, |R|\} \tag{4}$$

$$MB_k = \frac{B_k}{B_{max}}, k \in \{1, 2, \dots, |R|\} \tag{5}$$

D_k is the number of domains that route r_k goes through;
F_k is the number of flow operations for switching route r_k;
B_k is the minimum bandwidth of route r_k;
D_{max} is the number of domains in the network;
B_{max} is the largest bandwidth in the network;
F_{max} is equal to $2N$, with N the number of domains in the network.

4. Proposed Methodology

4.1. Our Consideration on Cost of Data Transfer Delay

In our study, we assume that the routes are not susceptible to failures or issues that could result in the cancellation of the data flow. So, the value of MF_k is zero and the model (see Equation (1)) can be modified as presented by Equation (6):

$$M_k = MD_k + MB_k \quad \forall k \in \{1, \dots, |R|\} \tag{6}$$

Furthermore, among the bandwidths of the links on route r_k, the value of the useful bandwidth (the smallest bandwidth on the chosen route) to allow the use of a service in our model is the minimum value on all the links of the chosen route Furthermore, we introduce a concept using the MIP proxy, which is the mobility anchor. Based on [20], this allows the improvement of node mobility management and the data transfer delay to be reduced. We summarize these principles into three points to outline this method.

- An anchor is a domain located on the data route between the mobile node (MN), which is also called the source, and its corresponding node (CN), which is also called the destination.
- An anchor is a domain located between the most connected domain and the one closest to the destination.
- An anchor must be unique.

In the context of our study, the variable MD_k is translated as the sum of the number of hops between the mobile node N and the anchor A, denoted as MD_{NA}, and between the anchor and the corresponding node C, denoted as MD_{AC}. Based on [18], for the closest anchor node to the destination, the route between the anchor and the destination becomes a mandatory path, so only one is needed, MD_{NA}, which is most likely to change. Therefore, we obtain

$$MD_k = MD_{NA}, \quad \forall k \in \{1, \ldots, |R|\} \tag{7}$$

So, the model can be summarized as follows:

$$M_k = MD_{NA} + B_k \quad \forall k \in \{1, \ldots, |R|\} \tag{8}$$

with

$$B_k = min\{B_e\}, \quad e \in E_k, \tag{9}$$

4.2. The RI-SDN Method

4.2.1. Algorithm Description

The algorithm of the RI-SDN method can be summarized in two stages: the first stage involves selecting the anchor domain, which is followed by the stage of suitable route selection.

The first stage, which aims to select the anchor domain, is summarized as illustrated in Figure 1. This stage consists of the following steps.

- **Step 1**: After initializing the network graph, the adjacency matrix is determined. The adjacency matrix is a table representing connections between different nodes of the network. For each pair of nodes, the matrix contains a value of 1 if a direct link exists between them, or 0 otherwise. This matrix is used to determine the set of most connected nodes, SMC. In the matrix, the row or column with the most 1s corresponds to the most connected node (that represent a domain).
- **Step 2**: If SMC is a singleton, then go to step 3. Otherwise, go to step 4.
- **Step 3**: If there does not exist a domain between the most connected domain and the destination domain, then the most connected domain is taken as the anchor domain. If there exists a single domain between the most connected domain and the destination domain, then this single domain is taken as the anchor domain. Otherwise, go to step 5.
- **Step 4**: For each domain $d \in SMC$, compute the distance between d and the destination domain. The domain d with the smallest distance to the destination domain is considered the most connected domain. Go back to step 3.
- **Step 5**: Determine the set of domains closest to the destination, SCD. If this set is a singleton, then the closest domain is taken as the anchor domain. Otherwise, a domain d belonging to SMC(SCD) is taken as the anchor domain. SMC(SCD) means the set of most connected domains among the domains belonging to SCD.

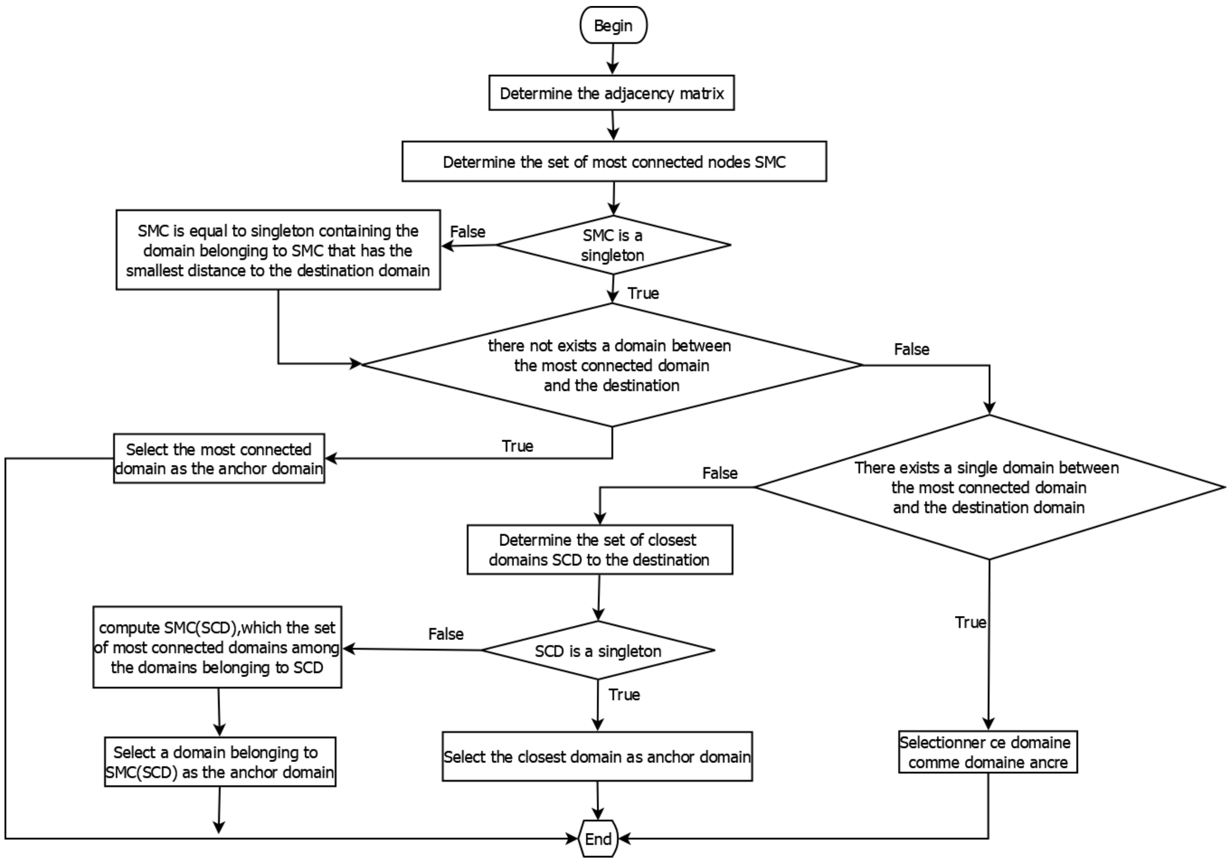

Begin
Determine the adjacency matrix
Determine the set of most connected nodes SMC
SMC is a singleton
False
True
SMC is equal to singleton containing the domain belonging to SMC that has the smallest distance to the destination domain
there not exists a domain between the most connected domain and the destination
False
True
Select the most connected domain as the anchor domain
There exists a single domain between the most connected domain and the destination domain
False
True
Determine the set of closest domains SCD to the destination
SCD is a singleton
False
True
compute SMC(SCD),which the set of most connected domains among the domains belonging to SCD
Select a domain belonging to SMC(SCD) as the anchor domain
Select the closest domain as anchor domain
Selectionner ce domaine comme domaine ancre
End

Figure 1. Algorithm for anchor domain selection.

The next stage of the RI-SDN method is based on the output of the first stage (i.e., the anchor domain selected) and aims to determine the data transfer path (also called the route) during inter-domain mobility. The next stage is summarized as shown in Figure 2. This stage consists in the following steps.

- **Step 1**: Determine RMA, the set of routes that pass through the mobility anchor MA.
- **Step 2**: If RMA is empty, then go to step 3. Otherwise, go to step 4.
- **Step 3**: Use the default route suggested by the controller for communication between the mobile node (MN) and its correspondent node (CN).
- **Step 4**: For each route $r_k \in RMA$, determine the smallest bandwidth B_k and compare to B_s (service's bandwidth). This is because the minimum bandwidth B_k of a route r_k is the smallest bandwidth of the links in the route r_k.
  If $B_k < B_s$, then add r_k to $RMA(Sup)$ and go to step 5, with $RMA(Sup)$ being the set of routes r_k that have a bandwidth link higher than B_s. Otherwise, add r_k $RMA(inf)$, with $RMA(inf)$ being the set of routes r_k that have a bandwidth link lower than B_s, and go back to step 3 if $RMA(Sup)$ is empty.
- **Step 5**: For each route $r_k \in RMA(Sup)$, compute M_k according to Equation (8).
- **Step 6**: Determine SRK, which is the set of routes having the smallest value of M_k. If SRK is a singleton, route r_k in SRK is selected as the suitable route. Otherwise go to step 7.
- **Step 7**: We take route r_k, where the sum of the bandwidth over each link is the smallest, with links having minimum bandwidths greater than the one required to ensure a quality-of-service service B_s, in order to avoid bandwidth wastage.

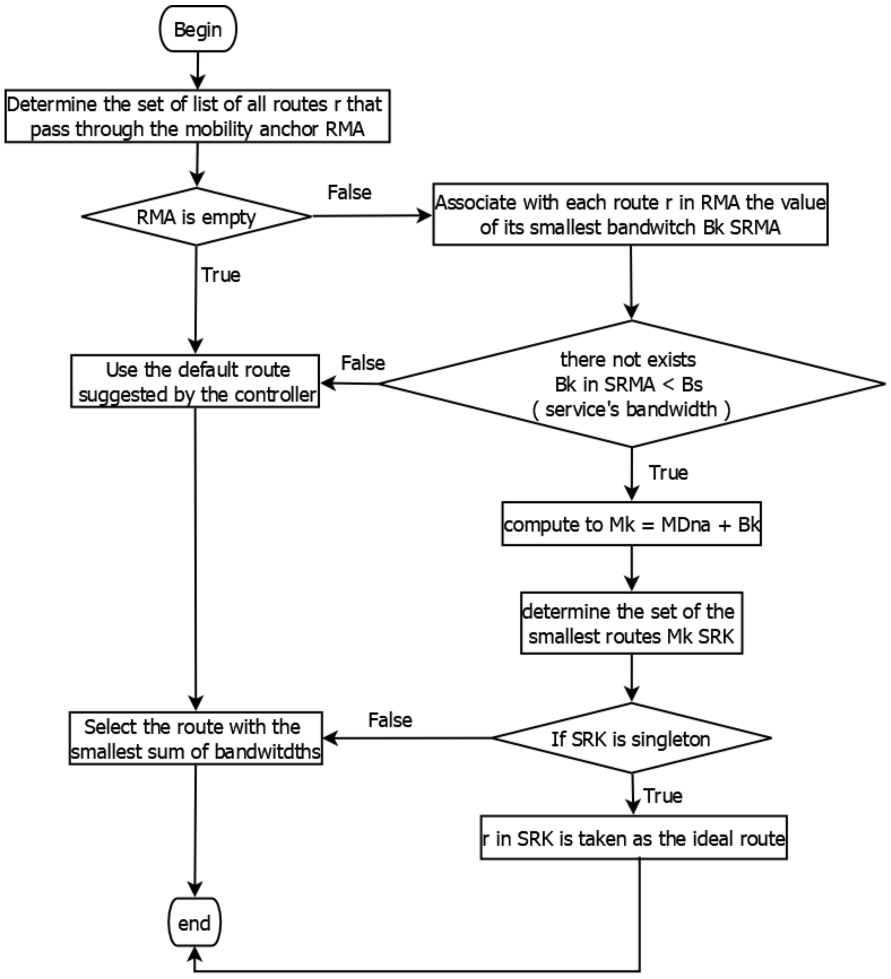

Begin
Determine the set of list of all routes r that pass through the mobility anchor RMA
RMA is empty
False
Associate with each route r in RMA the value of its smallest bandwitch Bk SRMA
True
Use the default route suggested by the controller
False
there not exists Bk in SRMA < Bs (service's bandwidth)
True
compute to Mk = MDna + Bk
determine the set of the smallest routes Mk SRK
Select the route with the smallest sum of bandwitdths
False
If SRK is singleton
True
r in SRK is taken as the ideal route
end

Figure 2. Algorithm for route selection.

4.2.2. Method Illustration

We present two figures, Figures 3 and 4, which serve as the basic architecture to illustrate our method. Figure 3 represents the network with eight (08) domains in which a corresponding node *CN* is fixed and a mobile node *MN* moves. When the mobile node moves from one domain to another and connects to it (each domain is represented by a node that is actually the domain controller), the controller of that domain then detects a new registration and initiates the handover procedure for that mobile node. Figure 4 is a simplified representation, showing the links between each domain with their weights (representing bandwidth values) and the domain numbers to locate the mobile's movement. The service requires a minimum bandwidth of 400 Mbps ($B_s = 4$). In the next two paragraphs, we illustrate the first stage and then the second stage of the RI-SDN method.

First stage: Mobility Anchor Selection

The adjacency matrix associated with our network is presented in Figure 5. Let us determine the mobility anchor.

- The red line has the most 1 s, the set of most connected nodes $SMC = \{5\}$. Thus, the domain associated with this line is the most connected and corresponds to node 5. The nodes between node 5 and the destination node are candidate nodes.
- SMC is a singleton. We check whether there are any nodes between this node and the destination node. On the adjacency matrix, nodes 6 and 7 are connected to both node 5 and node 8 (the destination node), as indicated by the black lines and the small green circles.

- Therefore, the set of domains closest to the destination, SCD, is computed. $SCD = \{6,7\}$. SCD is not a singleton, so we determine the set of most connected belonging to SCD, which is denoted by $SMC(SCD)$.
- $SMC(SCD) = \{6\}$. Thus, node 6 is taken as the anchor.

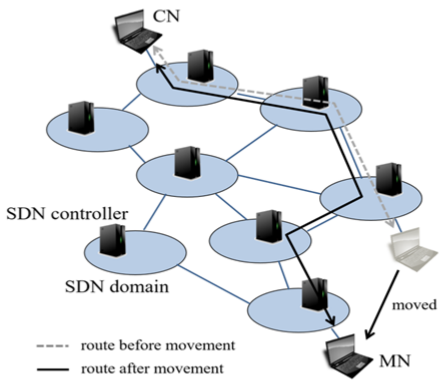


Figure 3. Basic architecture, from [19].

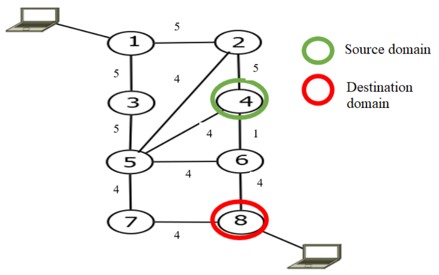
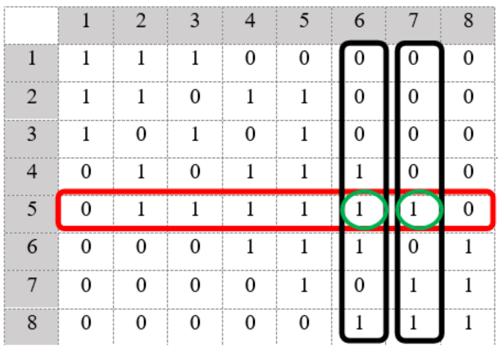


Figure 4. Basic simplified architecture.

Figure 6 shows the key concepts of our method (i.e., RI-SDN) and the route provided by our method from the architecture illustrated by Figure 4.



	1	2	3	4	5	6	7	8
1	1	1	1	0	0	0	0	0
2	1	1	0	1	1	0	0	0
3	1	0	1	0	1	0	0	0
4	0	1	0	1	1	1	0	0
5	0	1	1	1	1	1	1	0
6	0	0	0	1	1	1	0	1
7	0	0	0	0	1	0	1	1
8	0	0	0	0	0	1	1	1

Figure 5. Adjacency matrix.

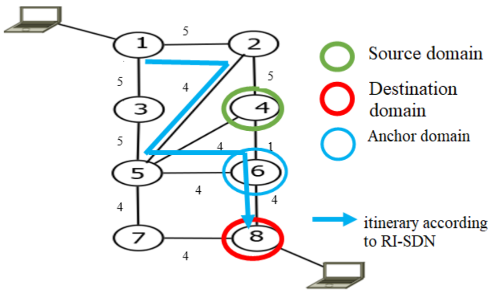


Figure 6. Architecture with routes.

Second stage:

The anchor domain is strategically positioned, and this strongly influences the model in [10], where a metric involves the number of domains crossed by the data flow.

- Determine RMA:
  $RMA = \{r_1 = \{1\text{-}3; 3\text{-}5; 5\text{-}6; 6\text{-}8\}; r_2 = \{1\text{-}2; 2\text{-}5; 5\text{-}6; 6\text{-}8\}; r_3 = \{1\text{-}2; 2\text{-}4; 4\text{-}6; 6\text{-}8\}; r_4 = \{1\text{-}2; 2\text{-}4; 4\text{-}5; 5\text{-}6; 6\text{-}8\}\}$
- RMA is not empty. So, for each route r_k of RMA, B_k is computed as follows:
  $B_1 = 4$, $B_2 = 4$, $B_3 = 1$, $B_4 = 4$
- Compare B_k and B_s:
  $B_1 \geq B_s$, $B_2 \geq 4$, $B_3 < B_s$, $B_4 \geq B_s$, then $RMA(Inf) = \{B_3\}$ and $RMA(Sup) = \{B_1, B_2, B_4\}$
- Calculation of M_k metrics:
  - For r_1 we have $M_1 = MD_1 + 1/B_1$
    $M_1 = 3 + 1/4 = 3,25$
  - For r_2 we have $M_2 = MD_2 + 1/B_2$
    $M_2 = 3 + 1/4 = 3,25$
  - For r_4 we have $M_4 = MD_4 + 1/B_4$
    $M_4 = 4 + 1/4 = 4,25$
- Determine SRK:
  $SRK = \{r_1; r_2\}$
- Sum of bandwidths for each $r_k \in SRK$:
  For r_1, this sum is equal to $5 + 5 + 4 + 4 = 18$;
  For r_2, this sum is equal to $5 + 4 + 4 + 4 = 17$;
  The result is that r_2 is the route that will be taken.

5. Performance Evaluation

We conducted a simulation to evaluate the performance of our RI-SDN (Inter-domain Software-Defined Networking) solution and the work in [19]. The following subsections present the performance criteria used, the simulation setup, and results analysis.

5.1. Performance Criteria

5.1.1. Data Transfer Delay

Data transfer delay is a key criterion in evaluating the performance of a routing algorithm, especially in networks where latency is critical (e.g., real-time networks). The transfer delay is calculated between the correspondent node (CN) and the mobile node (MN). In RI-SDN, the best path is the one with the smallest combined metric of the number of domains, bandwidth, and delay. Moreover, the RI-SDN method takes into account the delays on each link and the service requirement, whereas in [19], the focus is on a metric that considers the number of domains, bandwidth, and flow operations.

According to the RI-SDN method, the data transfer delay is given by the formula:

$$T_k = \frac{L}{B_k} + \sum_{e \in E_k}^{N_h} T_e \tag{10}$$

- L: Packet size to be transferred;
- B_k: Minimum bandwidth on the route r_k;
- N_h: The number of hops between the correspondent node (CN) and the mobile node (MN);
- T_e: The delay on link e.

5.1.2. Handover Latency

Handover latency is a critical factor in mobile networks where the *mobile node (MN)* moves from one network coverage area to another. This criterion measures the total time required to transfer the active connection of the MN from one access point (or domain) to another without interrupting communication.

Handover latency is defined as the delay between the moment the mobile node (MN) leaves the range of the old station or domain (access point) and the moment it is fully reconnected to the network through the new access point. This process involves several steps:

1. Movement detection: The network or the MN detects that the current connection is becoming weak or unstable.
2. Handover decision: A new access point or route is selected for the MN.
3. Data transfer: The context of the active session is transferred to the new access point.
4. Reconnection: The MN is connected to the new domain or access point, and the session is restored.

The total time for these steps constitutes the handover latency, and it is essential to minimize it to avoid a degradation of quality of service (QoS).

In the *RI-SDN* approach, handover latency is considered during route selection. The network performs real-time calculations to select routes that minimize the end-to-end latency, including during handovers. Equation (11) gives the expression of handover latency.

$$L_{handover} = t_{detection} + t_{decision} + t_{anchoring} + t_{transfer} + t_{reconnection} \tag{11}$$

- $L_{handover}$: Total handover latency.
- $t_{detection}$: Time to detect that the MN needs to change domain.
- $t_{decision}$: Time to select a new access point and route.
- $t_{anchoring}$: Time required to run the mobility anchor selection algorithm.
- $t_{transfer}$: Data transfer delay.
- $t_{reconnection}$: Time to reconnect the session.

5.2. Simulation Setup

For our experiments, we used a machine running a Linux operating system (Ubuntu 20.04 LTS 64-bit) with an Intel® Core™ i7-6600U CPU operating at a frequency of 2.60 GHz to 2.81 GHz and 20 GB of RAM. On the host system, we used Mininet-WiFi, which is an open source software, as the network emulator [21] to create a WiFi environment where we had multiple domains managed by an SDN controller, along with a mobile node and a corresponding node located in different domains. This emulator is ideal for our case because during emulation, mobile nodes have the ability to move while running standard Linux network software or Internet services. Mininet-WiFi is an extension of the Mininet software-defined network emulator [21]. The Mininet-WiFi developer did not modify any existing Mininet functionality, but added new functionality. We consider a fully connected network with more than three nodes. Cases where the nodes are less than or equal to three are not covered in this article. Indeed, with a number of nodes less than or equal to three, the options for determining the mobility anchor are extremely limited, if not non-existent. The nodes represent the domains of our network.

We prepared four grid network architectures of different sizes: 2 × 3, 3 × 3, 3 × 4, and 3 × 5 and the architecture according to [19], as indicated, respectively, in Figure 7, Figure 8, Figure 9, Figure 10, and Figure 4. The context of our study takes place after a reconnection of the mobile node in the new domain, thus we have the known position of the corresponding node identified as the source node, and the various destination positions depending on the size of the grid to evaluate the position of the mobile node. To demonstrate the effectiveness of our method, we study two cases. The first one to determine the route taken and the second one to calculate the data transfer delay.

When the architecture consists of more than three domains, we count the paths between a source and a destination passing through the chosen anchor domain according to the first algorithm (Algorithm for Anchor Domain Selection) by only moving up and down and from left to right in the matrix (without node reuse). Then, we calculate the different metrics according to the second algorithm (Algorithm for Route Selection).

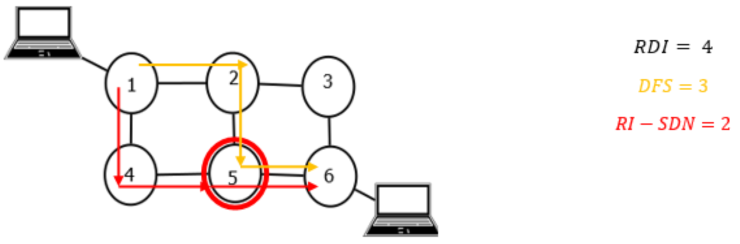


Figure 7. Architecture 2 × 3.

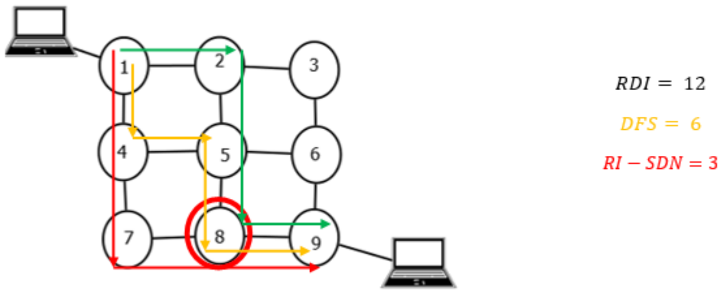


Figure 8. Architecture 3 × 3.

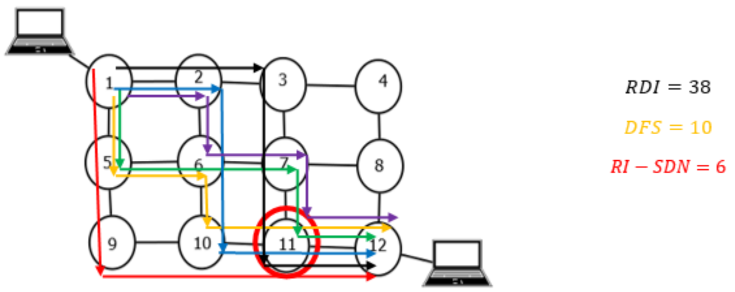


Figure 9. Architecture 3 × 4.

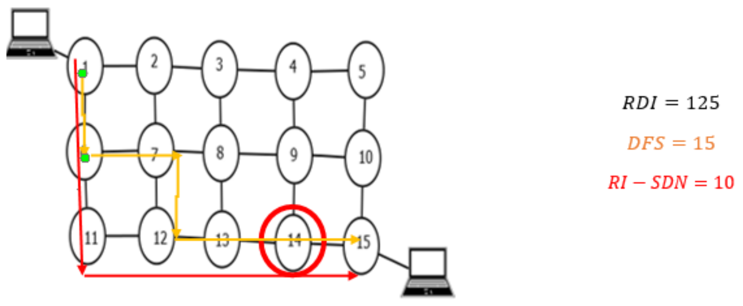


Figure 10. Architecture 3 × 5.

Thus, for an architecture of size m × n, if we only move down or to the right to go from the source to the destination (without cycles), the number of paths is determined by an RDI algorithm (path counting algorithm using Pascal's method). According to the tests in [19], the paths used are those with fewer hops. A Depth-First Search (DFS) algorithm allows us to obtain these paths. We conducted experiments to verify the effectiveness of our approach.

In Mininet-WiFi, we modified the example provided by the simulator for mobility by adding additional domains based on the architecture, and we established links between these domains to represent our architectures. The simulation tests were conducted with variable bandwidths and delays on each link. We repeated the experiment 1000 times for each architecture

5.3. Results Analysis

In this section, we present the most important results pertaining to our evaluation of RI-SDN through simulations. The results are documented in Figures 11 and 12.

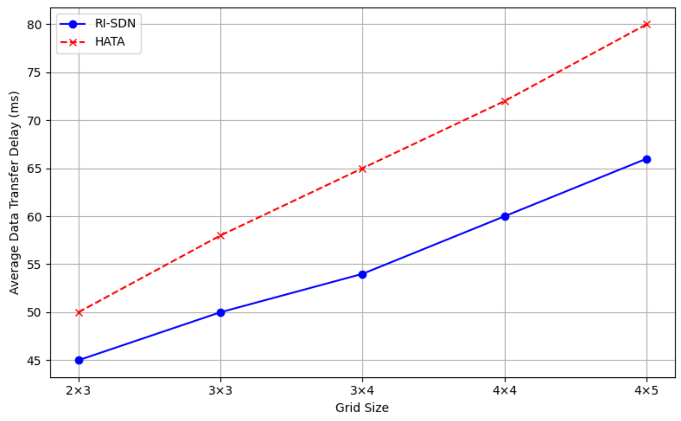


Figure 11. Data transfer delay.

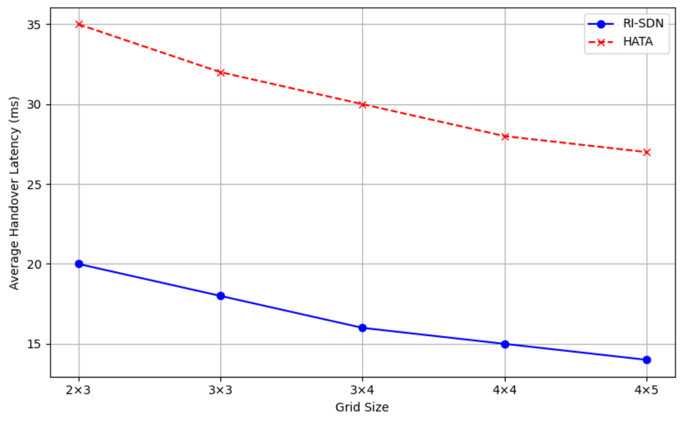


Figure 12. Handover latency.

5.4. Data Transfer Delay

The data transfer delay depends on the available bandwidth and the number of hops. Higher bandwidth and fewer hops reduce the data transfer delay between nodes.

Figure 11 presents a comparison of the average data transfer delay for RI-SDN (i.e., our method) and HATA across five different grid sizes (2 × 3, 3 × 3, 3 × 4, 4 × 4, 4 × 5). The average delays shown were obtained from 1000 runs of our simulation setup for each grid configuration. Note that HATA denotes the method proposed by Hata et al. in [19].

As shown in this figure, RI-SDN consistently maintains lower data delay transfer across all grid sizes compared to HATA.

For smaller grids (2 × 3, 3 × 3), the performance difference between RI-SDN and Hata et al. [19] is minimal, with RI-SDN having a slight advantage. This is because both methods have a relatively small number of possible routes, and the overhead of route selection is low.

However, as the grid size increases (3 × 4, 4 × 4, 4 × 5), RI-SDN begins to outperform [19] more significantly. By filtering out routes with lower bandwidth capacities, RI-SDN avoids congestion and inefficient paths, leading to lower average delays. In contrast, HATA does not apply such strict filtering, resulting in higher delays as more suboptimal routes are considered.

The largest grid size (4 × 5) shows the most considerable difference in delay performance. RI-SDN's average delay increases to 66 ms, while [19]'s delay reaches 80 ms, demonstrating RI-SDN's superior scalability and efficiency in larger, more complex networks.

RI-SDN outperforms the method proposed by Hata et al. in [19] (which is denoted by HATA) in terms of data transfer delay for several reasons. RI-SDN selects routes that simultaneously optimize bandwidth and reduce the number of domains traversed, which effectively reduces transfer delay. Furthermore, unlike [19], which uses a less restrictive route selection algorithm, RI-SDN applies rigorous filtering to avoid suboptimal paths with bandwidth below the required threshold.

5.5. Handover Latency

In this section, we focus on the handover latency—the delay experienced when a mobile node switches from one access point (AP) to another. The goal is to compare the handover performance between RI-SDN and Hata et al. [19] across different grid sizes, with a focus on how quickly the handover occurs and how each method impacts this latency.

Figure 12 illustrates the handover latency observed in RI-SDN and [19] across various grid sizes (2 × 3, 3 × 3, 3 × 4, 4 × 4, 4 × 5). The handover latency refers to the delay incurred when a mobile node switches from one access point to another. The average handover latencies shown were obtained from 1000 runs of our simulation setup for each grid configuration.

As shown in this Figure, RI-SDN consistently exhibits lower handover latency compared to HATA. For the smallest grid size (2 × 3), RI-SDN achieves an average latency of 20 ms, while [19]'s latency is significantly higher at 35 ms. This trend continues across all grid sizes, with RI-SDN maintaining a lower latency as the grid size increases. Note that HATA denotes the proposed method by Hata et al. in [19].

For larger grids (4 × 4 and 4 × 5), the difference in handover latency narrows slightly, but RI-SDN still outperforms HATA. Specifically, in the 4 × 5 grid, RI-SDN reduces the handover latency to 14 ms, compared to HATA's 27 ms. The efficiency of RI-SDN in minimizing route recalculations and focusing on high-bandwidth routes helps to reduce the overall handover time.

This confirms that in networks with frequent handovers, RI-SDN provides a more reliable and efficient solution, ensuring minimal latency during transitions. Meanwhile, HATA, while offering route diversity, may incur additional delays during handovers, which could impact performance in latency-sensitive applications.

RI-SDN provides better performance than [19] in terms of handover latency for several reasons. RI-SDN determines optimal routes upstream via the anchor, which reduces the need for recalculation when changing domains, unlike [19], where paths are recalculated at each transition. Moreover, the RI-SDN method places the mobility anchor as close as possible to the correspondent, thus reducing the number of hops required after a handover.

6. Conclusions

This paper presented an anchor-based method for inter-domain mobility management in SDN architectures. The proposed method addresses the challenge of reducing data transfer delay during mobile node movement across administrative domains.

The key contributions involve modeling the problem as an efficient route selection task, proposing algorithms for anchor-domain selection based on proximity to the data destination, and selecting routes that traverse the chosen anchor while considering bandwidth and the number of traversed domains.

However, it is essential to acknowledge certain limitations of our study. An evaluation of the traffic load between controllers will be included in future work. Approaches to limiting this load, such as the use of hierarchical or decentralized controllers, could be explored to guarantee the scalability of RI-SDN in large networks. Additionally, incorporating security considerations into the anchor and route selection and route processes is crucial. Future work could explore incorporating real-time network conditions and security issues.

Author Contributions: Conceptualization, A.A.J.R.K., A.F.A. and A.D.K.; methodology, A.A.J.R.K., A.F.A. and M.B.; implementation, A.A.J.R.K. and Z.F.O.T.; writing—original draft preparation, A.A.J.R.K. and A.F.A.; writing—review and editing, Z.F.O.T. and A.F.A.; supervision, M.B. and A.D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Acknowledgments: We would like to thank the anonymous reviewers for their insightful comments and suggestions that significantly improved the quality of this manuscript. Their feedback helped us to strengthen the clarity, focus, and overall impact of our work.

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

References

1. Munirathinam, S. Industry 4.0: Industrial Internet of Things (IIOT). In *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*; Elsevier: Amsterdam, The Netherlands, 2020; pp. 129–164. [CrossRef]
2. Masoudi, R.; Ghaffari, A. Software defined networks: A survey. *J. Netw. Comput. Appl.* **2016**, *67*, 1–25. [CrossRef]
3. Isyaku, B.; Mohd Zahid, M.S.; Bte Kamat, M.; Abu Bakar, K.; Ghaleb, F.A. Software Defined Networking Flow Table Management of OpenFlow Switches Performance and Security Challenges: A Survey. *Future Internet* **2020**, *12*, 147. [CrossRef]
4. Chattopadhyaya, S.; Sahoo, A.K. Software defined networks: Current problems and future solutions. *Mater. Today Proc.* **2022**, *49*, 2989–2993. [CrossRef]
5. Badotra, S.; Panda, S.N. Software-Defined Networking: A Novel Approach to Networks. In *Handbook of Computer Networks and Cyber Security*; Springer International Publishing: Cham, Switzerland, 2020; pp. 313–339. [CrossRef]
6. Zhao, Z.; Fan, X.; Xie, X.; Mao, Q.; Zhao, Q. AP-SDN: Action Program enabled Software-Defined Networking Architecture. *KSII Trans. Internet Inf. Syst.* **2023**, *17*, 1894–1915. [CrossRef]
7. Hassan, M.; Gregory, M.A.; Li, S. Multi-Domain Federation Utilizing Software Defined Networking—A Review. *IEEE Access* **2023**, *11*, 19202–19227. [CrossRef]
8. Lin, P.; Bi, J.; Wang, Y. East-West Bridge for SDN Network Peering. In *Communications in Computer and Information Science*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 170–181. [CrossRef]
9. Xiao, P.; Qu, W.; Qi, H.; Li, Z.; Xu, Y. The SDN controller placement problem for WAN. In Proceedings of the 2014 IEEE/CIC International Conference on Communications in China (ICCC), Shanghai, China, 13–15 October 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 220–224. [CrossRef]
10. Idri, A.; Amazal, F.A.; Abran, A. Analogy-based software development effort estimation: A systematic mapping and review. *Inf. Softw. Technol.* **2015**, *58*, 206–230. [CrossRef]
11. Wang, Y.; Bi, J.; Zhang, K. Design and Implementation of a Software-Defined Mobility Architecture for IP Networks. *Mob. Netw. Appl.* **2015**, *20*, 40–52. [CrossRef]
12. Gundavelli, S.; Leung, K.; Devarapalli, V.; Chowdhury, K.; Patil, B. Proxy Mobile ipv6. 2007. Available online: https://datatracker.ietf.org/doc/rfc5213/ (accessed on 22 October 2024).
13. Nguyen, T.T.; Bonnet, C.; Harri, J. SDN-based distributed mobility management for 5G networks. In Proceedings of the 2016 IEEE Wireless Communications and Networking Conference, Doha, Qatar, 3–6 April 2016 ; IEEE: Piscataway, NJ, USA, 2016. [CrossRef]
14. Duan, X.; Ma, X.; Wang, T.; Meng, Y.; Gu, L.; Zhao, X.; Dong, C.; Gong, X.; Bi, S.; Wang, L.; et al. Research on IPv6 mobility management mechanism based on SDN. In Proceedings of the 2nd International Conference on Computer Vision, Image, and Deep Learning, Liuzhou, China, 25–27 June 2021; Cen, F., bin Ahmad, B.H., Eds.; SPIE: Bellingham, WA, USA, 2021. [CrossRef]
15. Cordova, R.T.; Gondim, P.R.L.; Llerena, Y.P.; Lloret, J. SDN-DMM for intelligent mobility management in heterogeneous mobile IP networks. *Int. J. Commun. Syst.* **2019**, *32*, e4140. [CrossRef]
16. Giust, F. Distributed Mobility Management for a Flat Architecture in 5G Mobile Networks: Solutions, Analysis and Experimental Validation. Ph.D. Thesis, Universidad Carlos III de Madrid, Madrid, Spain, 2015.
17. Raza, S.M.; Kim, D.S.; Shin, D.; Choo, H. Leveraging proxy mobile IPv6 with SDN. *J. Commun. Netw.* **2016**, *18*, 460–475. [CrossRef]
18. Adon Jean Rodrigue Kanda, A.; Ferdinand Atta, A.; Babri, M.; Dooguy Kora, A. Mobility Management in a Cross-Domain SDN Architecture Leveraging the ipv6 Mobile Proxy. *Int. J. Adv. Res.* **2023**, *11*, 472–484. [CrossRef] [PubMed]

19. Hata, M.; Soylu, M.; Izumi, S.; Abe, T.; Suganuma, T. A Design of SDN Based IP Mobility Management Considering Inter-Domain Handovers and Its Evaluation. *Adv. Sci. Technol. Eng. Syst. J.* **2017**, *2*, 922–931. [CrossRef]
20. Bradai, A.; Benslimane, A.; Singh, K.D. Dynamic anchor points selection for mobility management in Software Defined Networks. *J. Netw. Comput. Appl.* **2015**, *57*, 1–11. [CrossRef]
21. Fontes, R.R.; Afzal, S.; Brito, S.H.B.; Santos, M.A.S.; Rothenberg, C.E. Mininet-WiFi: Emulating software-defined wireless networks. In Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 9–13 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 384–389. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.