Network Slicing for mMTC and URLLC Using Software-Defined Networking with P4 Switches

Yan-Jing Wu 1, Wen-Shyang Hwang 2,*, Chih-Yi Shen 2 and Yu-Yen Chen 2

1 Department of Information Technology and Communication, Shih Chien University, Kaohsiung Campus, Kaohsiung 84550, Taiwan; yanjing@g2.usc.edu.tw
2 Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 80778, Taiwan; f109154152@nkust.edu.tw (C.-Y.S.); f108154152@nkust.edu.tw (Y.-Y.C.)

* Correspondence: wshwang@nkust.edu.tw

Abstract: Massive machine-type communication (mMTC) and ultra-reliable low-latency communication (URLLC) are two key services in fifth-generation (5G) mobile wireless networks. These networks have been developed with extremely high service quality requirements: scalability for mMTC and reliability with low latency for URLLC. Fifth-generation network slicing will play a key role in supporting the distinct requirements of various services. Software-defined networking (SDN), a promising technology for network softwarization, physically separates the network control plane from the data plane by centrally controlling switches with an SDN controller. However, control channel bottleneck and processing delays due to this centralized control may reduce the scalability, reliability, and security of SDN. This paper proposes an SDN framework with programming protocol-independent packet processor (P4) switches (SDNPS), and defines a packet format containing in-band network telemetry data to simultaneously support heavy Internet of Things and URLLC traffic in 5G network slices. The method both satisfies the requirements of mMTC and URLLC and alleviates the load on the SDN controller. P4 is an advanced switch interface technology that provides enhanced stateful forwarding and reveals a persistent state on the SDN data plane. To demonstrate the superiority of SDNPS, simulations are performed on conventional SDNs and SDNPS. SDNPS outperforms the other schemes in terms of average throughput, packet loss ratio, and packet delay for both the mMTC and URLLC network slices.

Keywords: mMTC; URLLC; network slicing; INT; P4

1. Introduction

Numerous studies have investigated next-generation mobile communication network technologies, such as network architectures, communication protocols, information security and smart applications [1,2]. The performance of Internet of Things (IoT), in which connected devices transfer sensor data over the internet, is a key consideration in the development of fifth-generation (5G) networks. At the dawn of the era of 5G communication networks in 2020, the number of connected IoT devices had already reached 50 billion; each person had an average of 10.3 connected devices (Figure 1). By 2030, the number of IoT device nodes is expected to grow to 80 billion, and each person is expected to have an average of 20.5 connected devices [2].

The International Telecommunication Union Radio-communication Sector divided 5G communication services into three categories: mMTC (massive machine-type communication), URLLC (ultra-reliable low-latency communication) and eMBB (enhanced mobile broadband) [3,4]. Figure 2 displays the nine service quality requirements, such as latency, and the corresponding importance of these requirements for each of the three categories, as indicated by the length of the radial bars [5]. mMTC and URLLC have two extremely different service quality requirements of scalability and reliability with low latency, respectively. Fifth generation and future communication networks should be able to integrate...
heterogeneous services with different latency requirements, reliability scores, and data transfer rates in an environment with shared network infrastructure instead of providing individual network solutions for each service type. In other words, a communication network designed for IoT devices alone is unsuitable for supporting heterogeneous services in a shared network environment.

Figure 1. Number of connected IoT devices (2005–2030).

Figure 2. Service categories and quality requirements in 5G networks.

Fifth-generation network slicing (NS), a technology for integrating multiple heterogeneous services in a shared network infrastructure, is expected to be widely used in future networks. In NS, virtualization is used to slice a physical 5G network into multiple isolated end-to-end (E2E) logical networks of varying sizes and structures that are dedicated to different use cases. The equipment, access, transmission, and core network of each virtual network (i.e., network slice) are independent [5,6]. By using network slices, companies could ask internet service providers to provide a customized 5G network. For example, hospitals, with high security and reliability requirements, could use a network slice that enables low-latency, high-reliability applications such as remote surgery. With network softwarization, including software-defined networking (SDN), network function virtualization (NFV) [7,8], etc., different virtual networks can serve different functions, optimizing network resources usage. Figure 3 illustrates the 5G NS architecture as left and right blocks (framed by dotted lines). The left block represents the actual slicing implementation, and the right block represents the slice management and configuration. The left implementation
block is a multi-layer architecture comprising the service layer, network function layer, and infrastructure layer. The right control block is a centrally processed network unit, called the network slice controller, which monitors and manages the functions between any two of the three layers in the left block so as to efficiently coordinate the multiple coexisting network slices [5,9].

SDN is a promising technology for network softwarization for realizing 5G NS. Potential network slices can be defined in accordance with the relevant requirements of each SDN user. SDN can include artificial intelligence algorithms and provide flexible and programmable applications or services [10,11]. The Open Network Foundation [12] defined SDN as ‘The physical separation of the network control plane from the forwarding plane, and where a control plane controls several devices.’ This separation of the control plane from the data plane enables SDN to flexibly and centrally control the entire network and to quickly respond to changes in network conditions, business, markets, and user demands. The major difference between contemporary network operations and SDN is the creation of a virtualized control plane that can execute intelligent management decisions for network functions, remediating the gap between service delivery and network management (Figure 4) [13]. With an SDN, network control can be achieved directly and programmatically through a standardized southbound interface. For instance, the commonly used OpenFlow protocol [14] defines a communication mechanism between the controller in the control plane and the forwarding components in the data plane.

**Figure 3.** The 5G NS architecture.

**Figure 4.** Contemporary network and SDN.
The state information of all packets in the network relies on the actions of the controller rather than those of the OpenFlow switches in the data plane, substantially increasing the burden on the SDN controller [15]. The processing delay between the SDN controller and switches and the control channel bottleneck [16,17] may reduce the scalability, reliability, and security of 5G NS. Several advanced interface protocols for the SDN controller and switches have been proposed to provide enhanced state forwarding rules and enable data plane switches to obtain persistent state information, such as OpenState [18], programming protocol-independent packet processors (P4) [19], protocol-oblivious forwarding [20], stateful data plane architecture [21], and stateful network-wide abstractions for packet processing [22]. Among these, P4 is a protocol-independent, high-level programming language that enables programmers to modify how SDN switches process data packets. By maintaining the status of forwarding packets with P4, 5G NS can both support a variety of communication protocols and data transmission mechanisms and achieve independent monitoring, control, and management. Moreover, P4 implements a top-down design concept; the programmer or user determines the function of the underlying switch and how it should process packets, and the underlying communication protocol does not limit the services that the network can support.

In order to provide the heterogeneous service quality requirements in a 5G network and to achieve easy management, scalability, and high responsiveness to variations in network conditions, this paper proposes a framework of software-defined networking with P4 switches (SDNPS) to support both mMTC and URLLC in 5G network slices. The main contributions of the proposed SDNPS are the design of an SDN framework with P4 switches and defining a packet format containing in-band network telemetry (INT) [23] data that simultaneously support massive IoT and URLLC traffic in the core of 5G network slices. Thus, the extremely different service requirements of mMTC and URLLC can be satisfied, and the control load on the SDN controller can be alleviated. We assume that all of the switches in the data plane would support P4 and divide the forwarding rules of the P4 switches into two parts: one for mMTC and one for URLLC services. In the SDNPS, INT enables data packets to carry network status information such that P4 switches on the forwarding path can update the included status information before forwarding packets, thereby reducing the control load on the SDN controller. To the best of our knowledge, this paper is the first to implement 5G NS in SDN with P4 switches to support both mMTC and URLLC. Finally, we perform simulations of 5G NS on SDNs with traditional or P4 switches to demonstrate the effectiveness and superiority of the proposed SDNPS. The performance measures include the average throughput, packet loss ratio, and packet delay for the mMTC and URLLC network slices.

The remainder of this paper is organized as follows. Section 2 explores the related literature. Section 3 describes the framework and operating procedure of our proposed SDNPS. Section 4 analyzes and compares the simulation results. Finally, concluding remarks and future works are given in Section 5.

2. Related Works

Most studies on IoT in modern communication networks have aimed to reduce the high collision rate of the random access channel (RACH) competition mechanism in 5G. If numerous mMTC devices appear in a radio access network (RAN), the number of mMTC devices that can successfully connect is drastically reduced, reducing the mMTC connection density. Studies on this phenomenon can be broadly divided into those on LTE-machine (LTE-M) [24,25] and narrowband IoT (NB-IoT) communications [26–28]. These two technologies use different cell coverage areas and reserve different numbers of resource blocks (RBs) during a frame period. By adopting simpler coding techniques or lower data transmission rates for dedicated RBs, the mMTC connection density can be increased. Table 1 presents a comparison of the LTE-M and NB-IoT parameters based on 3GPP-Release 13 [29]. Several studies have applied artificial intelligence algorithms to RACH collision detection [30–33]; these algorithms can dynamically adjust the time that each mMTC device
accesses the RAN to reduce the probability of access collisions if numerous mMTC devices simultaneously attempt to connect.

Table 1. Comparison of LTE-M with NB-IoT.

| Parameters          | LTE-M                  | NB-IoT                  |
|---------------------|------------------------|-------------------------|
| Frequency band      | All                     | Some FDD bands          |
| Cell bandwidth      | 1.4–20 MHz              | 180 KHz                 |
| UE bandwidth        | 1.4 MHz                 | 180 KHz                 |
| Duplex mode         | TDD, FDD, Half duplex FDD | Half duplex FDD        |
| Mobility            | High                    | Low                     |
| Receiving antenna   | 1                       | 1                       |
| Maximum power       | 20/23 dBm               | 20/23 dBm               |
| Peak DL data rate   | 1 Mbps                  | 250 Kbps                |
| Peak UL data rate   | 1 Mbps                  | 250 Kbps                |

Key 5G network services include not only mMTC, but also URLLC and must support their heterogeneous requirements. Relevant research on supporting both mMTC and URLLC in 5G networks typically covers one of the following groups of topics:

1. Bandwidth resource allocation, differentiating access times, or RAN frequency reuse.
2. Dividing the entire network into multiple virtual E2E networks (i.e., network slices) by using virtualization technology. Network slices are independent of each other in their equipment, access, transport, and core networks.

Many authors have proposed solutions for 5G networks with numerous coexisting IoT devices and URLLC services [3,13,34–44]. The authors of [3,39–42] proposed mechanisms for bandwidth resource allocation, differentiation of access time, or frequency reuse in the RAN of 5G networks to enable the provision of multiple service types by reducing the probability of access collisions and increasing the efficiency of bandwidth use. Pokhrel et al. [3] indicated that even a network dominated by IoT services may require handling short-term emergencies, such as those at large-scale disaster sites or the wireless monitoring of factory automation, in which quality requirements are similar to those of URLLC. Therefore, 5G network designs must be effective for both mMTC and URLLC services.

Fifth-generation NS can simultaneously meet the heterogeneous quality requirements of mMTC and URLLC [13,34–38]. Fifth-generation NS separates the network into multiple virtual E2E networks (i.e., network slices); the equipment, access, transport, and core networks belonging to a network slice are independent of each other. Different network slices correspond to different services to ensure that each service’s distinct quality requirements can be satisfied. The authors of [34] observed that NS in an RAN can support delay-sensitive applications. To satisfy the transmission rate and delay requirements of different services, the authors proposed a RAN slicing mechanism for deterministic periodic, deterministic nonperiodic, and nondeterministic traffic. Yang et al. [35] established an analytical model to determine the optimal RAN bandwidth resource slices to maximize the connection density of IoT devices with an energy-saving method while supporting URLLC services. However, E2E coordination and management are necessary for NS. RAN slicing alone cannot guarantee that quality requirements are met in E2E, especially if the network equipment or status changes.

Hence, the authors of [13,36–38,43–45] proposed using SDN and NFV to implement 5G NS and facilitate network management, scaling, and adaptation. In [13,36], the functional concepts required to implement E2E NS at each vertical layer in SDN were discussed. Furthermore, the authors of [37] proposed a general 5G NS model for network components and computing resources from the edge to the core, with the principle that lightweight and complex computing should be performed at the edge and the upper layer of the infrastructure, respectively. Chahbar et al. [38] claimed to be the first to fully define each module and its function corresponding to each end user in the RAN, core network, and transmission network in terms of SDN and NFV technologies to achieve E2E NS.
mentioned in Section 1, on an SDN with traditional switches, the state information of all packets cannot be maintained in the data plane and relies only on the actions of the controller in the control plane, which may result in processing delays and a control channel bottleneck between the controller and switches. The solution to this problem is to use higher-level interface standards, such as P4 [19], between the controller and switches. Alvarez-Horcajo et al. [43] proposed an enhanced SDN switch with forwarding actions specified by the P4 language. Its SDN functions were enabled by using the P4 runtime as the control plane protocol to specify the forwarding rules in the programmable data plane. The enhanced switch could autonomously define forwarding rules by using P4 registers, thereby preventing SDN controller overload. The authors of [44] presented a programmable data transmission path design between user equipment-RAN and user plane function for industrial 5G networks. They also discussed methods of supporting URLLC services with lower latency in 5G networks by using P4 switches as required in industrial applications. Paolucci et al. [45] illustrated the potential of the P4 language, aiming to show its innovative functionalities at the data plane level, and provided five use cases. Despite that one of the five use cases mentioned in [45] is slicing and multi-tenancy, there is no description of how to implement and manage both mMTC and URLLC network slices and no definition for the format of packets carrying INT data in an SDN environment with P4 switches. In a word, these studies on SDNs for implementing 5G NS either have not fully considered the heterogeneous service quality requirements of mMTC and URLLC or have not completely defined the operation of the P4 switches.

3. The Proposed SDNPS

3.1. SDNPS Framework

First, we present the SDNPS architecture. The client services are divided into two categories—mMTC and URLLC—which are assigned different network slices (Figure 5). Each network slice is logically independent of the other in terms of network resources, including equipment, access, transmission, and core networks. The SDN comprises three planes: application, control, and data. The SDN controller in the control plane uses client context-aware forwarding rules or server context-aware strategies to manage the network slices. The client context comprises the client identity, service quality requirements, and virtual network resources to satisfy all incoming end user requests. With INT [23], each P4 switch along a path in the data plane can directly record the network status, including the queue occupancy, link throughput, and processing delay, from an INT header added by the ingress switch before forwarding. When a data packet carrying INT data arrives at the egress switch, the INT data are duplicated and sent upward to the INT data collector module in the application plane to monitor the status of each switch on a flow path. The NS controller is responsible for changing paths and informing the reroute module of path reconfigurations. The P4 switches are activated to update the forwarding rules when necessary. The SDNPS can both reduce the burden on the SDN controller and achieve real-time control of transmission paths in network slices. Thus, both mMTC and URLLC can meet their respective service quality requirements by using 5G NS with SDN and P4 switches.
the egress switch, the INT data are duplicated and sent upward to the INT data collector module in the application plane to monitor the status of each switch on a flow path. The NS controller is responsible for changing paths and informing the reroute module of path reconfigurations. The P4 switches are activated to update the forwarding rules when necessary. The SDNPS can both reduce the burden on the SDN controller and achieve real-time control of transmission paths in network slices. Thus, both mMTC and URLLC can meet their respective service quality requirements by using 5G NS with SDN and P4 switches.

Figure 5. SDNPS framework.

3.2. Data Packet Format

INT is a new monitoring technology based on P4. The main concept is collecting and reporting the network status in the data plane without the intervention of the control plane. In INT, network status information (for example, the switch queue occupancy, packet output rate, and processing delay) is encapsulated into a data packet or a predefined control packet by each P4 switch along a data transmission path. The packet carrying the collected network status information is extracted and sent to the application plane for further analysis prior to being forwarded to its destination. Thus, the load of bidirectional information exchange between the control plane and the data plane is considerably reduced, and up-to-date, accurate network status information in the data plane can be obtained and acted upon as quickly as possible. Figure 6 illustrates the procedure for INT. The INT source, INT transit hop, and INT sink are all P4 switching devices that support INT and represent the start, middle, and end point of a flow path, respectively. As shown in Figure 6, the INT procedure is divided into three steps:

1. When a data packet from the source host enters the data plane, the INT source can add an INT header to tell the subsequent P4 switches which information (called 'INT data') they should write.
2. The INT transit hop can write its own INT data into the specified field in accordance with the content indicated by the INT header.
3. When a data packet carrying INT data arrives at the INT sink, all of the recorded INT data along a flow path are extracted and reported to the collector module. The original data packet is then forwarded to the destination host.
Up to now, the P4 language and related specifications provide the following eight INT Metadata [46].

1. Switch ID: the unique ID of a switch.
2. Ingress port ID: the port ID of a received INT packet.
3. Ingress timestamp: the time that an INT packet is received.
4. Egress port ID: the port ID of a forwarded INT packet.
5. Hop latency: the processing time of an INT packet in the switch.
6. Egress port transmission (TX) link utilization: the link bandwidth utilization when an INT packet is forwarded.
7. Queue occupancy: queue occupancy observed at the switch when an INT packet is forwarded.
8. Queue congestion status: congestion status of the current queue.

Among the above Metadata, hop latency, egress port TX link utilization, and queue occupancy are used by the proposed SDNPS to manage the flow paths in the mMTC and URLLC slices. INT does not specify the packet encapsulation format. The format of the INT header and which INT data each P4 switch should write can be defined by the administrator. SDNPS writes 0 × 0801 in the EtherType field of the Ethernet header to identify if it is a P4 switch (Figure 7). If it is a P4 switch, the field ‘Flow_h’ follows the EtherType field. Flow_h (i.e., Flow Header) is mainly used to inform P4 switches as to whether the forwarded packet contains INT data because the INT data for each P4 switch are sent only periodically—not with every data packet. Figure 8 presents the format of the 4-byte flow header (denoted as Flow_h), comprising Protocol (1 byte), Counter (1 byte), and RouteID (2 bytes). The Protocol field is set to 0 × 00 and 0 × 01 in our framework to identify whether the data after the Flow Header are an IP header or INT data, respectively. The Counter field records the number of P4 switches a packet passes through; each P4 switch adds one. The RouteID is configured in accordance with the source IP and destination IP. Each RouteID is unique and indicates a flow path; packets containing INT data include these data from each P4 switch along the flow path indicated by the RouteID. Figure 9 presents the format of the 5-byte INT data (denoted as INT_h) for a P4 switch, comprising SwitchID (1 byte), Qoccupancy (1 byte), Qlatency (2 bytes), and Link_utilization (1 byte). The SwitchID field is used to record the identity of the switch, the Qlatency field is used to record the processing time, the Qoccupancy field is used to record the utilization ratio of the egress queue, and the Link_u field is used to record the bandwidth utilization of the egress link.

Figure 6. INT procedure.
3.3. Pseudocode of SDNPS

We assume that all switches in the data plane of SDN would support P4. Prior to describing the SDNPS pseudocode, we first define the symbols used by the SDNPS. As summarized in Table 2, $T_s$ denotes the sampling period for INT data. $H_d$, $H_q$, and $H_u$ are the thresholds of delay, queue occupancy, and link utilization, respectively, at any switch that triggers a path change in the same slice because a bottleneck at any switch in a flow path leads to severe performance degradation for the path. $D_i$, $Q_i$, and $L_i$ denote the processing delay, queue occupancy, and link utilization of switch $i$, respectively.

Table 2. Symbols and denotations.

| Symbols | Denotations |
|---------|-------------|
| $T_s$   | Sampling period for INT data |
| $H_d$   | Threshold of delay for each switch |
| $H_q$   | Threshold of queue occupancy for each switch |
| $H_u$   | Threshold of link utilization for each switch |
| $D_i$   | Hop Delay of switch $i$ |
| $Q_i$   | Queue occupancy of switch $i$ |
| $L_i$   | Link utilization of switch $i$ |

As listed in Table 3, when each URLLC or mMTC packet arrives at the ingress switch, it is forwarded through the assigned flow path in its corresponding slice. The protocol field in Flow_h is set to 0 × 01 every 0.5 s, and each switch along a flow path identified by the RouteID field adds its INT data following Flow_h before forwarding the packet. The INT data comprise the queue occupancy, link throughput, and processing delay for each switch $i$ along the flow path. When a packet carrying INT data arrives at the egress switch, the INT data are duplicated and sent to the application plane to collect the status of each switch on the flow path. The network slice controller is responsible for changing paths in each slice and informing the reroute module of any path reconfigurations. mMTC packets are rerouted to the other paths in the mMTC slice if the queue occupancy or link utilization of at least one switch is larger than $H_q$ or $H_u$, respectively. URLLC packets will be rerouted to the other paths in the URLLC slice if the processing delay, queue occupancy, or link utilization of at least one switch is larger than $H_d$, $H_q$, or $H_u$, respectively, because URLLC services are time critical. All P4 switches on a flow path are activated to update the forwarding rules when necessary.
Table 3. Pseudocode of SDNPS.

| Network Slicing |
|------------------|
| **m** paths in the mMTC slice: \( M = \{ f_j \mid 1 \leq j \leq m \} \)  |
| **r** paths in the URLLC slice: \( R = \{ f_r \mid 1 \leq j \leq r \} \)  |
| **Packet forwarding:**  |
| 1. If (an mMTC flow)  |
| 2. Assign the path with the minimum load in \( M \);  |
| 3. \( \text{Flow}_h.\text{RouteID} = f_t_j \);  |
| 4. If (a URLLC flow)  |
| 5. Assign the path with the minimum load in \( R \);  |
| 6. \( \text{Flow}_h.\text{RouteID} = f_r_j \);  |
| 7. \( \text{Flow}_h.\text{Protocol} = 0 \times 01 \) every \( T_s \) seconds at the ingress switch;  |
| 8. If (\( \text{Flow}_h.\text{Protocol} = 0 \times 01 \))  |
| 9. \( \text{Flow}_h.\text{Counter}++ \);  |
| 10. Record \( \text{INT} \) data by concatenating an \( \text{INT}_h \);  |
| 11. Forward the packet carrying \( \text{INT} \) data to the next hop;  |
| 12. If (at the egress switch)  |
| 13. Duplicate all of the \( \text{INT} \) data;  |
| 14. Send the above \( \text{INT} \) packet to the \( \text{INT} \) Data Collector;  |
| 15. Forward the original packet to the destination;  |
| 16. Else (\( \text{Flow}_h.\text{Protocol} = 0 \times 00 \))  |
| 17. \( \text{Flow}_h.\text{Counter}++ \);  |
| 18. Forward the packet to the next hop;  |
| **Network slice controller:**  |
| 19. If (\( Q_i > H_q \) or \( L_i > H_u \) on RouteID = \( f_t_j \))  |
| 20. Reroute to another path in \( M \) that can afford the mMTC flow;  |
| 21. If (\( D_i > H_d \) or \( Q_i > H_q \) or \( L_i > H_u \) on RouteID = \( f_r_j \))  |
| 22. Reroute to another path in \( R \) that can afford the URLLC flow; |

4. Performance Evaluation

In this section, the performance of SDNPS is evaluated through comparison with an SDN with traditional switches. The performance measures include the average throughput, packet loss ratio, and packet delay for mMTC and URLLC network slices.

4.1. Simulation Settings

The SDN topology used for performance evaluation is depicted in Figure 10. The parameters and their values used in the simulation are summarized in Table 4. We set up an SDN environment comprising four hosts and six P4 switches on the Mininet simulator [47]. The well-established Ryu controller is adopted for the control plane [48]. The communication interface between the data and control planes is OpenFlow Protocol V1.3 for traditional switches and SDNPS for the P4 switches. The Behavioral Model version 2 (BMv2) [49], employing the P4 language, is deployed in Mininet. The bandwidth of each link between the two switches is 20, 30, or 60 Mbps (Figure 10). The queue size of each switch is 50 packets. The Iperf tool [50] is used to generate UDP flows at a constant data rate of 50 Kbps for each machine-type device, and the number of machine-type devices varies from 100 to 500. Two URLLC UDP flows are transmitted at 5 and 17 Mbps. The source and destination hosts of the mMTC flows are H1 and H3, respectively, and those of the URLLC flows are H2 and H4, respectively.
The URLLC slice is assigned two flow paths, S1-S3-S5-S6 and S1-S3-S4-S6, with the fewest hops. To achieve high connection density instead of latency, the mMTC slice is assigned two flow paths, S1-S2-S5-S4-S6 and S1-S2-S4-S5-S6. Initially, the first flow path in either slice is always used, and the second flow path is used if the first path has service quality degradation. The sampling period for INT data is 0.5 s. The threshold values for the INT data are listed in Table 4. The simulation duration is 10 s. During the simulation, a URLLC flow of 5 Mbps is generated initially, and another URLLC flow of 17 Mbps is generated after 3 s; 100, 250, or 500 mMTC flows are simultaneously generated from the beginning to the end of the simulation.

4.2. Results and Discussions

The performance of the proposed SDNPS is compared with that of a conventional SDN with either static or dynamic routing. In the conventional SDN, the monitoring period with dynamic routing is fixed at 3 s; no monitoring period is used with static routing.
In this subsection, we obtain and discuss the average throughput, packet loss ratio, and packet delay for the mMTC and URLLC network slices in the conventional SDN and the proposed SDNPS.

4.2.1. URLLC Slice

Figure 11 shows the average packet loss ratio versus time for the URLLC slice. Figure 11a–c display the results for 100, 250, and 500 mMTC devices, respectively. The first flow path (i.e., S1-S3-S5-S6) in the URLLC slice with 5 Mbps of initial URLLC traffic becomes overloaded at 3 s when 17 Mbps of URLLC traffic are transmitted through the flow path. Figure 11 reveals that the average packet loss ratios with SDNPS (blue curves) always reduce to near zero more quickly after the third second than with the conventional SDN with either static (grey curves) or dynamic routing (orange curves), regardless of the total load of mMTC devices. This result occurs because with SDNPS, the INT data for each switch are carried by a data packet every 0.5 s, extracted at the egress switch, and sent to the network slice controller, enabling a rapid response to network congestion. The NS controller thus quickly decides to reroute the 17 Mbps URLLC flow to the second path (i.e., S1-S3-S4-S6) in the URLLC slice. By contrast, for the conventional SDN with dynamic routing, updating the forwarding rules for the switches in the data plane requires substantial information exchange between the SDN controller and every OpenFlow switch; thus, rerouting requires more time. For the conventional SDN with static routing, the average packet loss ratio is increased continuously after the third second because the forwarding rules do not change in the static routing unless the current path fails.

Figure 11. Cont.

(a) No. of mMTC devices = 100

(b) No. of mMTC devices = 250
Figure 11. Average packet loss ratio versus time in the URLLC slice for (a) 100; (b) 250; (c) 500 mMTC devices.

Figures 12–14 show the average throughput, packet delay, and packet loss ratio in the URLLC slice for 100, 250, and 500 mMTC devices. Because URLLC and mMTC are differentiated by NS, the number of mMTC devices does not affect the performance in any of the scenarios. Thus, the behavior of the mMTC slice does not affect that of the URLLC slice because the network slices are inherently independent of each other. However, in all scenarios, SDNPS exhibits considerably better performance in all metrics than do the other two schemes.
4.2.2. mMTC Slice

The key service quality requirement for mMTC is connection density, and the least significant service quality requirement is latency. Figures 15 and 16 show the average throughput and packet loss ratio per node, respectively, versus the number of mMTC devices. Figure 15 reveals that the average throughput per mMTC node with SDNPS (blue bars) is approximately equal to the packet generation rate (i.e., 50 Kbps), regardless of the number of nodes requesting connections. However, a slight reduction in throughput is observed for 500 nodes (i.e., heavy load) because processing time is required for path changes for some connections in the mMTC slice. SDNPS outperforms both the dynamic or and static SDNs, especially under heavy load in the slice. Thus, SDNPS achieves higher connection density. Similarly, Figure 16 reveals the superiority of SDNPS over the other two schemes in terms of the average packet loss ratio per mMTC node.
5. Conclusions

This paper proposes an SDNPS with an INT data packet format to simultaneously support mMTC and URLLC by using NS. The method satisfies the opposing service requirements of both mMTC and URLLC in 5G and alleviates the control load on the SDN controller. In SDNPS, INT data are recorded in data packets forwarded by P4 switches every sampling period, and these data are collected by an INT data collector module in the application plane. Before forwarding a packet with an INT header added by the ingress switch, each P4 switch along a flow path in the data plane records its current status from this header. The INT data comprise the queue occupancy, link throughput, and processing delay. When a packet carrying INT data arrives at the egress switch, the INT data are duplicated and sent to the application plane to determine the status of each switch along the flow path. The network slice controller is responsible for changing paths in each slice and informing the reroute module of these path reconfigurations. The P4 switches are instructed to update their forwarding rules as necessary. The SDNPS is verified through a simulation to substantially improve 5G NS performance for supporting mMTC and URLLC in the average throughput, packet loss, and packet delay, compared with a conventional SDN with dynamic or static routing. The proposed SDNPS is expected to improve the core network used in 5G NS. We intend to use SDNPS with a RAN to completely realize E2E 5G NS.

Author Contributions: Conceptualization, Y.-J.W.; Data curation, C.-Y.S.; Formal analysis, Y.-J.W.; Funding acquisition, W.-S.H.; Investigation, C.-Y.S.; Methodology, Y.-J.W.; Project administration, W.-S.H.; Resources, W.-S.H.; Software, C.-Y.S. and Y.-Y.C.; Supervision, C.-Y.S. and Y.-Y.C.; Writing—original draft, Y.-J.W.; Writing—review and editing, W.-S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported in part by Taiwan Ministry of Science and Technology under Grant No.: MOST 110-2221-E-158-001 and MOST 110-2221-E-992-029-MY2.

Data Availability Statement: The source cord of this work and data obtained via the simulation are available from the corresponding author upon request.

Acknowledgments: We thank the anonymous reviewers for their constructive comments, which improved the quality of this paper.

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

References
1. Palattella, M.R.; Dohler, M.; Grieco, A.; Rizzo, G.; Torsner, J.; Engel, T.; Ladid, L. Internet of Things in the 5G Era: Enablers, Architecture, and Business Models. IEEE J. Sel. Areas Commun. 2016, 34, 510–527. [CrossRef]
2. Chettri, L.; Bera, R. A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems. IEEE Internet Things J. 2020, 7, 16–32. [CrossRef]
3. Pokhrel, S.R.; Ding, J.; Park, J.; Park, O.-S.; Choi, J. Towards Enabling Critical mMTC: A Review of URLLC Within mMTC. IEEE Access 2020, 8, 131796–131813. [CrossRef]
4. 5G White Paper Version 1.0. Available online: https://www.ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf (accessed on 10 June 2022).
5. Foukas, X.; Patounas, G.; Elmokashfi, A.; Marina, M.K. Network Slicing in 5G: Survey and Challenges. IEEE Commun. Mag. 2017, 55, 94–100. [CrossRef]
6. Rost, P.; Mannweiler, C.; Michalopoulos, D.S.; Sartori, C.; Sciancalepore, V.; Sastry, N.; Holland, O.; Tayade, S.; Han, B.; Bega, D.; et al. Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks. IEEE Commun. Mag. 2017, 55, 72–79. [CrossRef]
7. Yang, G.; Yu, B.-Y.; Jin, H.; Yoo, C. Libera for Programmable Network Virtualization. IEEE Commun. Mag. 2020, 58, 38–44. [CrossRef]
8. Yang, G.; Yoo, Y.; Kang, M.; Jin, H.; Yoo, C. Bandwidth Isolation Guarantee for SDN Virtual Networks. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10.
9. Afolabi, I.; Taleb, T.; Samdanis, K.; Ksentini, A.; Flinck, H. Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions. IEEE Commun. Surv. Tutor. 2018, 20, 2429–2453. [CrossRef]
10. Kreutz, D.; Ramos, F.M.V.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-Defined Networking: A Comprehensive Survey. Proc. IEEE 2015, 103, 14–76. [CrossRef]
11. Lara, A.; Kolasa, A.; Ramamurthy, B. Network Innovation using OpenFlow: A Survey. *IEEE Commun. Surv. Tutor.* 2014, 16, 493–512. [CrossRef]

12. Open Networking Foundation (ONF). Available online: https://www.opennetworking.org/ (accessed on 10 June 2022).

13. Barakabitze, A.A.; Ahmad, A.; Mijumbi, R.; Hines, A. 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Comput. Netw.* 2020, 167, 106984. [CrossRef]

14. McKeown, N.; Anderson, T.; Balakrishnan, H.; Peterson, L.; Rexford, J.; Shenker, S.; Turner, J. OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Comput. Commun. Rev.* 2008, 38, 69–74. [CrossRef]

15. Yeganeh, S.H.; Tootoonchian, A.; Ganjali, Y. On scalability of software-defined networking. *IEEE Commun. Mag.* 2013, 51, 136–141. [CrossRef]

16. Bianco, A.; Giaccone, P.; Mahmood, A.; Ullio, M.; Vercellone, V. Evaluating the SDN control traffic in large ISP networks. In Proceedings of the 2015 IEEE International Conference on Communications, London, UK, 8–12 June 2015; pp. 5248–5253.

17. Yu, B.-Y.; Yang, G.; Yoo, C. Comprehensive Prediction Models of Control Traffic for SDN Controllers. In Proceedings of the 2018 4th IEEE Conference on Network Softwarization and Workshops, Montreal, QC, Canada, 25–29 June 2018; pp. 262–266.

18. Bianchi, G.; Bonola, M.; Capone, A.; Cascone, C. OpenState. *ACM SIGCOMM Comput. Commun. Rev.* 2014, 44, 1100–1107. [CrossRef]

19. Bosshart, P.; Daly, D.; Gibb, G.; Izzard, M.; McKeown, N.; Rexford, J.; Schlesinger, C.; Talayco, D.; Vahdat, A.; Varghese, G.; et al. P4: Programming Protocol-independent Packet Processors. *ACM SIGCOMM Comput. Commun. Rev.* 2014, 44, 87–95. [CrossRef]

20. Song, H. Protocol-oblivious Forwarding: Unleash the Power of SDN through a Future-proof Forwarding Plane. In Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, Hong Kong, China, 16 August 2013; pp. 127–132.

21. Zhu, S.; Bi, J.; Sun, C.; Wu, C.; Hu, H. SDPA: Enhancing Stateful Forwarding for Software-Defined Networking. In Proceedings of the 2015 IEEE International Conference on Network Protocols, San Francisco, CA, USA, 10–13 November 2015; pp. 323–333.

22. Arashloo, M.T.; Koral, Y.; Greenberg, M.; Rexford, J.; Walker, D. SNAP: Stateful Network-Wide Abstractions for Packet Processing. In Proceedings of the 2016 ACM SIGCOMM Conference, Florianópolis, Brazil, 22–26 August 2016; pp. 29–43.

23. Tan, L.; Su, W.; Zhang, W.; Lv, J.; Zhang, Z.; Miao, J.; Liu, X.; Li, N. In-band Network Telemetry: A Survey. *Comput. Netw.* 2020, 186, 107763. [CrossRef]

24. Althumali, H.; Othman, M. A Survey of Random Access Control Techniques for Machine-to-Machine Communications in LTE/LTE-A Networks. *IEEE Access* 2018, 6, 74961–74983. [CrossRef]

25. Balevi, E.; Gitlin, R.D. A Random Access Scheme for Large Scale 5G/IoT Applications. In Proceedings of the IEEE 5G World Forum, Silicon Valley, CA, USA, 9–11 July 2018; pp. 452–456.

26. Chen, M.; Miao, Y.; Hao, Y.; Hwang, K. Narrow Band Internet of Things. *IEEE Access* 2017, 5, 20557–20577. [CrossRef]

27. Migabo, E.M.; Djouani, K.D.; Kurien, A.M. The Narrowband Internet of Things (NB-IoT) Resources Management Performance State of Art, Challenges, and Opportunities. *IEEE Access* 2020, 8, 97658–97675. [CrossRef]

28. Malik, H.; Sarmiento, J.L.R.; Alam, M.M.; Imran, M.A. Narrowband-Internet of Things (NB-IoT): Performance Evaluation in 5G Heterogeneous Wireless Networks. In Proceedings of the 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, Limassol, Cyprus, 11–13 September 2019; pp. 1–6.

29. 3GPP Release 13. Available online: https://www.3gpp.org/release-13 (accessed on 10 June 2022).

30. Sharma, S.K.; Wang, X. Toward Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions. *IEEE Commun. Surv. Tutor.* 2020, 22, 426–471. [CrossRef]

31. Magrin, D.; Pielli, C.; Stefanovič, Č.; Zorzi, M. Enabling LTE RACH Collision Multiplicity Detection via Machine Learning. In Proceedings of the 2019 International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, Aviron, France, 27–31 May 2019; pp. 1–8.

32. Modina, N.; Ferrari, R.; Magarini, M. A machine learning-based design of PRACH receiver in 5G. *Proc. Comput. Sci.* 2019, 151, 1100–1107. [CrossRef]

33. Tello-Oquendo, L.; Pacheco-Paramo, D.; Pla, V.; Martinez-Bauset, J. Reinforcement Learning-Based ACB in LTE-A Networks for Handling Massive M2M and H2H Communications. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–7. [CrossRef]

34. Garcia-Morales, J.; Lucas-Estan, M.C.; Gozalvez, J. Latency-Sensitive 5G RAN Slicing for Industry 4.0. *IEEE Access* 2019, 7, 143139–143159. [CrossRef]

35. Yang, P.; Xi, X.; Quek, T.Q.S.; Chen, J.; Cao, X.; Wu, D. RAN Slicing for Massive IoT and Bursty URLLC Service Multiplexing: Analysis and Optimization. *IEEE Internet Things J.* 2021, 8, 14258–14275. [CrossRef]

36. Popovski, P.; Trillingsgaard, K.F.; Simeone, O.; Durisi, G. 5G Wireless Network Slicing for eMBB, URLLC, and mMTC: A Communication-Theoretic View. *IEEE Access* 2018, 6, 55765–55779. [CrossRef]

37. Kokkinos, P.; Soumplis, P.; Kretsis, A.; Varvarigos, E.M. Network Slicing in 5G Infrastructures from the Edge to the Core. In Proceedings of the 12th IEEE/IFIP International Symposium on Communication Systems, Networks and Digital Signal Processing, Porto, Portugal, 20–22 July 2020; pp. 1–5. [CrossRef]

38. Chaabbar, M.; Diaz, G.; Dandoush, A.; Cerin, C.; Ghoumid, K. A Comprehensive Survey on the E2E 5G Network Slicing Model. *IEEE Trans. Netw. Serv. Manag.* 2021, 18, 49–62. [CrossRef]
39. Bockelmann, C.; Pratas, N.K.; Wunder, G.; Saur, S.; Navarro, M.; Gregoratti, D.; Vivier, G.; De Carvalho, E.; Ji, Y.; Stefanovic, C.; et al. Towards Massive Connectivity Support for Scalable mMTC Communications in 5G Networks. *IEEE Access* 2018, 6, 28969–28992. [CrossRef]

40. Ren, H.; Pan, C.; Deng, Y.; Elkashlan, M.; Nallanathan, A. Resource Allocation for URLLC in 5G Mission-Critical IoT Networks. In Proceedings of the 2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6. [CrossRef]

41. Varsier, N.; Dufrenè, L.-A.; Dumay, M.; Lampin, Q.; Schwoerer, J. A 5G New Radio for Balanced and Mixed IoT Use Cases: Challenges and Key Enablers in FR1 Band. *IEEE Commun. Mag.* 2021, 59, 82–87. [CrossRef]

42. Franchi, F.; Marotta, A.; Rinaldi, C.; Graziosi, F.; D’Errico, L. IoT-based Disaster Management System on 5G uRLLC Network. In Proceedings of the 2019 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM), Paris, France, 18–20 December 2019; pp. 1–4. [CrossRef]

43. Álvarez-Horcajo, J.; Martinez-Yelmo, I.; Lopez-Pajares, D.; Carral, J.A.; Savi, M. A Hybrid SDN Switch Based on Standard P4 Code. *IEEE Commun. Lett.* 2021, 25, 1482–1485. [CrossRef]

44. Gokarslan, K.; Sandal, Y.S.; Tugcu, T. Towards a URLLC-Aware Programmable Data Path with P4 for Industrial 5G Networks. In Proceedings of the 2021 IEEE International Conference on Communications Workshops, Montreal, QC, Canada, 14–23 June 2021; pp. 1–6. [CrossRef]

45. Paolucci, F.; Cugini, F.; Castoldi, P.; Osinski, T. Enhancing 5G SDN/NFV Edge with P4 Data Plane Programmability. *IEEE Netw.* 2021, 35, 154–160. [CrossRef]

46. In-Band Network Telemetry Specifications. Available online: https://p4.org/specs/ (accessed on 10 June 2022).

47. Mininet. Available online: https://github.com/mininet/mininet (accessed on 10 June 2022).

48. Ryu Controller. Available online: https://ryu-sdn.org/index.html (accessed on 10 June 2022).

49. Behavioral Model Version 2 (BMv2). Available online: https://github.com/p4lang/behavioral-model (accessed on 10 June 2022).

50. Iperf. Available online: https://iperf.fr (accessed on 10 June 2022).