Труды МАИ. 2024. № 136
Trudy MAI, 2024, no. 136

Научная статья
УДК 621.396
URL: https://trudymai.ru/published.php?ID=180677

ОСОБЕННОСТИ ОБРАБОТКИ СЕТЕВОГО ТРАФИКА НА КОММУТАТОРАХ ТРЕТЬЕГО УРОВНЯ С ОТКРЫТЫМИ ОПЕРАЦИОННЫМИ СИСТЕМАМИ

Игорь Геннадьевич Бужин[1]✉, Вероника Михайловна Антонова[2], Роман Владимирович Поздняков[3], Юрий Борисович Миронов[4]

[1,2,3,4]Московский технический университет связи и информатики, МТУСИ, Москва, Россия

[1]i.g.buzhin@mtuci.ru✉

[2]v.m.antonova@mtuci.ru

[3]p.v.pozdnyakov@mtuci.ru

[4]i.b.mironov@mtuci.ru

Аннотация. В современных системах управления и обработки информации авиационно-космических летательных аппаратов приобретает популярность использование в качестве сетевой инфраструктуры whitebox-коммутаторов с открытыми сетевыми операционными системами на базе сетевых чипов ASIC. Цель данной статьи является: выявление особенностей обработки сетевого трафика в whitebox-коммутаторах и разработка общих рекомендаций для их устранения. Результаты: Выявлены особенности и проблемные места при обработке сетевого трафика с помощью whitebox-коммутаторов третьего уровня на базе ASIC, а также

предложены общие рекомендации для предотвращения сбоев и ошибок при обработке трафика.

***Ключевые слова*:** whitebox-коммутатор, ASIC, Kernel Routing, Сетевая Операционная Система (СОС), Сетевой трафик

Для цитирования: Бужин И.Г., Антонова В.М., Поздняков Р.В., Миронов Ю.Б. Особенности обработки сетевого трафика на коммутаторах третьего уровня с открытыми операционными системами // Труды МАИ. 2024. № 136. URL: https://trudymai.ru/published.php?ID=180677

Original article

FEATURES OF NETWORK TRAFFIC PROCESSING ON THIRD-LEVEL SWITCHES WITH OPEN OPERATING SYSTEMS

Igor G. Buzhin[1]✉, Veronika M. Antonova[2],
Roman V. Pozdnyakov[3], Yuri B. Mironov[4]
[1,2,3,4]Moscow technical university of communications and informatics,
Moscow, Russia
[1]i.g.buzhin@mtuci.ru✉
[2]v.m.antonova@mtuci.ru
[3]p.v.pozdnyakov@mtuci.ru
[4]i.b.mironov@mtuci.ru

Abstract In modern control and information processing systems of aerospace aircraft, the use of whitebox switches with open network operating systems based on ASIC network chips as a network infrastructure is gaining popularity, thanks to which various network

functions become available, independence from network equipment manufacturers appears. At the same time, the functioning of such devices and the principle of processing network traffic have their own characteristics that should be taken into account when operating white ABTOPEbox switches. The purpose of the study: to identify the features of network traffic processing in whitebox switches and to develop some recommendations for their elimination.

As a result, this article analyzes the hardware and software architecture of network equipment, as a result of which it is revealed that the architecture of a network device consists of a data transmission layer (Data plane) and a control plane. The Data plane layer is represented by an ASIC network chip with SDK and SAI tools. The Control plane level is represented by a CPU chip with a Kernel Routing (Linux) tool and a CLI. Also, as a result of the operation of whitebox switches, such traffic processing features as the desynchronization of Kernel Routing and ASIC, a violation of interface configuration logic and a decrease in port throughput were revealed. To eliminate these shortcomings, some recommendations have been developed, according to which, in order to successfully process traffic in whitebox switches, it is necessary to control the interface configuration on ASIC ingress pipelines, check the synchronization of Kernel routing Linux and ASIC. To maintain the required packet throughput, it is necessary to monitor the correctness of specifying egress interfaces and next-hop when processing traffic from input ports.

Keywords: future generation mobile networks, load balancing, network layers, network connectivity, virtualised infrastructure, data networks

For citation: Buzhin I.G., Antonova V.M., Pozdnyakov R.V., Mironov Yu.B. Features of network traffic processing on third-level switches with open operating systems. *Trudy MAI*, 2024, no.136. URL: https://trudymai.ru/published.php?ID=180677

Введение

В настоящее время сетевая инфраструктура может строится на базе whitebox-коммутаторов с открытыми сетевыми операционными системами. Whitebox-коммутатор представляет собой открытое сетевое устройство с разделением программного и аппаратного обеспечения. Такие устройства работают под управлением сетевых операционных систем (СОС) - набор программных компонентов для управления коммутаторами и другими сетевыми устройствами с широкой функциональностью на базе Linux - дистрибутивов. СОС может работать на различных аппаратных платформах посредством спецификации Switch Abstraction Interface (SAI), которая позволяет одним и тем же программным компонентам СОС работать на различных аппаратных устройствах. SAI поддерживает туннелирование, управление L3-маршрутизацией, настройку QoS (Quality of Service) и т.п. Код SAI открыт (написан на C).

Данный подход позволяет избавится зависимости от производителей оборудования, самостоятельно реализовывать специфические сетевые функции, а также повышать эффективность обработки сетевого трафика [1, 2]. Однако, при реализации сетевого оборудования на базе whitebox-коммутаторов и открытых СОС могут возникать ошибки и снижение производительности сетевого оборудования при обработки сетевого трафика. Таким образом, необходимо выявить возможные

«узкие» места при реализации сетевого оборудования на базе whitebox-коммутаторов и открытых СОС для задач обработки и передачи сетевого трафика.

Архитектура сетевого оборудования

Упрощенная аппаратная архитектура сетевого оборудования изображена на рис. 1.

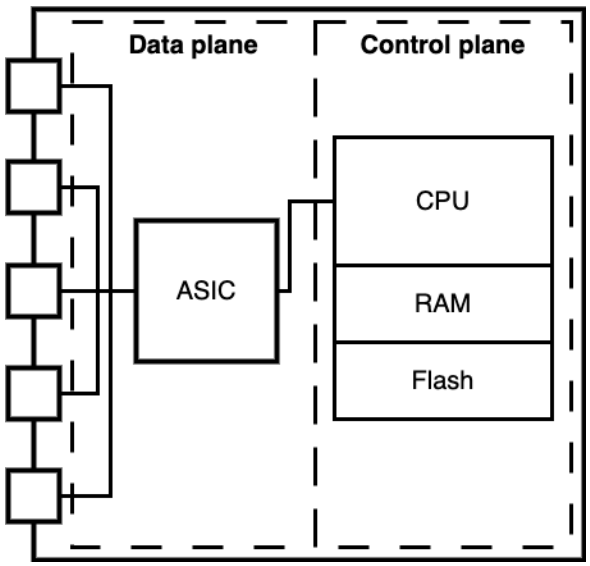

Рисунок 1 - Упрощенная аппаратная архитектура сетевого оборудования

Архитектура сетевого устройства состоит из 2х логических уровней – уровень передачи данных (Data plane) и уровень управления (Control plane). Основными задачами Data plane является быстрая обработка большого количества сетевого трафика, надежность и простота логики на основе таблиц, разбор заголовков различных уровней, выбор выходного порта, определение выходного набора заголовков, буферизация. В основе Data plane, как правило, лежит ASIC (Application Specific Integrated Circuit) [3, 4] – интегральная микросхема специального назначения с предопределённым набором функций Data plane, которые выполняются аппаратно. Control plane отвечает за логику работы сетевого устройства для обеспечения в дальнейшем возможности передачи пакетов

(заполнение различных таблиц, например, маршрутизации, отработку различных служебных протоколов ARP/STP/и пр.), управляет ASIC, обрабатывает только служебный трафик и имеет сложную логику. Control plane [5] основан на традиционных CPU+RAM+Flash, на которых реализуются алгоритмы со сложной логикой для реализации различных сетевых сервисов.

Для управления аппаратной частью и предоставления пользовательского интерфейса устанавливают сетевую операционную систему (COC или NOS) [6]. Для whitebox-коммутаторов доступны открытые COC на основе Linux. COC должна иметь полных набор сетевых функций для применения на различных сетях связи, в том числе и центрах обработки данных. За реализацию различных сетевых функций отвечают network demons. После установки COC программно-аппаратная архитектура сетевого устройства будет иметь вид, показанный на рис. 2.

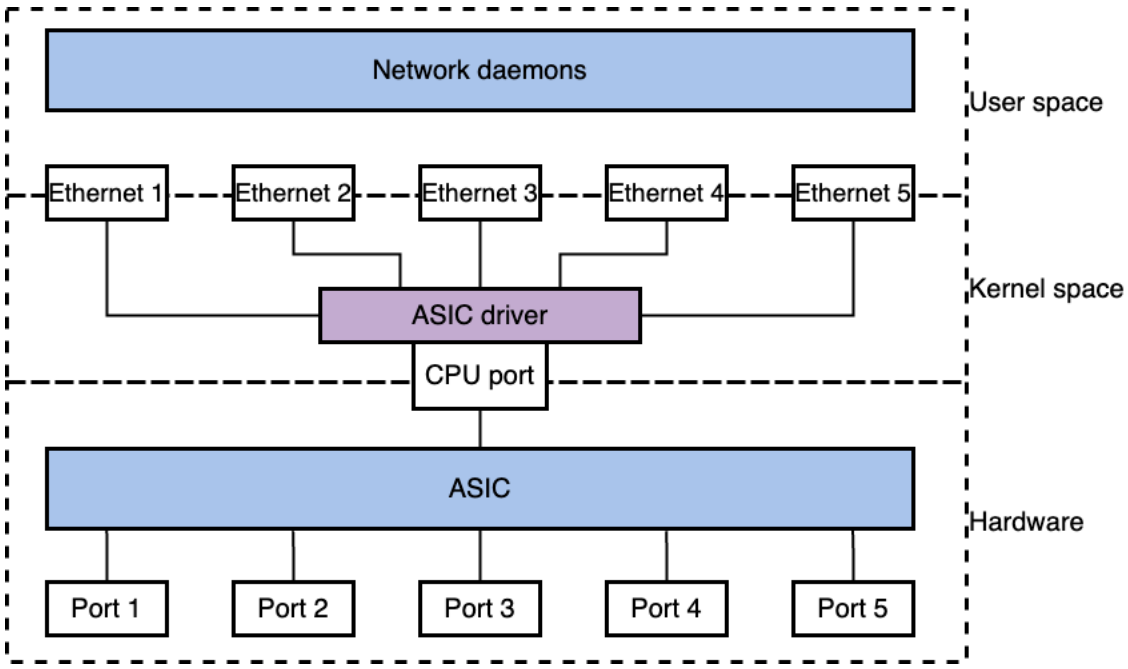

Рисунок 2 - Упрощенная программно-аппаратная архитектура сетевого оборудования

Особенности эксплуатации whitebox-коммутаторов третьего уровня в инфокоммуникационных сетях

Сетевой трафик приходит на физические порты, которые обрабатываются ASIC. Далее, весь трафик поступает на CPU порт с предварительно присвоенной меткой от производителя ASIC, в которой указано с какого физического порта принят данный пакет. Control plane представлена СОС с драйвером ASIC, которая создает интерфейсы (netlink) и по тегу с Hardware уровня распределяет трафик по сетевым интерфейсам. Также в СОС есть сетевые демоны (процессы СОС, которые работают в фоновом режиме без прямого участия пользователя), которые обрабатывают служебный трафик (OSPF, BGP и т.д.) и взаимодействуют с физическими портами через интерфейсы для реализации конкретных сетевых сервисов.

СОС логически состоит их двух больших областей: Linux-подобная ОС и ПО ASIC. В Linux есть система netlink, которая управляется через библиотеку libnl [7]. ПО ASIC управляется через SDK (поставляется производителем ASIC) [8, 9]. В связи с особенностями SDK каждого производителя для предоставления унифицированного API для ASIC используют SAI, структура которого представлена на рис. 3.

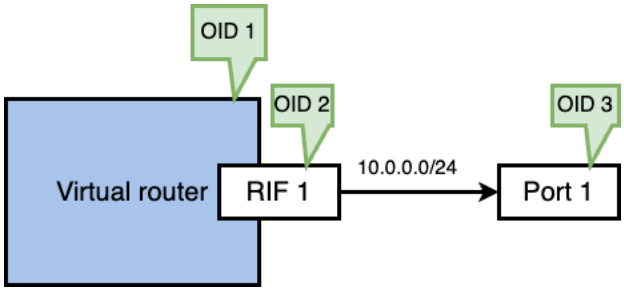

Рисунок 3 - Структурная схема SAI

SAI представляет собой набор команд (ключ-значение). Внутри ASIC есть блок Virtual router, который отвечает за весь L3 функционал. Разработчик с помощью команд SAI создает L3 интерфейс в Virtual router. Для это необходимо создать маршрут в сеть 10.0.0.0 которая будет связывать с физическим портом. Блоки SAI идентифицируются OID (Object ID).

Разберем процесс создания L3 интерфейса. Администратор в командной строке (CLI) переходит в конфигурационный режим и присваивает IP-адрес интерфейсу. На уровне Linux появился интерфейс с MAC-адресом и direct-маршрутом. При использовании whitebox-коммутаторов важно следить за полной синхронизацией конфигураций ASIC и Linux – они должны быть идентичными. В SAI в virtual router создается интерфейс с двумя маршрутами – один для самого коммутатора (на CPU) с маской 32 и второй закрепляется за физическим портом коммутатора. SDK ASIC создает L3 интерфейс в VLAN 4095. Такой VLAN применяется с целью предотвращения пересечения трафика из других VLAN. VLAN 4095 является служебным для создания L3 интерфейсов [10]. Далее SDK создает выходной (egress) интерфейс из порта и маршрут для этого выхода. Так как по умолчанию у коммутатора не будет информации о next-hop коммутаторе, то он создает маршрут на CPU порт. Таким образом, вся схема движения трафика на уровне ASIC представлена на рис. 4.

Ingress конвейер порта осуществляет парсинг (сбор по заданным параметрам) информации из заголовков пакетов и их сопоставление с информацией Filed Processor [11], FIB (таблица для ускоренной пересылки пакетов) [12], ACL (списки доступа) [13] и т.д., которая хранится в CAM или TCAM таблицах.

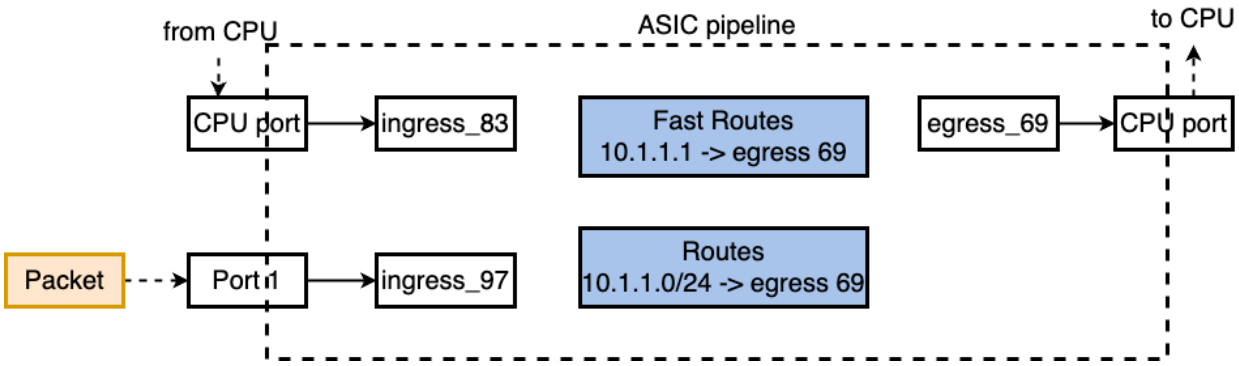

Рисунок 4 - Схема движения трафика на уровне ASIC

Например, в блоке Filed Processor (таблица записей для сопоставления служебных протоколов, в которой чтение происходит последовательно) могут быть записи о протоколах ARP [14] и ICMP [15]. По логике работы TCP/IP, поле с информацией ARP должно быть выше, чем поле с информацией ICMP и какой-либо другой информации. Но оказывается, что в стандартной реализации ASIC большинства whitebox-коммутаторов первым правилом в блоке Filed Processor идет правило, согласно которому broadcast трафик должен блокироваться. Таким образом, в блоке Filed Processor ASIC информация из заголовка пакета сначала сопоставляется с правилом о запрете broadcast трафика. Если на этот порт приходит запрос ping, то сопоставление с правилом ICMP происходить не будет, т.к. этот запрос будет заблокирован правилом выше. Для устранения данной проблемы необходимо настроить приоритет протоколов ARP и trap через Control Plane Policing [16]:

switch1# show running-configuration

!

copp-action queue4_group2

set trap-action copy

set trap-queue 4

set trap-priority 4

…

!

class arp priority 0

set copp-action queue4_group2

После устранения данной проблемы коммутатор сможет принимать ARP пакеты. После этого в Kernel routing появляется информация о next hop (информацию о следующем порту в маршруте и его MAC адрес). Теперь нужно синхронизировать информацию из Kernel routing Linux с ASIC. Для этого через SAI необходимо добавить соседа (neighbor). После чего SDK ASIC создает маршрут до нужного выходного порта. Тогда полная схема отправки ARP-ответа будет выглядеть следующим образом:

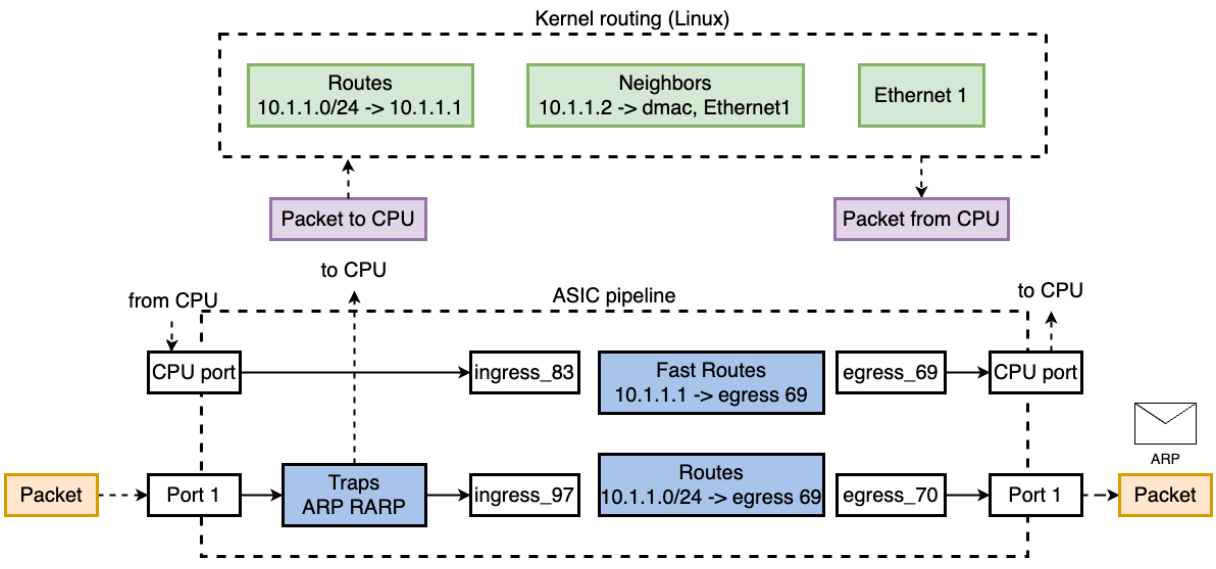

Рисунок 5 - Схема отправки ARP-ответа

Теперь через CLI можно создать маршрут командой ip route. После проверки (show ip route) должен появиться введенный маршрут в нужную сеть. SAI создает

маршрут до next hop нужного порта и SDK добавляет маршрут через порт, созданный ранее. После этого ping между коммутаторами должен заработать.

Спустя время могут появиться сбои. Часть ping запросов (не все запросы) могут быть потеряны. Это может быть связаны со временем неактивной работы neighbor. Трафик какое-то время не передавался через данный neighbor и запись ip neighbor [17] пометилась как неактивная (STALE). Таким образом, в ASIC ушла команда на удаление egress интерфейса. Для предотвращения такой ситуации рекомендуется периодически отравлять ARP запросы до next-hop (ARP-запросы по таймеру до 90 секунд).

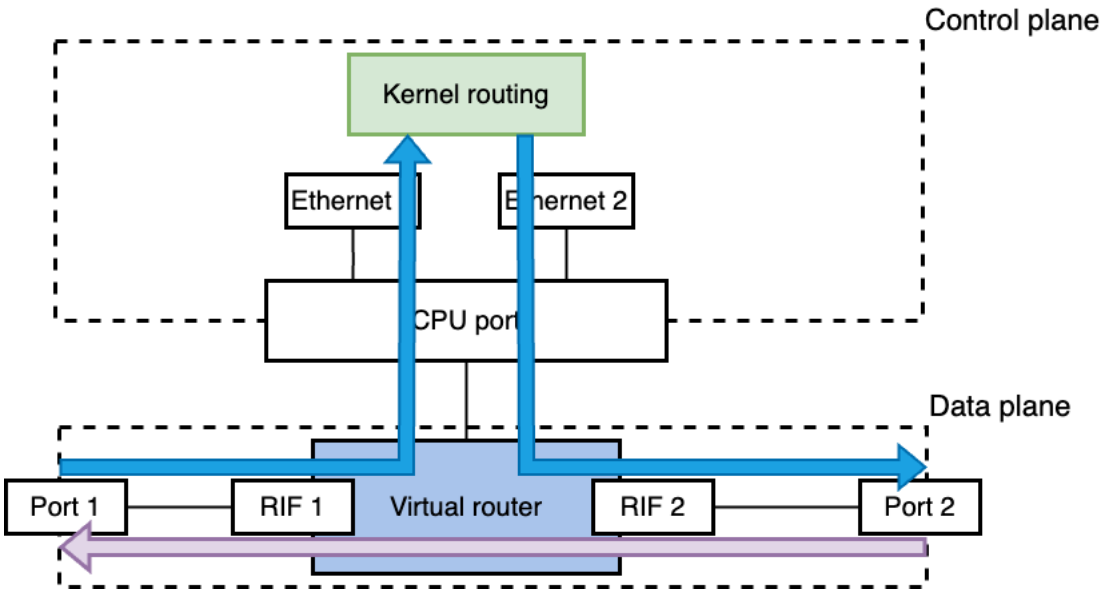

Рисунок 6 - Схема движения через CPU порт и в обход CPU порта

Также в whitebox-коммутаторах может наблюдаться такая проблема, как понижение скорости трафика, например, на 25G порту до 10G. Такая ситуация может возникнуть, если будет указан неверный маршрут до egress интерфейса. Трафик, придя на порт, может отправляться в Linux через CPU порт (рис. 6), который ограничен по пропускной способности. Таким образом, при ошибочном

указании egress интерфейса на тот, который ведет в CPU порт, может возникнуть ограничение по пропускной способности трафика. При отладке коммутатора необходимо следить за правильностью всех next-hop и их жизнеспособностью.

Заключение

В данной статье проведен анализ аппаратной и программно-аппаратной архитектуры сетевого оборудования, в результате которого выявлено, что архитектура сетевого устройства состоит уровня передачи данных (Data plane) и уровня управления (Control plane). Уровень Data plane представлен сетевым чипом ASIC с инструментами SDK и SAI. Уровень Control plane представлен чипом CPU с инструментом Kernel Routing (Linux) и CLI. Также в результате эксплуатации whitebox-коммутаторов были выявлены такие особенности обработки трафика, как рассинхронизация Kernel Routing и ASIC, нарушение логики настройки интерфейсов и понижение пропускной способности портов. Для устранения указанных недостатков разработаны методические рекомендации, согласно которым для успешного обработки трафика в whitebox-коммутаторах необходимо контролировать настройку интерфейса на ingress конвейерах ASIC, проверять синхронизацию Kernel routing Linux и ASIC. Для подержания необходимой пропускной способности пакетов необходимо контролировать правильность указания egress интерфейсов и next-hop при обработке трафика с входных портов.

Список источников

1. Tajammal M.B., Durad H., Iqbal R.N. Secure Switch Development using Open Network Linux on Bare Metal Switch, Development of Network Operating System for Bare Metal Switches (Nos) // VFAST Transactions on Software Engineering, 2020, vol. 8, no. 1, pp. 43-54. DOI: 10.21015/vtse.v8i1.575

2. Gupta K. et al. ASIC: Aligning sparse in-the-wild image collections // Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 4134-4145. DOI: 10.1109/ICCV51070.2023.00382

3. Saquetti M. et al. A Terabit Hybrid FPGA-ASIC Platform for Switch Virtualization // 2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), IEEE, 2021, pp. 73-78. DOI: 10.1109/ISVLSI51109.2021.00024

4. Wang S. et al. Making multi-string pattern matching scalable and cost-efficient with programmable switching asics // IEEE INFOCOM 2021-IEEE Conference on Computer Communications, IEEE, 2021, pp. 1-10. DOI: 10.1109/INFOCOM42981.2021.9488796

5. Fang J. et al. TB-TBP: a task-based adaptive routing algorithm for network-on-chip in heterogenous CPU-GPU architectures // The Journal of Supercomputing, 2024, vol. 80, no. 5, pp. 6311-6335. DOI: 10.1007/s11227-023-05700-7

6. Blöcher M. et al. Switches for HIRE: Resource scheduling for data center in-network computing // Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, 2021, pp. 268-285. DOI: 10.1145/3445814.3446760

7. Von Arnim C. et al. Updating the Linux TAPRIO Scheduler in Deterministic Time // 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), IEEE, 2022, pp. 1-7. DOI: 10.1109/ETFA52439.2022.9921594

8. Ejjeh A., Adve V., Rutenbar R. Studying the potential of automatic optimizations in the Intel FPGA SDK for OpenCL // arXiv preprint arXiv:2201.03558, 2022.

9. Deierling K. NVIDIA's Resource Transmutable Network Processing ASIC // 2023 IEEE Hot Chips 35 Symposium (HCS), IEEE Computer Society, 2023. pp. 1-14. DOI: 10.1109/HCS59251.2023.10254697

10. Gentile A.F., Fazio P., Miceli G. A Survey on the Implementation and Management of Secure Virtual Private Networks (VPNs) and Virtual LANs (VLANs) in Static and Mobile Scenarios // Telecom–MDPI, 2021, vol. 2, no. 4, pp. 430-445. DOI: 10.3390/telecom2040025

11. Suresh P. et al. Field-programmable gate arrays in a low power vision system // Computers & Electrical Engineering, 2021, vol. 90, pp. 106996. DOI: 10.1016/j.compeleceng.2021.106996

12. Rosa E.C., de Oliveira Silva F. A review on recent NDN FIB implementations for high-speed switches // International Conference On Advanced Information Networking and Applications. Cham: Springer International Publishing, 2022, vol. 3, pp. 288-300. DOI: 10.1007/978-3-030-99619-2_28

13. Amin R. et al. Auto-configuration of ACL policy in case of topology change in hybrid SDN // IEEE Access, 2016, vol. 4, pp. 9437-9450. DOI: 10.1109/ACCESS.2016.2641482

14. Bruschi D., Ornaghi A., Rosti E. S-ARP: a secure address resolution protocol // 19th Annual Computer Security Applications Conference, Proceedings IEEE, 2003, pp. 66-74. DOI: 10.1109/CSAC.2003.1254311

15. Huang J. et al. Adjusting packet size to mitigate TCP incast in data center networks with COTS switches // IEEE Transactions on Cloud Computing, 2018, vol. 8, no. 3, pp. 749-763. DOI: 10.1109/TCC.2018.2810870

16. Đerić N. et al. Towards Understanding the Performance of Traffic Policing in Programmable Hardware Switches // 2021 IEEE 7th International Conference on Network Softwarization (NetSoft), IEEE, 2021, pp. 70-78. DOI: 10.1109/NetSoft51509.2021.9492560

17. Ning B. et al. Collective behaviors of mobile robots beyond the nearest neighbor rules with switching topology // IEEE transactions on cybernetics, 2017, vol. 48, no. 5, pp. 1577-1590. DOI: 10.1109/TCYB.2017.2708321

18. Wu D. et al. Accelerated service chaining on a single switch ASIC // Proceedings of the 18th ACM Workshop on Hot Topics in Networks, 2019, pp. 141-149. DOI: 10.1145/3365609.3365849

19. Monika B.K., Amaresha S.K., Yellampalli S.S. ASIC implementation of switch architecture used in NoC // 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), IEEE, 2017, pp. 758-761. DOI: 10.1109/SmartTechCon.2017.8358473

20. Piasetzky Y. et al. Switch asic programmability in hybrid mode // 2018 IEEE 26th International Conference on Network Protocols (ICNP), IEEE, 2018, pp. 448-449. DOI: 10.1109/ICNP.2018.00067

21. Zhang L.L. et al. A scheduler ASIC for a programmable packet switch // IEEE Micro, 2000, vol. 20, no. 1. pp. 42-48. DOI: 10.1109/40.820052

22. Муратчаев С.С., Волков А.С., Маргарян Р.А., Бахтин А.А. Разработка адаптивной версии протокола маршрутизации OLSRv2 в сетях MANET // Труды МАИ. 2022. № 123. URL: https://trudymai.ru/published.php?ID=165556. DOI: 10.34759/trd-2022-123-13

23. Бахтин А.А., Волков А.С., Солодков А.В., Свиридов И.А. Система распознавания модуляции сигналов на основе нейронной сети с использованием ПЛИС // Труды МАИ. 2021. № 121. URL: https://trudymai.ru/published.php?ID=162660. DOI: 10.34759/trd-2021-121-13

References

1. Tajammal M.B., Durad H., Iqbal R.N. Secure Switch Development using Open Network Linux on Bare Metal Switch, Development of Network Operating System for Bare Metal Switches (Nos), *VFAST Transactions on Software Engineering*, 2020, vol. 8, no. 1, pp. 43-54. DOI: 10.21015/vtse.v8i1.575

2. Gupta K. et al. ASIC: Aligning sparse in-the-wild image collections, *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4134-4145. DOI: 10.1109/ICCV51070.2023.00382

3. Saquetti M. et al. A Terabit Hybrid FPGA-ASIC Platform for Switch Virtualization, *2021 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, IEEE, 2021, pp. 73-78. DOI: 10.1109/ISVLSI51109.2021.00024

4. Wang S. et al. Making multi-string pattern matching scalable and cost-efficient with programmable switching asics, *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*, IEEE, 2021, pp. 1-10. DOI: 10.1109/INFOCOM42981.2021.9488796

5. Fang J. et al. TB-TBP: a task-based adaptive routing algorithm for network-on-chip in heterogenous CPU-GPU architectures, *The Journal of Supercomputing*, 2024, vol. 80, no. 5, pp. 6311-6335. DOI: 10.1007/s11227-023-05700-7

6. Blöcher M. et al. Switches for HIRE: Resource scheduling for data center in-network computing, *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2021, pp. 268-285. DOI: 10.1145/3445814.3446760

7. Von Arnim C. et al. Updating the Linux TAPRIO Scheduler in Deterministic Time, *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2022, pp. 1-7. DOI: 10.1109/ETFA52439.2022.9921594

8. Ejjeh A., Adve V., Rutenbar R. Studying the potential of automatic optimizations in the Intel FPGA SDK for OpenCL, *arXiv preprint arXiv:2201.03558*, 2022.

9. Deierling K. NVIDIA's Resource Transmutable Network Processing ASIC, *2023 IEEE Hot Chips 35 Symposium (HCS), IEEE Computer Society*, 2023. pp. 1-14. DOI: 10.1109/HCS59251.2023.10254697

10. Gentile A.F., Fazio P., Miceli G. A Survey on the Implementation and Management of Secure Virtual Private Networks (VPNs) and Virtual LANs (VLANs) in Static and Mobile Scenarios, *Telecom–MDPI*, 2021, vol. 2, no. 4, pp. 430-445. DOI: 10.3390/telecom2040025

11. Suresh P. et al. Field-programmable gate arrays in a low power vision system, *Computers & Electrical Engineering*, 2021, vol. 90, pp. 106996. DOI: 10.1016/j.compeleceng.2021.106996

12. Rosa E.C., de Oliveira Silva F. A review on recent NDN FIB implementations for high-speed switches, *International Conference On Advanced Information Networking and Applications*. Cham: Springer International Publishing, 2022, vol. 3, pp. 288-300. DOI: 10.1007/978-3-030-99619-2_28

13. Amin R. et al. Auto-configuration of ACL policy in case of topology change in hybrid SDN, *IEEE Access*, 2016, vol. 4, pp. 9437-9450. DOI: 10.1109/ACCESS.2016.2641482

14. Bruschi D., Ornaghi A., Rosti E. S-ARP: a secure address resolution protocol, *19th Annual Computer Security Applications Conference*, Proceedings IEEE, 2003, pp. 66-74. DOI: 10.1109/CSAC.2003.1254311

15. Huang J. et al. Adjusting packet size to mitigate TCP incast in data center networks with COTS switches, *IEEE Transactions on Cloud Computing*, 2018, vol. 8, no. 3, pp. 749-763. DOI: 10.1109/TCC.2018.2810870

16. Đerić N. et al. Towards Understanding the Performance of Traffic Policing in Programmable Hardware Switches, *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*, IEEE, 2021, pp. 70-78. DOI: 10.1109/NetSoft51509.2021.9492560

17. Ning B. et al. Collective behaviors of mobile robots beyond the nearest neighbor rules with switching topology, *IEEE transactions on cybernetics*, 2017, vol. 48, no. 5, pp. 1577-1590. DOI: 10.1109/TCYB.2017.2708321

18. Wu D. et al. Accelerated service chaining on a single switch ASIC, *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 2019, pp. 141-149. DOI: 10.1145/3365609.3365849

19. Monika B.K., Amaresha S.K., Yellampalli S.S. ASIC implementation of switch architecture used in NoC, *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, IEEE, 2017, pp. 758-761. DOI: 10.1109/SmartTechCon.2017.8358473

20. Piasetzky Y. et al. Switch asic programmability in hybrid mode, *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, IEEE, 2018, pp. 448-449. DOI: 10.1109/ICNP.2018.00067

21. Zhang L.L. et al. A scheduler ASIC for a programmable packet switch, *IEEE Micro*, 2000, vol. 20, no. 1. pp. 42-48. DOI: 10.1109/40.820052

22. Muratchaev S.S., Volkov A.S., Margaryan R.A., Bakhtin A.A. *Trudy MAI*, 2022, no. 123. URL: https://trudymai.ru/eng/published.php?ID=165556. DOI: 10.34759/trd-2022-123-13

23. Bakhtin A.A., Volkov A.S., Solodkov A.V., Sviridov I.A. *Trudy MAI*, 2021, no. 121. URL: https://trudymai.ru/eng/published.php?ID=162660. DOI: 10.34759/trd-2021-121-13

Статья поступила в редакцию 06.05.2024
Одобрена после рецензирования 15.05.2024
Принята к публикации 27.06.2024
The article was submitted on 06.05.2024; approved after reviewing on 15.05.2024; accepted for publication on 27.06.2024