Research Statement

Wenfei Wu (wenfeiwu@tsinghua.edu.cn)

1 Introduction

My ultimate research goal is to make computer networks be the basic infrastructure like water and electricity for humans to access knowledge without any barriers. It still takes a long way for the society to achieve this goal, and that is why I devote to the research of networked systems.

In the research of networked systems, I prefer to building *full-stack* solutions for requirements in production networks — first inventing new primitives in the data plane and then rebuilding the mechanisms in the control plane and management plane. Following this philosophy, I made contributions in the areas of machine learning infrastructure, network function DevOps, and virtual network diagnosis.

2 Multi-Tenant Distributed Machine Learning System

This series of research started from 2018 until now.

With the increasing dataset size and the computation workload, machine learning systems evolve to be distributed (Distributed Machine Learning, i.e., DML). Most existing DML systems (e.g., PyTorch, TensorFlow) view the underlying network as a black box providing uniform and sufficient bandwidth, which, however, is not always true. In practical deployment, DML systems do not always have a dedicated cluster, and they could probably share the infrastructure. In this case, DML jobs would face the following challenges: (1) existing DML jobs are suffering from the insufficient bandwidth, (2) existing networks need new high-performant bandwidth isolation mechanisms for multi-tenant DML jobs; (3) a control-plane job placement mechanism is needed for DML jobs to efficiently share the infrastructure.

2.1 In-network Aggregation Transmission Protocol

In a typical DML architecture worker-parameter-server (worker-PS), there exists a traffic incast problem — multiple workers exchanging data with a single PS, which overwhelms the PS's access link. Meanwhile, a recent progress in programmable switches provides the opportunity to offload the aggregation operation from the PS to switches, and this would potentially reduce the traffic volume at the bottleneck link.

With this intuition, we build a new data-plane transport layer protocol (Aggregation Transmission Protocol, i.e., ATP) for multi-tenant learning. ATP consists of the networking stack on end hosts and the aggregation service on switches. It provides three features: (1) a distributed resource sharing mechanism on the switch, which allows multiple jobs benefits from the aggregation service, (2) a transmission protocol for stream aggregation, which provides reliability (for accuracy), congestion control (for contention), and fallback (for cases beyond switch's capability), and (3) engineering optimization to boost the throughput. This work is published in NSDI'21. The prototype and source code is currently anonymously released to the public. [11]

2.2 High-Performance Rate Limiters for Job Traffic Isolation

When multiple jobs share the network, there are usually policies to describe their sharing quota of the network bandwidth, which are enforced by data-plane rate limiters. We first design a rate limiter for traditional networks. The rate limiter has the features of accuracy and low latency, which are crucial for high-performance computation and service. It applies Explicit Congestion Notification (ECN) in the rate-limiting algorithm and interacts with the end-host TCP stack, which reduces queuing delay at the rate limiter. This work is accepted by APNet'17. [7]

The second performance isolation design is a rate limiter for programmable networks. It has the advantages of extremely scalable (1X million instances on a single switch), functionality, and manageability. This work refines the memory usage of the leaky bucket algorithm and implements a rate limiter with 6 bytes; thus, it can be scalable on a single switch. We show its feasibility for production network applications. This work is published in INFOCOM'21.[8]

2.3 Multi-tenant DML Job Management

With the data-plane primitives ready (ATP and rate limiters), we are proceeding with the job placement problem. The main goal is (1) to build a controller which takes the network topology and job placement task as input, generates per-device configuration rules, and installs the rules to devices, and (2) design the best job placement algorithm, with which multiple jobs are placed into infrastructure and the resource utilization (bandwidth, GPU, memory, etc.) is maximized. This work is ongoing now.

3 DevOps for Network Functions

This series of research started from 2016 until now.

Network Functions (NF) are the set of appliances (e.g., firewall, cache) deployed in networks which improve the network performance and security. The trend of implementing NFs in software is called Network Function Virtualization (NFV), which eases the network management. In the current NFV ecosystem, NFs are provisioned by vendors like Cisco, Juniper, and Huawei; and NFs are operated by network operators like Azure, Aliyun, and campus network operators. However, *the views of NF developers and NF operators diverge, causing an intrinsic contradiction in the NF deployment*. NF developers view (and deliver) the NFs as monolithic pieces of software, without specifying their internal behavior logic; while NF operators usually need NF behavior models to manage NFs (e.g., network verification). With such an intrinsic contradiction, there is no confidence that the NF behavior model in NF operation can represent the actual behavior of the NF code in the ordinary operation; in serious cases, the misrepresented NF models could lead NF to be misconfigured, causing runtime errors and network outage. To solve this problem, I propose a layering methodology to bridge both groups — designing a standard abstraction layer to decouple and enhance both the NF development (in the data plane) and NF operation (in the control plane).

3.1 Design Abstractions for NF Behavior Models

The first step is to design a domain-specific language (DSL) to define NF behaviors formally. We achieve this goal by summarizing NF behaviors from a set of collected commercial NF programs. In one black-box approach, we set up an NF as a running instance, inject packet traces, and observes its output. We apply L* algorithm to the input/output traces and extract a finite automaton (FA). This work was accepted by NSDI'19.[14]

Meanwhile, we also take a white-box approach. We take the NF program code and apply compiler techniques to extract execution paths, and formalize an execution path as a conditional state transition. This work also validates that most NFs can be described as a FA. This work was accepted by HotNets'16.[22]

3.2 Cross-platform Development using NF Behavior Models

With the empirical measurement, we define an NF DSL and use the DSL to program typical NF models. Then we build an NF compiler which translates NF models to NF executables. The compiler framework design refers to that of LLVM, which allows network operators to customize NF models according to the runtime environment. The framework is named NFD, and it has 14 use-case NFs and supports 6 runtime environments (Linux, OpenNetVM, DPDK, GPU, SGX, OpenNF). This work is published in INFOCOM'21.[9]

One use case is that we design an NF that can perform network coding. The NF logic is a combination of optimistic network coding and reliable transmission mechanism, and the environmental integration includes DPDK and shared memory. It achieves high bandwidth saturation and reliability in a high-throughput but lossy network (targeting 5G networks). This work was accepted by IPCCC'19 and won the best paper runner-up.[12]

3.3 NF Verification using NF Abstractions

With the NF DSL abstraction in the data plane ready, we further design NF verification tools in the management plane. Its significance is that such a tool helps the NF operator to validate the correctness of NF configuration, avoiding runtime error. We implement a symbolic execution engine (SEE) for NFD: given a configured NF, the SEE outputs all execution paths of an NF model. Compared with existing solutions (e.g., VMN, BUZZ), our SEE supports the verification of time-driven logic. This work was accepted by MASCOT'20. [17]

3.4 Joint Optimization for NF Program with Runtime Configurations

Finally, we synthesize NF development and deployment and propose NF optimization solutions for NF DevOps. Assume the runtime network-wide configuration is given, the NF code can be statically injected with the configuration, then compiler optimizations (e.g., constant folding, dead code elimination) can be applied. We collect practical network configurations and NFs developed by the DSL, and customize the program optimization techniques to show the performance gain. This work is accepted by SOSR'20, and its long version accepted by INFOCOM'21.[3]

In other scenarios where the NF code is from different parties and cannot be consolidated, we design interfaces to instrument individual NFs. The interfaces can collect NF behaviors in the runtime and merge the logic on a new fast data path, so that the performance is improved. This work was published in ICDCS'19.[10]

4 Network Diagnostic Solutions

This series of research was done during my Ph.D., from 2010 to 2015.

Modern computer networks evolve to be a more complicated system, including more parties in operation and more hardware/software components than ever before. This trend complicates the network troubleshooting. For example, when a cloud tenant observes virtual network failures, the tenant needs to interact with the cloud operator, and the operator needs to look into more complicated network appliances (introduced by virtualization) for root cause detection. Targeting such new difficulties in network management, I conduct the following research: (1) design network monitoring primitives in the data plane, (2) design network diagnostic language and applications in the control plane, and (3) analyze the relationship between operation and failures using data science techniques.

4.1 Performance Diagnosis for Software Data Plane

In modern virtualized networks, a large variety of software components (e.g., virtual switches, virtualized NFs) form the software data plane. And they also introduce performance issues to the end-to-end network performance. I made an analysis of the data path that packet traverses, model the data plane as basic elements, and inject performance counters (e.g., bytes, packets, I/O time) into the elements. Using the new monitoring primitives, I quantitatively define metrics for performance issues: bottleneck, contention, and bugs. Using the data-plane monitors, the metrics is computed to identify the root cause in the software data plane. This work was published in IMC'15.[19]

4.2 Virtual Network Diagnostic Services

In modern public clouds, tenants configure their virtual network via the control plane. While their applications are migrated to the cloud, their ability to diagnose the network is not. The virtualization prevents them from accessing the virtual appliances (e.g., firewall, virtual routers). Thus, I built a ground-up service for cloud tenants, namely Virtual Network Diagnostic Service (VND). VND has interfaces for tenants to specify the suspected virtual appliances to diagnose, then VND uses the data plane monitor to dump information into a database; VND then provides SQL interfaces for tenants to program diagnostic applications, which would be executed on the database. VND builds such groud-up services and provides several example diagnostic applications (e.g., RTT, loss, bottleneck detection). This work was published in SoCC'13 and won the best student paper award.[21]

4.3 Management Plane Analytics

Network management includes a lot of practices to devices, and a network can be healthy or not under various practices. We made a quantitative analysis of the causal relationship between management practices and network health. We quantify a network's practices by the complexity of configuration files and operation actions, and quantify a network's health by the number of tickets. We use statistical analysis techniques such as Mutual Information and Quasi-Experimental Design to confirm the most influential practices to a network's health. We find the scale, device heterogeneity, and operation frequency are the major factors causing network failures. This work was published in IMC'15.[5]

5 Other Works

I made other research in a wide range of computer networks, including new transport layer protocol design for data centers [18]; future evolvable and secure Internet architecture[2, 6]; SDN-based scalable mobile core network and radio access network[15, 20]; video streaming optimization[16]; Internet DNS with SGX enhancement[13]; P4 program compilation optimization[1]; transaction network update[4]; SDN controler failure rollback[23].

References

[1] Anubhavnidhi Abhashkumar, Jeongkeun Lee, Jean Tourrilhes, Sujata Banerjee, **Wenfei Wu**, Joon-Myung Kang, and Aditya Akella. "**P5: Policy-driven optimization of P4 pipeline**". In: *Proceedings of the 2017 Symposium on SDN Research*. SOSR '17. Google Scholar Citation 15. 2017. URL: `https://dl.acm.org/doi/abs/10.1145/3050220.3050235`.

[2] Ashok Anand, Fahad Dogar, Dongsu Han, Boyan Li, Hyeontaek Lim, Michel Machado, **Wenfei Wu**, Aditya Akella, David G Andersen, John W Byers, et al. "**XIA: An architecture for an evolvable and trustworthy Internet**". In: *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. HotNets '11. Google Scholar Citation 154. 2011. URL: https://dl.acm.org/doi/abs/10.1145/2070562.2070564.

[3] Bangwen Deng, **Wenfei Wu***, and Linhai Song. "**Redundant Logic Elimination in Network Functions**". In: *Proceedings of the 2020 Symposium on SDN Research*. SOSR '20. Google Scholar Citation 0. 2020. URL: https://dl.acm.org/doi/10.1145/3373360.3380832.

[4] Qi-An Fu and **Wenfei Wu***. "**TUS: A Transactional Update Service for SDN Applications**". In: *Proceedings of the 9th ACM SIGOPS Asia-Pacific Workshop on Systems*. APSys '18. Google Scholar Citation 0. 2018. URL: https://dl.acm.org/doi/abs/10.1145/3265723.3265728.

[5] Aaron Gember-Jacobson, **Wenfei Wu**, Xiujun Li, Aditya Akella, and Ratul Mahajan. "**Management plane analytics**". In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. IMC '15. Google Scholar Citation 16. 2015. URL: https://dl.acm.org/doi/abs/10.1145/2815675.2815684.

[6] Dongsu Han, Ashok Anand, Fahad Dogar, Boyan Li, Hyeontaek Lim, Michel Machado, Arvind Mukundan, **Wenfei Wu**, Aditya Akella, David G. Andersen, John W. Byers, Srinivasan Seshan, and Peter Steenkiste. "**XIA: Efficient Support for Evolvable Internetworking**". In: *Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation*. NSDI '12. Google Scholar Citation 196. 2012. URL: https://www.usenix.org/conference/nsdi12/technical-sessions/presentation/han_dongsu_xia.

[7] Keqiang He, Weite Qin, Qiwei Zhang, **Wenfei Wu**, Junjie Yang, Tian Pan, Chengchen Hu, Jiao Zhang, Brent Stephens, Aditya Akella, and Ying Zhang. "**Low Latency Software Rate Limiters for Cloud Networks**". In: *Proceedings of the 2017 Asia-Pacific Workshop on Networking*. APNet '17. Google Scholar Citation 9. 2017. URL: https://dl.acm.org/doi/abs/10.1145/3106989.3107005.

[8] Yongchao He, **Wenfei Wu***, Xuemin Wen, Haifeng Li, and Yongqiang Yang. "**Scalable On-Switch Rate Limiters for the Cloud**". In: *the 2021 IEEE International Conference on Computer Communications*. INFOCOM '21. 2021.

[9] Hongyi Huang, **Wenfei Wu***, Yongchao He, Bangwen Deng, Ying Zhang, Yongqiang Xiong, Guo Chen, Yong Cui, and Peng Cheng. "**NFD: A Development Framework for Cross-Platform Network Functions**". In: *the 2021 IEEE International Conference on Computer Communications*. INFOCOM '21. 2021.

[10] Yimin Jiang, Yong Cui, **Wenfei Wu**, Zhe Xu, Jiahan Gu, K. K. Ramakrishnan, Yongchao He, and Xuehai Qian. "**SpeedyBox: Low-Latency NFV Service Chains with Cross-NF Runtime Consolidation**". In: *Proceedings of the 39th IEEE International Conference on Distributed Computing Systems*. ICDCS '19. Google Scholar Citation 6. 2019. URL: https://ieeexplore.ieee.org/document/8885120.

[11] ChonLam Lao, Yanfang Le, Kshiteej Mahajan†, Yixi Chen, Wenfei Wu, Aditya Akella, and Michael Swift. "**ATP: In-Network Aggregation for Multi-Tenant Learning**". In: *the 18th USENIX Symposium on Networked Systems Design and Implementation*. NSDI '21. 2021.

[12] Junfeng Li, Dan Li, **Wenfei Wu**, K. K. Ramakrishnan, Jinkun Geng, Fei Gui, Fanzhao Wang, and Kai Zheng. "**Sphinx: A Transport Protocol for High-Speed and Lossy Mobile Networks**". In: *Proceedings of the 38th IEEE International Performance Computing and Communications Conference*. IPCCC '19. Google Scholar Citation 0. 2019. URL: https://ieeexplore.ieee.org/abstract/document/8958769.

[13] Qingxiu Liu, **Wenfei Wu***, Qingsong Liu, and Qun Huang. "**T2DNS: A Third-Party DNS Service with Privacy Preservation and Trustworthiness**". In: *Proceedings of the 29th International Conference on Computer Communications and Networks*. ICCCN '20. Google Scholar Citation 0. 2020. URL: https://ieeexplore.ieee.org/document/9209638.

[14] Soo-Jin Moon, Jeffrey Helt, Yifei Yuan, Yves Bieri, Sujata Banerjee, Vyas Sekar, **Wenfei Wu**, Mihalis Yannakakis, and Ying Zhang. "**Alembic: Automated Model Inference for Stateful Network Functions**". In: *Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation*. NSDI '19. Google Scholar Citation 8. 2019. URL: `https://www.usenix.org/conference/nsdi19/presentation/moon`.

[15] Mehrdad Moradi, **Wenfei Wu**, Li Erran Li, and Zhuoqing Morley Mao. "**SoftMoW: recursive and reconfigurable cellular WAN architecture**". In: *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*. CoNEXT '14. Google Scholar Citation 63. 2014. URL: `https://dl.acm.org/doi/abs/10.1145/2674005.2674981`.

[16] Qingmei Ren, Yong Cui, **Wenfei Wu**, Changfeng Chen, Yuchi Chen, Jiangchuan Liu, and Hongyi Huang. "**Improving Quality of Experience for Mobile Broadcasters in Personalized Live Video Streaming**". In: *2018 IEEE/ACM 26th International Symposium on Quality of Service*. IWQoS '18. Google Scholar Citation 1. 2018. URL: `https://ieeexplore.ieee.org/document/8624178`.

[17] Harsha Sharma, **Wenfei Wu***, and Bangwen Deng. "**Symbolic Execution for Network Functions with Time-Driven Logic**". In: *Proceedings of the 2020 Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*. MASCOTS '20. Google Scholar Citation 0. 2020. URL: `https://cloud.tsinghua.edu.cn/f/7a4fef25d7eb4bb281d2/`.

[18] **Wenfei Wu**, Yizheng Chen, Ramakrishnan Durairajan, Dongchan Kim, Ashok Anand, and Aditya Akella. "**Adaptive data transmission in the cloud**". In: *2013 IEEE/ACM 21st International Symposium on Quality of Service*. IWQoS '13. Google Scholar Citation 5. 2013. URL: `https://ieeexplore.ieee.org/abstract/document/6550262`.

[19] **Wenfei Wu**, Keqiang He, and Aditya Akella. "**Perfsight: Performance diagnosis for software dataplanes**". In: *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. IMC '15. Google Scholar Citation 30. 2015. URL: `https://dl.acm.org/doi/abs/10.1145/2815675.2815698`.

[20] **Wenfei Wu**, Li Erran Li, Aurojit Panda, and Scott Shenker. "**PRAN: Programmable radio access networks**". In: *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. HotNets '14. Google Scholar Citation 28. 2014. URL: `https://dl.acm.org/doi/abs/10.1145/2670518.2673865`.

[21] **Wenfei Wu**, Guohui Wang, Aditya Akella, and Anees Shaikh. "**Virtual network diagnosis as a service**". In: *Proceedings of the 4th annual Symposium on Cloud Computing*. SoCC '13. Google Scholar Citation 33. 2013. URL: `https://dl.acm.org/doi/abs/10.1145/2523616.2523621`.

[22] **Wenfei Wu**, Ying Zhang, and Sujata Banerjee. "**Automatic Synthesis of NF Models by Program Analysis**". In: *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. HotNets '16. Google Scholar Citation 27. 2016. URL: `https://dl.acm.org/doi/abs/10.1145/3005745.3005754`.

[23] Ye Yu, Ying Zhang, **Wenfei Wu**, and Chen Qian. "**NetCP: Consistent, Non-interruptive and Efficient Checkpointing and Rollback of SDN**". In: *2018 IEEE/ACM 26th International Symposium on Quality of Service*. IWQoS '18. Google Scholar Citation 3. 2018. URL: `https://ieeexplore.ieee.org/abstract/document/8624142`.