A Vision For Hierarchical Federated Learning in Dynamic Service Chaining

Abdullah Bittar
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Canada
abdullahbittar@cmail.carleton.ca

Changcheng Huang
Dept. of Systems and Computer Engineering
Carleton University
Ottawa, Canada
huang@sce.carleton.ca

2022 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN) | 978-1-6654-7334-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/NFV-SDN56302.2022.9974900

***Abstract*—We have seen the tremendous expansion of machine learning (ML) technology in Artificial Intelligence (AI) applications, including computer vision, voice recognition, and many others. The availability of a vast amount of data has spurred the rise of ML technologies, especially Deep Learning (DL). Traditional ML systems consolidate all data into a central location, usually a data center, which may breach privacy and confidentiality rules. The Federated Learning (FL) concept has recently emerged as a promising solution for mitigating data privacy, legality, scalability, and unwanted bandwidth loss problems. This paper outlines a vision for leveraging FL for better traffic steering predictions. Specifically, we propose a *hierarchical FL* framework that will dynamically update service function chains in a network by predicting future user demand and network state using the FL method.**

***Index Terms*—Service Function Chaining, SFC, Federated Learning, FL, Dynamic**

I. Introduction

Network Function Virtualization (NFV) implements Network Functions (NFs) and middlebox services in software instead of purpose-built hardware appliances [1]. NFs are commonly chained, where a packet is handled by a succession of NFs, known as Service Function Chaining (SFC), before being forwarded to the destination. SFC plays a vital role in next-generation networks supporting technologies such as 5G, IoT, and edge computing.

SFC orchestration is considered an NP-hard problem. Most previous work either focused on static or dynamic orchestration. However, these methods are heuristic and do not adapt to the traffic trend in the network in large state spaces. There is a need for a dynamic operation to achieve service chaining, which will reduce configuration and management complexities. With the emerging programmable networks [2], [3], the time has come to re-think the network architecture to leverage the benefits.

Deep Learning (DL) systems can absorb a large amount of data and execute several tasks that occasionally surpass human performance. Traditional DL systems are usually centralized, storing and processing all data. This creates a new set of challenges, such as privacy and confidentiality of the dataset and scalability. Federated Learning (FL), a decentralized learning process, can be a solution for dynamic orchestration while preserving users' privacy.

Self-driving networks are becoming the trend [4]. With self-driving networks, network operators merely define high-level intentions, and the network is self-managed. The ambition of a self-driving network is appealing, but how to achieve it is an open question. This paper proposes a novel method for improving service chaining orchestration. We lay out a vision for leveraging FL for better traffic orchestration prediction. The FL approach can predict user demand and network state to aid dynamic traffic steering decisions. Our FL framework consists of multiple FL agents in the network running the DL model on local data and sharing the DL model's high-level features with the central controller for global model training.

This paper will cover a background on FL concept and related work in Section II, problem statement and motivation in Section III, opportunities and challenges in Section IV, current status in Section V, and finally, a conclusion in Section VI.

II. Background and Related Works

This section will include background details on Federated Learning (FL) in subsection II-A and related work in Subsection II-B.

A. Federated Learning (FL)

Google researchers first proposed the FL idea in 2016 [5] as a viable solution to the problems of communication costs, data privacy, and legality. FL seeks to create a collaborative ML model based on participants called 'clients' while a central coordinator called 'server' oversees the process. The FL consists of two phases: model training and model inference. During model training, clients (end devices, organizations, or individuals) use local data to train local models and then share high-level model features with a server; no user data is exchanged between the client and the server. The server gathers the trained high-level features to create and update a higher-level model, which sends back model parameters to the clients for local model refining. In the inference phase, clients use the refined models to process their data. In a formal manner, consider N client data owner F_i, $i = 1, ..., N$ participating in training an ML model using their respective datasets $D_i, i = 1, ..., N$. In a conventional approach, all data $D_i, i = 1, ..., N$ are collected together at one server to train

an ML model M_{SUM}. In the FL approach, client data owners collaboratively train a model M_{FED} without collecting all data $D_i, i = 1, ..., N$ into a centralized location.

B. Related Works

Over the past few years, many SFC orchestration approaches have been proposed. NFVdeep [6] proposed to deploy SFCs for requests with different QoS requirements automatically. Deep reinforcement learning, serialization and backtracking methods were used to deal with SFC deployment in the real-time network. STAR [7] divides the entire routing path into several path segments. It leverages the characteristics of path segments to allow packets from different requests to share the same forwarding rules. In [8], the authors present a learning-based (deep-Q learning) dynamic VNF scaling scheme to adaptively trigger and perform VNF resource scaling and migration decisions based on detected traffic statistical changes in a real-world non-stationary traffic trace, in order to satisfy the probabilistic delay requirement consistently. However, none of the abovementioned works consider the whole SFC process dynamically.

We found one paper [9] focusing on a similar topic, SFC orchestration. The paper proposed a novel scalable SFC chaining algorithm for NFV-enabled networks through a federated reinforcement learning technique. However, their work is limited to Virtual Machines (VMs) and does not scale to large-scale networks.

III. Problem Statement and Motivation

Previous research papers proposed solutions for different kinds of issues to improve SFC: chain composition and load balancing [10], making NF software-defined [11], parallel NFs [12], dividing chain into sub-chain connections between hops [13], enhancing OS scheduler for NFs chain fairness [14], support L4-L7 middleboxes [15], adding In-band Network Telemetry (INT) to NF [16], fault-tolerant to the entire chain [17], and performance diagnosis tool for NF chain using queuing periods [18]. Notably, the last few articles (years) have focused on improving the chain, not on its dynamism.

Future direction/research in improving SFC has been discussed recently. In [19], the author proposes improving the SFC architecture with a well-synchronized monitoring system that collects the necessary network data. In [20], the author mentioned that a dynamic scenario in the real world is still an open research area. Furthermore, the author in [21] identified that "research on the SFC traffic steering problem is in its infancy" and added that the online dynamic SFC orchestration field in inter-DC multi-domain scenarios is still a big challenge. Finally, [22] highlighted that more work is needed to get the current network state and dynamically update the current Service Function (SF) path in real time.

In our previous work [23], we designed and implemented an SFC network in Kubernetes using the Network Service Mesh framework. Further, we demonstrated a prototype [24] that dynamically allows users to deploy network function chains using a web interface-based orchestration. However, our work falls short in predicting user and network states to achieve better traffic orchestration.

Our analysis of the previous works mentioned in this section, future research directions, and our learned lessons indicate a lack of a flexible and efficient solution for a dynamic SFC framework. All research papers focus on adding a new feature to improve SFC (such as adding parallel NFs or adding INT to NFs). However, no research paper focused on improving the SFC framework to make it dynamic. Therefore, we were motivated to propose a dynamic SFC framework. Our general goal is to lower costs by using resources more efficiently. Specifically, we suggest using a more dynamic SFC to increase resource efficiency. Dynamic SFC has overhead, which raises scalability challenges. We propose *hierarchical FL* (HFL) to minimize the overhead and achieve scalability. A side benefit is privacy that comes with FL architecture. HFL imposes challenges, and we focus on *how to predict the future network state and user demand to dynamically update SFC in the network to achieve better traffic orchestration in the next-generation network.* To tackle this challenge, we must consider two critical questions in the design process. The first question is, what are the challenges to collecting dynamic information on network state and user demand? The second question is how to distribute high-level features.

IV. Opportunities and Challenges

Many research and standardization projects have enabled the SFC paradigm, although most have concentrated on one administrative domain. Due to lack of resources, tight latency or resilience requirements, or functional reasons, a service may need deployment in more than one domain. In SDN and NFV, cloud servers supply compute resources for VNFs, which compose the SFC. After the placement process, traffic needs to be routed between VNFs. This problem is exacerbated when VNFs are located in different domains, facing routing difficulties. Furthermore, SFC requires dynamic operations to reduce configuration and management complexity. Here, an FL architecture in subsection IV-A can be leveraged to provide solutions for a dynamic network operation described in IV-B that would remove the scalability addressed in IV-C and bandwidth utilization and privacy issues discussed in IV-D. However, this will introduce new challenges in IV-E, which need further research.

A. Architecture Proposal

The framework we argue for leverages a critical feature of FL architecture: the decentralized concept. Figure 1 illustrates the proposed framework. The HFL framework consists of an FL central controller with multiple FL agents across numerous domains. Each FL agent will use local data to train a local DL model and send the high-level features to the central controller. The central controller uses the aggregated local features to train a global DL model that can predict global traffic states and demands in longer terms. This process iterates as much as needed until an optimal solution is achieved. Meanwhile, the central controller sends parameter recommendations from its

global model to the agent in each domain so that each agent can refine its model to incorporate global features. During the inference phase, agents will feed dynamic high-level traffic features to the central controller. The central controller can then predict global long-term traffic trends among different domains and distribute them to the agents in each domain. By utilizing these high-level features, the agents in each domain can predict local traffic more accurately and help SDN controllers schedule resources in both the long and short terms. Hence, the SDN controller will update the network switches accordingly. We propose the FL agent reside on servers and not switches because switches have limited resources for the FL agent to run on, and secondly, switches only see traffic flow. In contrast, servers provide actual workload and resource usage.

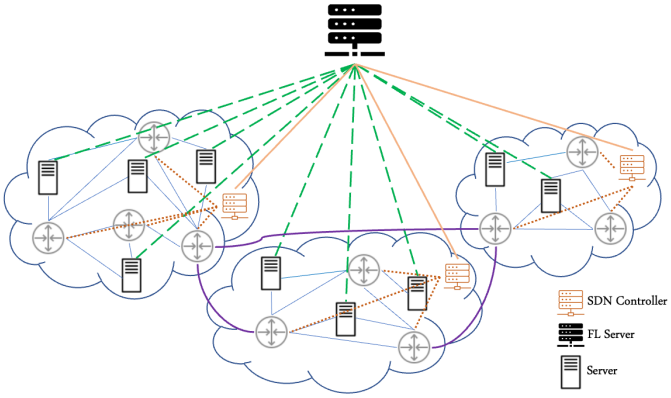

Fig. 1. Hierarchical FL (HFL) Architecture

B. Dynamic Operation

Dynamic SFC has multiple dimensions, which include dynamic flow migration, resource scheduling/embedding, and resource scaling. Dynamic resource management among embedded services is vital to enable efficient and fair service delivery over virtual networks with a QoS guarantee. Although dynamic operation for SFC has been studied before, more work is needed to develop a dynamic system with all dimensions. One of the essential opportunities FL provides is its dynamic nature. Stochastic traffic arrival requests cause significant fluctuations in network state and traffic, necessitating a suitable model to represent these dynamic changes in network transition. As indicated in Figure 1, the HFL agents in the network will convey the aggregated future resource utilization and workload to the central controller. The central controller will utilize agent output features and train a global model, leading to a prediction of the future network state at higher granularity. The refined model in each HFL agent will provide a much faster, more accurate, dynamic, and more efficient method than conventional solutions [25]. Hence, the FL model with its agents eliminates static orchestration and predicts network state and user demands to trigger dynamic adaptation.

C. Scalability

Scalable, rapid, and agile network operation is essential for future networks. The opportunity of having scaling features in a network can result in low maintenance costs, better user experience, and greater agility. SDN has a centralized architecture design, thus lacking scalability to fulfill large-scale network requirements. With millions or billions of user devices, scalability issues arise. FL can eliminate such scalability limitations with its decentralized approach [26]. The HFL model can learn users' trends in a network and act accordingly when an increase in demand enters a network. Using the dispersed HhFL agents in the network, HFL agents can share their aggregated states (resource utilization and workload) with the central controller to trigger and perform resource scaling. The central controller will utilize HFL agents' states and find an optimized scaling scheme. This would be possible only if the network is dynamic since flow migration will occur between resources to consistently satisfy the probabilistic delay requirement.

D. Bandwidth Utilization and Privacy

In the traditional distributed DL model, there is heavy communication between the central node and worker nodes [27], leading to considerable bandwidth consumption. Furthermore, bandwidth may be lost due to loss of packets or/and retransmission. Transferring a large amount of data between nodes in a network is not trivial and requires a tuning algorithm for successful transmission. HFL utilizes bandwidth more efficiently to overcome centralized training and inference issues such as computational demand and network bandwidth limitation. HFL agents will train a DL model using local data, which will not be transferred to the central controller, saving tremendous bandwidth. Each HFL agent will only send the aggregated high-level features to the central controller for training and inferencing. Reducing network communication will reduce bandwidth costs significantly, especially when saving on raw data transmission.

In general, centralized DL systems are concerned with various issues, most crucially, the long-neglected security and privacy of user data. As advocated by [28], FL can be the solution. FL provides a method for preserving user privacy by decentralizing data from the central server to end devices. DL models will only be applied to local datasets with sensitive data and heterogeneity. With its decentralized data approach, FL has been one of the fastest-growing areas of DL in recent years, as its security and privacy characteristics promise compliance with rising user data protection legislation [29].

E. Challenges

Leveraging FL in obtaining network state and users' demand from the network that follows similar architecture as in Figure 1 faces new challenges. Some of the unique challenges we anticipate are:

- Communication: in FL, communication is a crucial bottleneck. Communication-efficient approaches that dynamically transmit high-level features between the central

controllers and agents during the inference phase are necessary. To reduce communication in the HFL approach, there are two crucial factors to consider: *(i)* decreasing the total number of communication rounds through long-term prediction or *(ii)* reducing the amount of communicated messages through aggregated features at each round.

- Privacy: although one of FL's goals is privacy by protecting data generated in each device and only sharing high-level features, the shared high-level features may still divulge sensitive information to third parties or the central controller. To maintain privacy in an FL network, there is a tradeoff between privacy and reduced model performance or system efficiency.
- FL and SDN incorporation: the proposed HFL framework imposes a new set of challenges when combining HFL and SDN. HFL will convey future network predictions to SDN, where SDN will update the network accordingly. Further investigation into the communication protocol between FL and SDN is required.
- FL in production: practically, there is a need for a framework that outlines specific action steps when the FL agent communicates with the central controller. For example, there might be significant variations in the FL agent model update in a small time interval, non-malicious errors and steps in adding or removing devices from the global FL model.

V. Current Status

The design of our proposed HFL framework differs from other FL architectures in that it is network-oriented, i.e., using FL to solve networking problems. The closest FL architecture to our approach is Vertical FL (VFL). However, VFL focuses on exchanging parameters instead of features [30]. VFL is more ML-oriented than network-oriented. In ML, features represent the inputs or the outputs at various levels, which change with different inputs during the inference stage. Parameters are the variables defining a model, which do not change with different inputs during the inference stage. Our approach is to exchange parameters and aggregate features between agents and the central model rather than exchanging parameters only.

Hence, we propose using the HFL framework to solve the most significant challenge in traffic information distribution, namely scalability. With the increasing NFV-based applications, traffic volume in a network is substantial and dynamic, and the resources required to operate the service (VM/container) fluctuate over time [6]. This leads to a critical question regarding the data collection granularity, microflow or macroflow? The idea of using HFL is to make the process distributed and scalable. Using the HFL approach, a DL model will be leveraged to extract aggregate features from microflow information at each router or service node. HFL agents will then send the high-level features to a central location or all other routers/service nodes in a macroflow manner. Hence, this will aid routers/service nodes in steering traffic efficiently.

We need information from the network to obtain current user demand and network state for dynamically updating SFC in the network for better traffic steering. Two types of network information can be gathered: *(i)* resource management which focuses mainly on server resources, e.g., CPU and *(ii)* memory, and congestion control which targets network bandwidth and flow completion time.

Choosing what information to collect from the network is not a trivial task. Hence, we reviewed some high-quality research papers to see what information they used for their algorithms. DCTCP and D2TCP rely on ECN as the congestion signal and react proportionally. Protocols such as TIMELY, SWIFT, CDG [31], and DX [32] rely on RTT measurements for window update calculation. XCP [33], D3 [34], and RCP [35] rely on explicit network feedback based on rate calculations within the network. HPCC uses INT protocol while PowerTCP [36] also uses the same feedback signal as HPCC but uses the notion of power to update window size.

Similarly, we have surveyed high-quality research papers to choose what parameters for resource management. DFR [37] and Apollo uses CPU and memory data. Quincy is a queue-based scheduling approach. TetriSched leverages metadata from Ryan [38], a novel reservation-based scheduling system by proposing Reservation Definition Language (RDL), to provide a scheduling system rather than a reservation system to make short-term job placement and ordering decisions. Karios [39] uses the Least Attained Service (LAS) as the scheduling policy by implementing the number of cores and the quantum time for the smallest amount of time so far of service execution. Finally, Hydra [40] leverages the Yarn Federation architecture, in which a collection of loosely coupled sub-clusters coordinates to provide the illusion of a single massive cluster.

VI. Conclusion

This paper argues for a more dynamic operation network using the HFL approach. The HFL approach has great potential to solve different network issues, such as privacy, scalability, and bandwidth utilization. Developing and implementing dynamic network orchestration has been difficult in the past because the technologies were not supportive enough. However, network programmability emerged a few years ago and is becoming mature. Furthermore, ML approaches have been widely adopted to solve different network issues, giving the network administrator more flexibility. This paper proposes a new FL framework called *Hierarchical FL* as a solution for dynamic SFC. HFL framework aims to minimize the overhead and achieve scalability while preserving privacy. This is possible by leveraging the decentralization nature of the FL approach.

References

[1] Joel Halpern, Carlos Pignataro, et al. Service function chaining (sfc) architecture. In *RFC 7665*, pages 1–32. 2015.

[2] Jiarong Xing, Yiming Qiu, Kuo-Feng Hsu, Hongyi Liu, Matty Kadosh, Alan Lo, Aditya Akella, Thomas Anderson, Arvind Krishnamurthy, TS Eugene Ng, et al. A vision for runtime programmable networks. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*, pages 91–98, 2021.

[3] Gianni Antichi and Gábor Rétvári. Full-stack sdn: The next big challenge? In *Proceedings of the Symposium on SDN Research*, pages 48–54, 2020.

[4] Eric Liang, Hang Zhu, Xin Jin, and Ion Stoica. Neural packet classification. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 256–269. 2019.

[5] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[6] Yikai Xiao, Qixia Zhang, Fangming Liu, Jia Wang, Miao Zhao, Zhongxing Zhang, and Jiaxing Zhang. Nfvdeep: Adaptive online service function chain deployment with deep reinforcement learning. In *Proceedings of the International Symposium on Quality of Service*, pages 1–10, 2019.

[7] Ruixin Chen and Jin Zhao. Scalable and flexible traffic steering for service function chains. *IEEE Transactions on Network and Service Management*, 2022.

[8] Weihua Zhuang and Kaige Qu. Dynamic vnf resource scaling and migration: A machine learning approach. In *Dynamic Resource Management in Service-Oriented Core Networks*, pages 85–129. Springer, 2021.

[9] Haojun Huang, Cheng Zeng, Yangmin Zhao, Geyong Min, Yingying Zhu, Wang Miao, and Jia Hu. Scalable orchestration of service function chains in nfv-enabled networks: A federated reinforcement learning approach. *IEEE Journal on Selected Areas in Communications*, 39(8):2558–2571, 2021.

[10] Zafar Ayyub Qazi, Cheng-Chun Tu, Luis Chiang, Rui Miao, Vyas Sekar, and Minlan Yu. Simple-fying middlebox policy enforcement using sdn. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 27–38, 2013.

[11] Anat Bremler-Barr, Yotam Harchol, and David Hay. Openbox: a software-defined framework for developing, deploying, and managing network functions. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 511–524, 2016.

[12] Chen Sun, Jun Bi, Zhilong Zheng, Heng Yu, and Hongxin Hu. Nfp: Enabling network function parallelism in nfv. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 43–56, 2017.

[13] Pamela Zave, Ronaldo A Ferreira, Xuan Kelvin Zou, Masaharu Morimoto, and Jennifer Rexford. Dynamic service chaining with dysco. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 57–70, 2017.

[14] Sameer G Kulkarni, Wei Zhang, Jinho Hwang, Shriram Rajagopalan, KK Ramakrishnan, Timothy Wood, Mayutan Arumaithurai, and Xiaoming Fu. Nfvnice: Dynamic backpressure and scheduling for nfv service chains. *IEEE/ACM Transactions on Networking*, 28(2):639–652, 2020.

[15] Guyue Liu, Yuxin Ren, Mykola Yurchenko, KK Ramakrishnan, and Timothy Wood. Microboxes: High performance nfv with customizable, asynchronous tcp stacks and dynamic subscriptions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 504–517, 2018.

[16] Jianzhe Liang, Jun Bi, Yu Zhou, and Cheng Zhang. In-band network function telemetry. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*, pages 42–44, 2018.

[17] Milad Ghaznavi, Elaheh Jalalpour, Bernard Wong, Raouf Boutaba, and Ali José Mashtizadeh. Fault tolerant service function chaining. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 198–210, 2020.

[18] Junzhi Gong, Yuliang Li, Bilal Anwer, Aman Shaikh, and Minlan Yu. Microscope: Queue-based performance diagnosis for network functions. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 390–403, 2020.

[19] Ahmed M Medhat, Tarik Taleb, Asma Elmangoush, Giuseppe A Carella, Stefan Covaci, and Thomas Magedanz. Service function chaining in next generation networks: State of the art and research challenges. *IEEE Communications Magazine*, 55(2):216–223, 2016.

[20] Karamjeet Kaur, Veenu Mangat, and Krishan Kumar. A comprehensive survey of service function chain provisioning approaches in sdn and nfv architecture. *Computer Science Review*, 38:100298, 2020.

[21] Shuyi Wang, Haotong Cao, and Longxiang Yang. A survey of service function chains orchestration in data center networks. In *2020 IEEE Globecom Workshops (GC Wkshps*, pages 1–6. IEEE, 2020.

[22] Haruna Umar Adoga and Dimitrios P Pezaros. Network function virtualization and service function chaining frameworks: A comprehensive review of requirements, objectives, implementations, and open research challenges. *Future Internet*, 14(2):59, 2022.

[23] Abdullah Bittar, Ziqiang Wang, Amir Aghasharif, Changcheng Huang, Gauravdeep Shami, Marc Lyonnais, and Rodney Wilson. Service function chaining design & implementation using network service mesh in kubernetes. In *Asian Conference on Supercomputing Frontiers*, pages 121–140. Springer, Cham, 2022.

[24] Ziqiang Wang, Abdullah Bittar, Changcheng Huang, Chung-Hong Lung, and Gauravdeep Shami. A web-based orchestrator for dynamic service function chaining development with kubernetes. In *2022 IEEE 8th International Conference on Network Softwarization (NetSoft)*, pages 234–236. IEEE, 2022.

[25] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[26] Morgan Ekmefjord, Addi Ait-Mlouk, Sadi Alawadi, Mattias Åkesson, Desislava Stoyanova, Ola Spjuth, Salman Toor, and Andreas Hellander. Scalable federated machine learning with fedn. *arXiv preprint arXiv:2103.00148*, 2021.

[27] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, pages 1–33, 2022.

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[29] White House. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *White House, Washington, DC*, 1:120, 2012.

[30] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

[31] David A Hayes and Grenville Armitage. Revisiting tcp congestion control using delay gradients. In *International Conference on Research in Networking*, pages 328–341. Springer, 2011.

[32] Changhyun Lee, Chunjong Park, Keon Jang, Sue Moon, and Dongsu Han. Accurate latency-based congestion feedback for datacenters. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 403–415, 2015.

[33] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 89–102, 2002.

[34] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. Better never than late: Meeting deadlines in datacenter networks. *ACM SIGCOMM Computer Communication Review*, 41(4):50–61, 2011.

[35] Nandita Dukkipati and Nick McKeown. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, 36(1):59–62, 2006.

[36] Vamsi Addanki, Oliver Michel, and Stefan Schmid. {PowerTCP}: Pushing the performance limits of datacenter networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 51–70, 2022.

[37] Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, and Ion Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *8th USENIX Symposium on Networked Systems Design and Implementation (NSDI 11)*, 2011.

[38] Carlo Curino, Djellel E Difallah, Chris Douglas, Subru Krishnan, Raghu Ramakrishnan, and Sriram Rao. Reservation-based scheduling: If you're late don't blame us! In *Proceedings of the ACM Symposium on Cloud Computing*, pages 1–14, 2014.

[39] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. Kairos: Preemptive data center scheduling without runtime estimates. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 135–148, 2018.

[40] Carlo Curino, Subru Krishnan, Konstantinos Karanasos, Sriram Rao, Giovanni M Fumarola, Botong Huang, Kishore Chaliparambil, Arun Suresh, Young Chen, Solom Heddaya, et al. Hydra: a federated resource manager for data-center scale analytics. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 177–192, 2019.