CON: A Computation-Oriented Network for Efficient Edge Intelligence

Ruidong Li, Tomoki Hirayama, Kaiping Xue, Peiying Ruan, and Hitoshi Asaeda

Abstract

Edge intelligence enables edge devices to collaboratively perform computations and generate knowledge where distributed computations act as the main innovative technologies. The existing solutions for distributed computations mainly focus on the designs of computations, such as distributed machine learning algorithms. By contrast, edge intelligence poses the design requirements for efficient networking technology, including one request multiple reply in a time Interval (ORMRI), computation aggregation, and chained computation, which have not been well addressed. To satisfy those requirements, we design a Computation-Oriented Network (CON) to enable the efficient discovery, task allocation, and collaborative computations for edge devices in this article. In CON, model and datasets are afforded with the names for direct resource discovery, and a hybrid computation oriented routing (HCOR) mechanism has been proposed, which combines proactive and reactive routing approaches. Through CON, computation requests and replies can be sent to and collected from multiple suitable edge devices, respectively; computation aggregation is enabled during the computation result replying process; and chained computation is supported through sequentially popping out model names. We examine the performance of CON through analysis and implementation, which shows that the proposed CON can greatly reduce the communication bandwidth consumption compared to the existing end-to-end approach.

Introduction

Billions of mobile devices and small things, such as sensors, RFIDs, actuators, and robots, are generating tremendous amounts of big data [1], where computations play the crucial role to provide intelligent and Metaverse services. The computations, such as big data processing, mainly rely on the centralized clouds to perform data analytics, which however cannot satisfy the societal needs on privacy, diversity, and efficiency. Recently accelerated by the success of big data analytics and Internet of Things (IoT) technologies, edge intelligence has attracted wide attentions as the method to solve those problems, which moves the computation from centralized cloud to edge devices.

For edge intelligence, the distributed computations, such as distributed machine learning and inference [2], act as the main innovative technologies to enable edge devices to collaboratively perform computations and generate knowledge [3, 4]. The existing work in this area mainly focused on the design of computation algorithms to enable them to achieve more accurate and correct prediction, decision, or classification. However, very few attentions have been put on enabling network technologies to efficiently support the distributed computations at edge.

Regarding networking for edge intelligence, how to discover close edge devices with computation resources, federate those devices to efficiently perform computations, and further provide services is an indispensable challenging issue. Recently, interests on that issue within Internet Research Task Force (IRTF) and IEEE are also growing as are evidenced by the formation of a new research group, Computation in the Network Research Group (COINrg) [5], and new special interest group of IEEE Intelligent Internet Edge (IIE) [6], respectively.

To address this issue, we focus on a typical category of distributed computation, distributed machine learning, to design networking technologies, where computations are conducted over datasets at edge devices. For distributed machine learning, the systems for processing massive amounts of data largely rely on utilizing a number of edge devices, each of which holds a relatively small storage capacity and computing power, rather than one expensive large server [2].

The logic communication structure in a distributed machine learning scenario, which illustrates the relations among users and edge devices, can be tree, peer-to-peer, or chain. To enable a network to naturally and efficiently provide communications for distributed machine learning, those structures pose the design requirements on networking technology as follows:

- Provide a novel communication paradigm of One Request Multiple Reply in a time Interval (ORMRI). For distributed machine learning, the function enabling a user to send one "computation request" and then receive multiple "computation results" in a time interval after that is beneficial for whatever tree or peer-to-peer structure. This communication paradigm we call ORMRI is to be supported.
- Support computation aggregation. The inter-

Ruidong Li (corresponding author) is with Kanazawa University, Japan; Tomoki Hirayama was with Kanazawa University and is now working at Sansan Inc., Japan; Kaiping Xue is with University of Science and Technology of China, China; Peiying Ruan is with NVIDIA, Japan; Hitoshi Asaeda is with the National Institute of Information and Communications Technology, Japan.

Digital Object Identifier: 10.1109/MNET.001.2200200

mediate nodes aggregating the computations results obtained from multiple upstream nodes and then forwarding the aggregated one to the downstream node(s) is necessary for all three communication structures. The intermediate node here denotes router, and we use them interexchangely in this article.
- Offer the chained computation. Performing computation over another node after completing computation task at one node is the basic requirement for a chain structure.

Currently, distributed machine learning is implemented based on the Internet networking technology following end-to-end approach, where TCP/IP is used for data transmissions from sources to destinations. Obviously, the Internet networking technology cannot easily satisfy the design requirements on ORMRI, computation aggregation, and chained computation, since it intrinsically deviates the design goal and violates the original end-to-end design principle. For the other emerging networking paradigms, such as information-centric network/named data network (ICN/NDN) [7] and software-defined network (SDN) [8], they also hold different purposes and cannot inherently support those logic communication structures.

To satisfy those design requirements, we design a Computation-Oriented Networking (CON) to enable the efficient discovery, task allocation, and collaborative computation for the close edge devices with computation resources. In CON, models and datasets are afforded with the names, which can be used to directly request and achieve computations over specific dataset(s) and model(s). Further, we design a Hybrid Computation Oriented Routing (HCOR) combining proactive and reactive routing, through which computation requests and replies can be sent to and collected from multiple suitable edge devices, respectively. Also the computation aggregation and chained computation are provided in CON. We implement the forwarding and aggregation function of the proposed CON, and the performance evaluations show that CON can greatly reduce more than 25 percent bandwidth consumption under full and complete tree topology compared to the end-to-end approach.

The remainder of this article is organized as follows. In the next section, we discuss related work. Following that, we describe the use scenario and outline the design requirements. Then we elaborate on the design of CON. We provide the performance evaluation and then conclude our work.

Related Work

Edge intelligence refers to the distributed computation collaboratively provided by the connected edge devices proximity to users for data collection, caching, processing, and analysis based on artificial intelligence [3, 4, 9]. The innovative technologies for edge intelligence is distributed machine learning [2].

The computations at edge become feasible with the increase in number of devices connected to the Internet, where data processing can be performed at the devices surrounding users [1]. It shows great potential for the future applications with requirements on local computation, privacy, or real-time services, such as digital twins.

For distributed machine learning, most existing work is to find the accurate or correct algorithms for prediction, classification, or decision-making. In contrast, the networking technologies to efficiently support the communications among users and edge devices have not been well addressed...

Distributed machine learning [2] is a typical category of computation at edge. It has two fundamental different ways of partitioning the computation problem across edge devices, data-parallel approach (e.g., federated learning [10]) and model-parallel approach (e.g., ensemble learning [11]). For distributed machine learning, most existing work is to find the accurate or correct algorithms for prediction, classification, or decision-making. In contrast, the networking technologies to efficiently support the communications among users and edge devices have not been well addressed, which is the targeted issue in this article.

The existing networking technology to support the distributed machine learning is the end-to-end approach, the Internet TCP/IP protocol suit. Its main function is to move data from one device to another device. There was also application service design to provide multicast/multicollect service [12]. However, the end-to-end approach does not naturally provide model or dataset discovery, which needs an additional third-party to provide the matching service between computation requests and computation resources. Besides, it cannot easily satisfy the design requirements on ORMRI, computation aggregation, and chained computation, since it was intrinsically designed for communication establishment between two end nodes and those requirements violate its end-to-end design principle.

On the other hand, ICN/NDN [7] and SDN [8] have emerged as the promising future network architectures. ICN/NDN focuses on the fast data retrieval using their names under the assumption that data can be cached at routers. With it, one data chunk can be acquired through issuing one request in a quick manner, which does not aggregate multiple replies per-request and does not chain the replies. SDN does not change the basic rules of end-to-end approach, which only enables the network management to change from a decentralized model to a centralized model. Therefore, both of them do not provide method to natively enable edge nodes to collaborate with each other on ORMRI, computation aggregation and chaining. Also they cannot be easily modified to enable the support of those functions, because both of them aim to efficiently move data from one place to another place per-request instead of collaborative computations.

Distributed Computation

For the centralized computation, data are collected from physical environment and transmitted to a centralized cloud for processing and analyzing. This approach may bring large data transmission costs and delays because of remote computation. As a promising approach to solve those problems, edge intelligence [3, 9] brings the computations from centralized cloud to edge devices, where computations can be completed locally. Along this technological road, users at edge discover edge devices with available computation resourc-

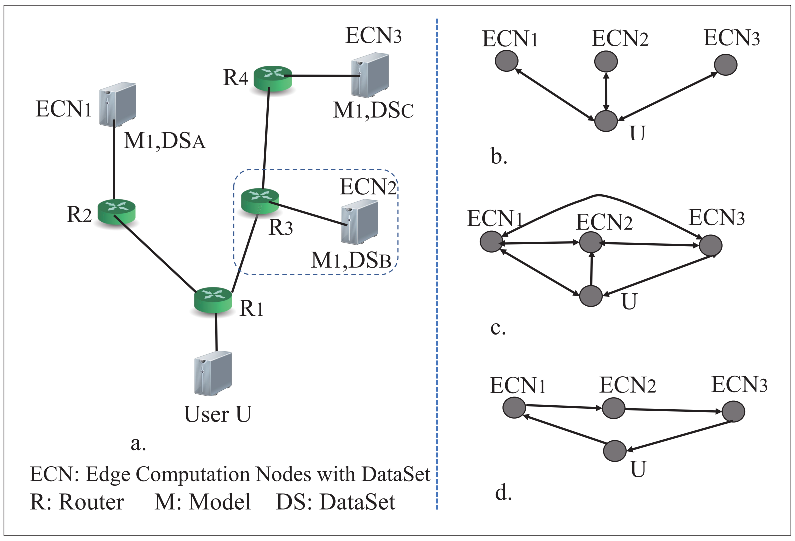


FIGURE 1. Physical topology and logic structure: a) physical topology; b) logic structure: tree; c) logic structure: peer-to-peer; d) logic structure: chain.

es and further assign computation tasks to them. Then, those edge devices will locally analyze data and provide partial results, which are further merged into a (partial) result through collaboration and aggregation. Finally, this result is replied to the requesting user. Here we focus on distributed machine learning [2] to find the network design requirements posed from the distributed computation use scenarios.

To realize such parallel computations at edge, whether logic communication structure of edge devices can match physical topology or not plays a crucial role to enable data exchanges and aggregations to be efficient. Fig. 1 illustrates an example physical topology and logic structures for distributed machine learning. As in Fig. 1a, there are three edge devices with computation resources, ECN_1, ECN_2, and ECN_3, four routers, R_1, R_2, R_3, and R_4, and one user U. ECN_1, ECN_2, and ECN_3 can perform the computation on machine learning model, M_1, while they hold different datasets, DS_A, DS_B, and DS_C, respectively. Routers may also be equipped with computation resources, and thus ECN_2 can be set together with R3 to reflect such possibility in Fig. 1a. In this example topology, User U wants to perform the computations of model M_1 on datasets DS_A, DS_B, and DS_C.

The right side of Fig. 1 shows the possible logic structures for edge computation use cases, which can be tree, peer-to-peer, or chain. As the tree structure in Fig. 1b, ECN_1, ECN_2, and ECN_3 carry out the computations and send results back to U, respectively and U merges the results to generate the final knowledge. The tree structure shown in Fig. 1b is the most simple one, which can be upgraded to a hierarchical one. For example, data are analyzed and results are forwarded to downstream nodes; downstream nodes aggregate them and further forward to lower downstream nodes for mergence. Its typical use scenario is federated learning [10]. Peer-to-peer logic structure corresponds to the peer-to-peer learning [13], which is basically an extension of tree structure. With it, one leader node is selected for each round; tree structure is formed; and then data analysis results under each tree structure are merged.

For a chain structure in Fig. 1d, user U sends computation request to ECN_1, who performs computation and sends the analysis results to ECN_2. ECN_2 carries out the similar action and moves forward. This process continue iteratively until the final result is replied to U. One typical use case for this logic structure is cyclic transfer learning [14].

The logic structure shows the interactions among ECNs and user, while the physical topology provides the communication services to support those interactions. The main interaction for tree structure is ORMRI. That is, user sends out one request and gets multiple replies from the close edge devices in a time interval. In the example physical topology in Fig. 1a, U sends out request for computation tasks, and ECN_1, ECN_2, and ECN_3 locally perform the corresponding computations (e.g., Model M_1 with datasets as DS_A, DS_B, and DS_C) and reply the results back to U, respectively. The intermediate nodes then conduct the aggregation over those results during transmissions and return the final merged result to U. For a hierarchical tree structure, nodes at the same level can send out the aggregated computation results, which will be further aggregated at the downstream devices. Peer-to-peer structure is an extension of tree structure and similar interactions are required. Hence, the design requirements on networking for these two structures includes the novel ORMRI communication paradigm and computation aggregation. For a chain structure, another requirement on networking is the chained computation, where request will be sent out or forwarded to the next edge device together with the computation results after completing the computations at the current node.

Therefore, the design requirements on networking to support distributed machine learning can be summarized as follows:
- ORMRI
- Computation aggregation
- Chained computation.

These requirements have not been well addressed in the literature as identified above.

The existing network technologies to provide communication services for distributed machine learning includes the end-to-end approach, TCP/IP, and the emerging networking technologies, such as ICN/NDN [7] and SDN [8].

With the end-to-end approach, users' requests and the available computation resources surrounding a user are dynamic and unpredictable, and there is no method to support the computation resource discovery. Thus, an additional mapping server to manage the edge devices with computation resources and map the computation request to suitable ECNs is necessary. That is, when a user wants to query local ECNs to perform targeted computations, it first consults this server, which monitors and manages the computation resources in local area. This server helps it to find out a set of suitable ECNs, and replies the related information of them, such as IP addresses, available resources, and access methods, to user. Then user sends out computation requests to those ECNs independently and retrieves computation results one by one through end-to-end communication protocol. It will be costly if the computation resources are highly dynamic, since the mapping server needs to realtime update the resource availability information for correctly matching requests with resources. Besides the cost to maintain the service of such mapping server, obviously end-to-end approach does not provide the chained computation and computation aggregation, which leads to large traffic for data and parameter exchange.

For the emerging technologies, ICN/NDN only supports the method to retrieve data or find function on one of the closest server(s) with a specific name in a fast way. Obviously, it also does not support ORMRI, computation aggregation, and chained computation. SDN is similar to the end-to-end approach, which also cannot satisfy those requirements.

Computation-Oriented Networking (CON)

To satisfy the design requirements on networking derived from distributed computations, we design a CON to enable users to efficiently discover the close computation resources and datasets to complete computation tasks. Especially, we propose a Hybrid Computation Oriented Routing (HCOR) mechanism combining proactive and reactive discoveries to find and request close edge computation devices to complete computation tasks collaboratively.

System Architecture

In CON, each CON node has a unique ID, and model and datasets are afforded with names, which can be used to request and achieve computation functions over specific model(s) and dataset(s). With those names, users can directly request the suitable ECNs to perform computations without relying on additional mapping server.

In CON, user sends out a Computation Request (CReq) specifying the name(s) of model(s) and dataset(s) to the network; CON nodes (routers) forward this CReq hop-by-hop to the close edge devices with computation resources and the requested datasets, which perform the corresponding computations. If logic structure is tree or peer-to-peer, the computation result will be replied to users along the reverse path forwarding CReq. The intermediate routers may aggregate computation results according to the setting on field of LocalAgg in the CReq. If the logic structure is chain, edge device will sequentially pop out model name and dataset name. Then computation results at this node will be included in CReq and forwarded to the next potential edge devices with computation resources and datasets. In addition, we design alive monitoring function to monitor computation status. With it, computation status request (CStatR) can be utilized to query computation status when waiting for the computation results reply (CResR).

In CON, HCOR mechanism is proposed, which combines proactive and reactive routing designs. One user wants to discover the surrounding edge devices and enables them to collaboratively complete a computation task together, which might have a logic structure of tree, peer-to-peer, or chain. Users acquire those services through the HCOR mechanism. If an edge device can continuously provide computation

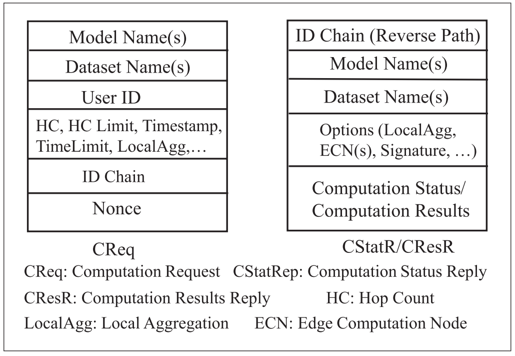

Model Name(s)
Dataset Name(s)
User ID
HC, HC Limit, Timestamp, TimeLimit, LocalAgg,…
ID Chain
Nonce
CReq
ID Chain (Reverse Path)
Model Name(s)
Dataset Name(s)
Options (LocalAgg, ECN(s), Signature, …)
Computation Status/ Computation Results
CStatR/CResR
CReq: Computation Request
CStatRep: Computation Status Reply
CResR: Computation Results Reply
HC: Hop Count
LocalAgg: Local Aggregation
ECN: Edge Computation Node

FIGURE 2. Packet types.

service, proactive routing approach is adopted, where routing entry will be pre-installed through announcements, such as link state method. If an edge device is equipped with dynamic resources, reactive routing approach will be utilized, where the resources will be discovered hop-by-hop through flooding the CReq in local area.

For proactive routing, each CON node (router) can become aware of the other nodes and the edge devices with stable resources in its neighborhood by the use of several techniques, such as local broadcasts known as hello messages. The edge devices with stable computation resources and datasets announce their existences to the network; and the paths to acquire them are registered at the routers' Computation Request Forwarding Tables (CRFTs) beforehand. CRFT is the forwarding information table designed to realize the ORMRI communication paradigm.

For edge devices with dynamic computation resources and datasets, they may provide resources during discontinuous time interval(s) for many reasons, such as the instability of network connection, and the willingness on the provision. Those resources will not be recorded at the CRFTs beforehand. For those devices, routers do not have to discover and maintain a route to them until one user is desired to achieve their services. They will be discovered through reactive routing mechanism.

As described previously, there are three types of packets, CReq, CStatR and CResR in CON as described in Fig. 2. It contains the fields of model name(s) and dataset name(s) to directly route the CReq to the potential multiple ECNs. User ID is used to return CStatR and CResR back to requestor. Besides, it holds the fields of Hop Count (HC) and HC Limit to restrict the scope of discovery, timestamp and timelimit to show the limitation on computation time, and LocalAgg to illustrate whether to enable the local aggregation or not. ID chain is included to record the path to edge devices, which can be utilized to return CStatR and CResR to user through the same path forwarding CReq.

Regarding the CStatR and CResR, besides the model names, dataset names, and computation results, the names of the ECNs and their signatures are optionally included to trace the computation responsibility of ECNs. In CON, the management of the resource can be potentially realized through the blockchain technology [15], though it will not be included for resource discovery.

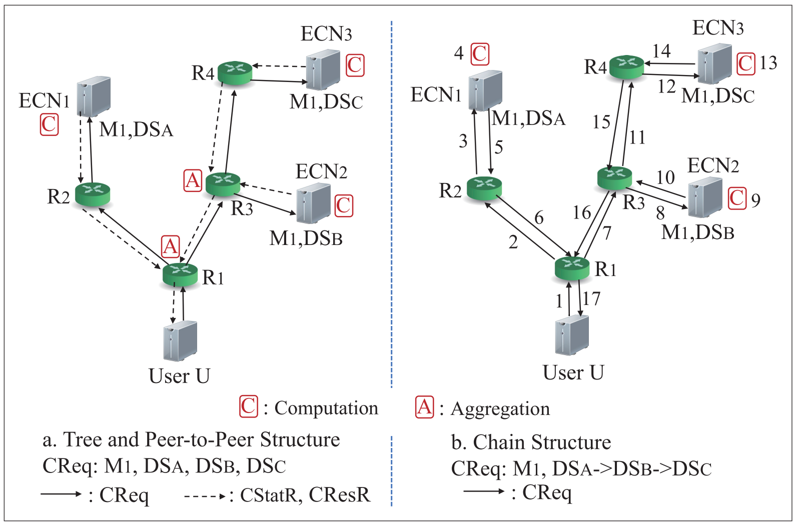
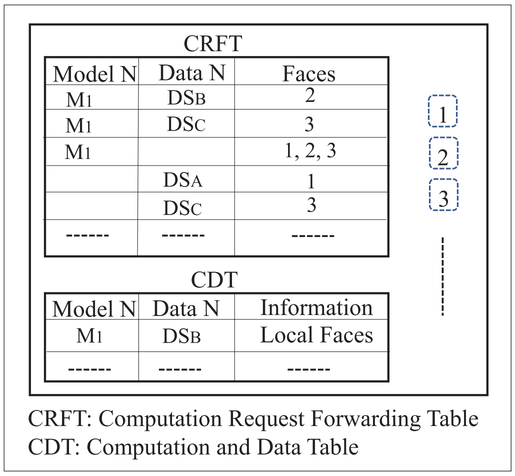


FIGURE 3. Packet flow.



CRFT

Model N	Data N	Faces
M1	DSB	2
M1	DSC	3
M1		1, 2, 3
	DSA	1
	DSC	3
------	------	------

CDT

Model N	Data N	Information
M1	DSB	Local Faces
------	------	------

CRFT: Computation Request Forwarding Table
CDT: Computation and Data Table

FIGURE 4. CON node.

Hybrid Computation Oriented Routing (HCOR)

The HCOR is composed of two processes, computation and dataset discovery process and CStatR and CResR forwarding process.

The computation and dataset discovery process is initiated whenever a user needs to request the close ECNs to collaboratively complete a computation task with model M and dataset DS, for which it may or may not have routing record in its CRFT.

User initiates Computation and Dataset discovery by broadcasting a CReq to its neighbors. User firstly sends the CReq to request the computation with specifying model and dataset names. Routers check CRFT and forward CReq based on the matching entry in CRFT if there is. Otherwise, CReq will be forwarded to all potential faces. Also this CON node (router) ID will be recorded at the CReq for tracing back to user.

If a CON node (edge device) holds the requested model and dataset and have enough computation resources, it invokes the corresponding computation to process this CReq locally. At the same time, this CON node gets the path from the requesting user to this edge device. It also registers itself with an entry in the CRFT as being in the process of computation. This entry is associated with a timeout using the TimeLimit as specified the CReq. Further it will respond CReq with the computation status information, such as the progress of computation, through the reverse path of forwarding CReq.

CStatR and CResR forwarding process is based on the reverse path recorded at the received CReq, which will be kept at ID chain (reverse path) fields of CStatR/CResR packets. The CON nodes (routers) forward the CStatR/CResR packets according to the CON node ID listed at the packet until it reaches user. After user receives those packets, it also knows the path from itself to the edge devices that perform computations.

Take Fig. 3 as an example to show the packet flow for sending CReq and receiving CStatR and CResR. Figure 3a illustrates such processes for tree and peer-to-peer structures. User U sends out the CReq, which is forwarded by R_1, further R_2, R_3, R_4. Finally, the CReq reaches ECN_1, ECN_2, and ECN_3, where the computations are performed. The computation results are returned from them, aggregated at R_3 and R_1, respectively, and the final result is returned to U. For the chain structure in Fig. 3b, the CReq is first sent to ECN_1, which performs computations. Then the CReq with the result will be forwarded to ECN_2, and ECN_3 sequentially for computation aggregation.

During the computation process, users can check the computation status periodically or per-request through the alive monitoring function. It sends out CReq with specifying the path to the destination edge devices to confirm computation status. This CReq packet for confirming status is forwarded based on the node ID list specified at CReq packet instead of model name(s) and dataset name(s). If a CON node is the node conducting computation, it will reply the current status of computation to user. For reliability of the CON, the alive monitoring function is used to find the abnormal node that suddenly fails on performing computation. If users receive the failure information, they will re-issue the CReq and request the computation again.

When computation is completed by edge device, the computation results will be returned according to the pre-specified reverse path and the CResR is forwarded hop-by-hop to user. If the time costed reaches Timelimit while computation is still in progress, it will stop the computation in progress. If user does not further require the computation over one specific edge device, it will send CReq with specifying a new TimeLimit to this node. After receiving this CReq, the serving edge device finds the current time exceeds TimeLimit and stops the corresponding computation.

Node Design

In CON, node's ID can be an IP address or an identifier. The names of models and datasets are hierarchical ones with semantic meanings, which are utilized to request the corresponding computations over specific datasets. Take /CNN/DS_A/DS_B/DS_C as the example. It is to train a Convolutional Neural Network (CNN) model over the datasets, DS_A, DS_B, and DS_C.

As in Fig. 4, a CON node (router) mainly has two data structures: CRFT and Computation and Data Table (CDT).

The CRFT is used to forward the CReq packets toward the potential ECNs. It allows a list of outgoing faces instead of only one face. Besides, it has two matching process for each entry: model name and dataset names and exact match is utilized to match the names in CReq and the ones in CRFT. The CDT shows the local computation resources and local dataset that can be provided by this CON node. If the requested model and dataset can be found in local storage/cache, the corresponding computation will be performed locally at this CON node.

Based on the designs of packet and CON node, the basic operations of CON are as follows. When a user wants to discover the surrounding ECNs to ask them to perform computations, it will send out the CReq. The intermediate nodes check whether the local resource can be used for the requested computation through matching the names in CReq with the names in CDT and checking whether the local computation is enough. If it is, local computation will be performed; the node ID will be recorded; and the CStatR will be replied. Otherwise, it will increase the HC with 1, and then continue to forward the CReq to the faces listed in the matching entry or default entry in CRFT, where default entry is used for reactive forwarding. This process continues until it reaches HC limit or all model and dataset names are found.

Performance Evaluations

Herein we investigate how the proposed CON can utilize computation aggregation to reduce the computation result transmission cost compared to the end-to-end approach. The critical performance indicator is the bandwidth consumption reduction ratio (BCRR), which is defined as the total amount of reduced bandwidth (the amount of bandwidth consumption using end-to-end approach – the amount of bandwidth consumption using CON) over the total amount of bandwidth consumed using end-to-end approach.

The network topology under investigation is assumed to be a full and complete tree topology characterized by degree and depth. The root of a tree is the user who sends out CReq; the leaf nodes are the edge devices that conduct computations; and the intermediate nodes are the routers. Each node is either a leaf or possesses the exact degree number of child nodes.

We let each intermediate nodes (routers) to aggregate the computation results obtained from the upstream routers or leaves. We vary the degree from 2 to 10 and the depth from 1 to 10, and obtain the BCRR as Fig. 5.

In Fig. 5, we can observe that the BCRR increases as the depth increases and also as the degree increases. For the situation with both degree and depth as 2, the bandwidth consumption will be reduced with 25 percent by CON, which is the lowest value of BCRR. It means the bandwidth consumption will be reduced with at least 25 percent. In Fig. 5, it can be also observed that bandwidth consumption will be dramatically reduced with 89 percent for the situation with both degree and depth as 10. We can see that the proposed CON can reduce the bandwidth consumption greatly through computation aggregation compared to the end-to-end approach.

To validate such analysis, we also implement the CON main function of forwarding and experiment the same tree topology using Ubuntu docker containers in local environment. The workstation for experiments holds Intel Xeon 2 CPU with Gold 6132, 3.700GHz 24core and 2 GPUs of NVIDIA Quadro RTX 5000 and the operation system is Ubuntu 18.04.6 LTS. For the computation, we employ the simple linear regression with the aggregation as averaging the model parameters. We set the maximum number of Docker containers as 45, which reaches the capacity limit of the workstation. The experiment results match the above analysis result, which is proved to be trustworthy.

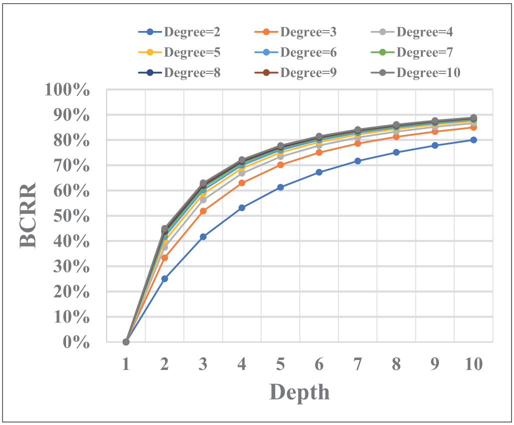


FIGURE 5. Bandwidth consumption reduction ratio by CON.

Conclusions and Future Directions

In this article, we identify the design requirements on networking toward the efficient edge intelligence as ORMRI, computation aggregation, and chained computation. To satisfy those requirements, we design a computation-oriented network (CON), where a hybrid computation oriented routing mechanism combining proactive and reactive routing approaches has been proposed. Through the analysis and experiments, we observed that the bandwidth consumption can be greatly reduced with at least 25 percent compared to the end-to-end approach. For the future directions, there are several open challenges, for example, how to match the computation tasks and the available resources; how to differentiate the trustworthy nodes from the untrusted ones; how to assure the security for such decentralized computations.

Acknowledgment

This work was partly supported by JSPS KAKENHI Grant Number 19H04105 and Support Center for Advanced Telecommunications Technology Research (SCAT).

References

[1] R. Li *et al.*, "Achieving High Throughput for Heterogeneous Networks with Consecutive Caching and Adaptive Retrieval," *IEEE Trans. Network Science and Engineering*, vol. 7, issue 4, Oct.-Dec. 2020.

[2] J. Verbraeken *et al.*, "A Survey on Distributed Machine Learning: State-of-the-Art and Research Challenges," *ACM Computing Surveys*, vol. 53, no. 2, Mar. 2021, pp. 1–33.

[3] S. Deng *et al.*, "Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence," *IEEE Internet of Things J.*, vol. 7, no. 8, 2020.

[4] J. Tao, T. Han, and R. Li, "Deep Reinforcement Learning Based Scheme for Intrusion Detection in Aerial Computing

Networks," *IEEE Network*, vol. 35, no. 4, July/Aug. 2021.
[5] IRTF Computing in the Network Research Group (COINrg), https: //datatracker.ietf.org/rg/coinrg/; accessed Feb. 28, 2022.
[6] IEEE SIG on Intelligent Internet Edge (IIE), https: //github.com/IEEEITCIIE/Home/wiki; accessed Feb. 28, 2022.
[7] V. Jacobson *et al.*, "Networking Named Content," *Proc. 5th Int'l Conf. Emerging Networking Experiments and Technologies (ACM CONEXT)*, Rome, Italy, 2009, pp. 1–12.
[8] K. Kirkpatrick, "Software-Defined Networking," *Commun. ACM*, vol. 56, no. 9, Sept. 2013, pp. 16–19.
[9] Y. Zhao *et al.*, "Stability-Based Analysis and Defense against Backdoor Attacks on Edge Computing Services," *IEEE Network*, vol. 35, no. 1, Jan./Feb. 2021.
[10] Q. Yang *et al.*, "Federated Machine Learning: Concept and Applications," *ACM Trans. Intelligent Systems and Technology*, vol. 10, no. 2, Mar. 2019, pp. 1–19.
[11] X. Dong *et al.*, "A Survey on Ensemble Learning," *Frontiers of Computer Science*, vol. 14, 2020, pp. 241–58.
[12] Y. Atsumi *et* al., "A Route Establishment Method in Activenet and Enhancement of ANTS for Next Generation Internet (in Japanese)," *J. Information Processing*, vol. 42, no. 12, Dec. 2001, pp. 2897–2908.
[13] R. Bostrom, S. Gupta, and J. Hill, "Peer-to-Peer Technology in Collaborative Learning Networks: Applications and Research Issues," *Int'l. J. Knowledge and Learning*, vol. 4, no. 1, 2008.
[14] K. Weiss, T. Khoshgoftaar, and D. Wang, "A Survey of Transfer Learning," *J. Big Data*, vol. 3, no. 9, 2016.
[15] R. Li *et al.*, "A Blockchain-Based Data Lifecycle Protection Framework for Information-Centric Network," *IEEE Commun. Mag.*, vol. 57, no. 6, June 2019, pp. 20–25.

Biographies

Ruidong Li [SM] (liruidong@ieee.org) is an associate professor at Kanazawa University, Japan. He is the recipient of the best paper awards for IEEE ICC 2022 and IWCMC 2022. He serves as the vice chair of IEEE Internet Technical Committee (ITC), and served as chairs for several conferences, such as the general co-chairs for MOBILWare 2022, IEEE MSN 2021, CPSCom 2021, and TPC co-chairs for IWQoS 2021, IEEE MSN 2020, BRAINS 2020. He is a member of ACM and IEICE.

Tomoki Hirayama is currently an iOS Developer at Sansan, Inc. He received a bachelor degree from Kanazawa University in Mar. 2022 after receiving an associate degree from Salesian Polytechnic (Tokyo Salesian College of Technology). He is an expert in iOS application with UI/UX, back-end service, and machine learning programs.

Kaiping Xue [SM] is a professor in the School of Cyber Security, University of Science and Technology of China (USTC). His research interests include next generation Internet architecture design, transmission optimization and network security. He serves on the Editorial Board of several journals, including the IEEE TDSC, IEEE TWC, and IEEE TNSM. He is an IET Fellow.

Peiying Ruan [M] is currently a senior data scientist at NVIDIA for deep learning in healthcare and life science. She received M.Sc. and Ph.D. degrees in Informatics from Kyoto University in 2012 and 2015, respectively. Before joining NVIDIA in 2017, she was a postdoctoral fellow at the National Institute of Advanced Industrial Science and Technology (AIST), Japan. Her research interests include machine learning, bioinformatics, and medical imaging. She is a member of RSNA.

Hitoshi Asaeda [SM] is a director of Network Architecture Laboratory, National Institute of Information and Communications Technology (NICT). He holds a Ph.D. from Keio University. He was previously with IBM Japan, Ltd., Inria, France, and a project associate professor at Keio University. He was a general co-chair of IEEE/ACM IWQoS 2021 and ACM ICN 2022. He is a senior member of IEICE, and a member of ACM.