

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 26 April 2022

Z. Li
K. Yao
Y. Li
China Mobile
23 October 2021

A Compute Resources Oriented Scheduling Mechanism based on Dataplane
Programmability
draft-li-coinrg-compute-resource-scheduling-00

Abstract

With massive data growing in the internet, how to effectively use the compute resources has become a quite hot topic. In order to cool down the pressure in today's large data centers, some compute resources have been moved towards the edge, gradually forming a distributed Compute Force Network. Force is a physical cause which can change the state of a motion or an object. We refer the definition from physics and extend its philosophy to network that in future, the network can be a compute force which can facilitate the integration of different kinds of compute resources, no matter hardware or software, making the computation fast and effective. In this draft, we present a compute resources oriented scheduling mechanism based on dataplane programmability, which can effectively schedule and manage compute resources in the network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 26 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

Internet-Draft

Network Working Group

October 2021

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Conventions Used in This Document	3
2.1.	Terminology	3
2.2.	Requirements Language	3
3.	Design	3
3.1.	Network Topology	3
3.2.	Mechanism Statement	5
4.	Typical Way of Realization	7
5.	Security Considerations	8
6.	IANA Considerations	8
7.	Normative References	8
	Authors' Addresses	8

[1.](#) Introduction

As Moore's law has been gradually reaching its limitation, the computation of massive data and diverse computational requirements can not be satisfied by simply upgrading the computation resources on a single chip. There become an emerging trend that domain specific computation resources like GPU, DPU and programmable switches are becoming more and more popular, generating diverse use cases in the network. For example, in network computing and in memory computing. In network computing means using programmable switches or DPUs to offload network functions so as to accelerate network speed. And in memory computing means that the computer memory does not only serve as the storage, but also provide the computation. With the development of these domain specific architectures, network should serve as a force which could facilitate the integration of all these different types of computation resources, in turn forming a Compute Force Network. In CFN, how to effectively schedule these computation resources is a topic that's worthy of studying.

Current ways to do compute resources allocation include extending protocols like DNS so as to realize the awareness and scheduling of compute resources, but the management of these compute resources must be done in the centralized controller. a DNS client wants to do some computing tasks, e.g. Machine learning models training, and the

client will send a request to DNS server. Then, DNS server will inform the client which compute node is available at the moment. However, activating and deactivate this compute node to work, e.g. creating a virtual machine, is done by centralized controller, which we think is not very efficient and timely, considering massive data waits to be computed in the network. The weakness above has provoked an idea to realize the scheduling and management of compute resources by extending current routing protocols like SRv6 with the help of programmable network elements. The detailed design is presented in this draft.

[2.](#) Conventions Used in This Document

[2.1.](#) Terminology

CFN Compute Force Network

DNS Domain Name Service

SRv6 Segment Routing over IPv6

GPU Graphics Processing Unit

DPU Data Processing Unit

[2.2.](#) Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14](#)[\[RFC2119\]](#)[\[RFC8174\]](#) when, and only when, they appear in all capitals, as shown here.

[3.](#) Design

The detailed design of the mechanism is presented in this section. A

typical topology will be shown below and the definition of each part of the network topology will be given, and then the whole procedure will be explained clearly the second subsection.

3.1. Network Topology

The network topology is shown in figure below where there are several major parts inside, namely consumer, computation manament node, compute node with programmable DPU, and programmable network element.

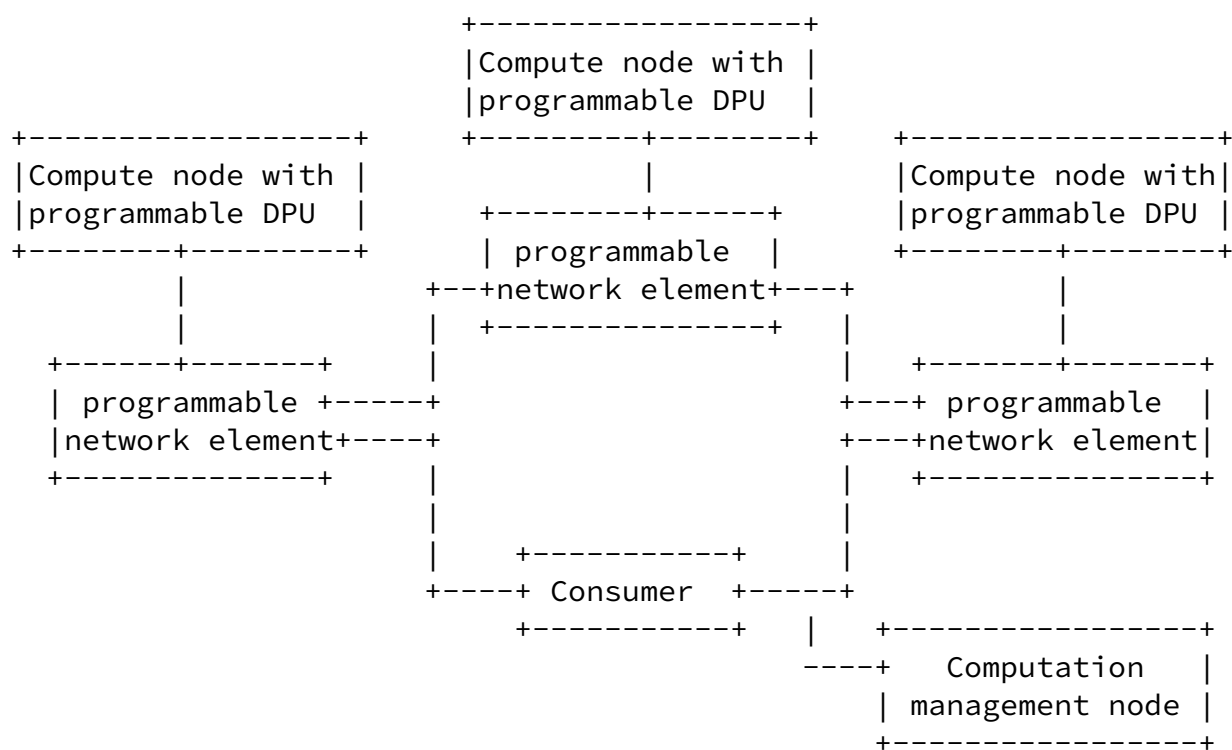


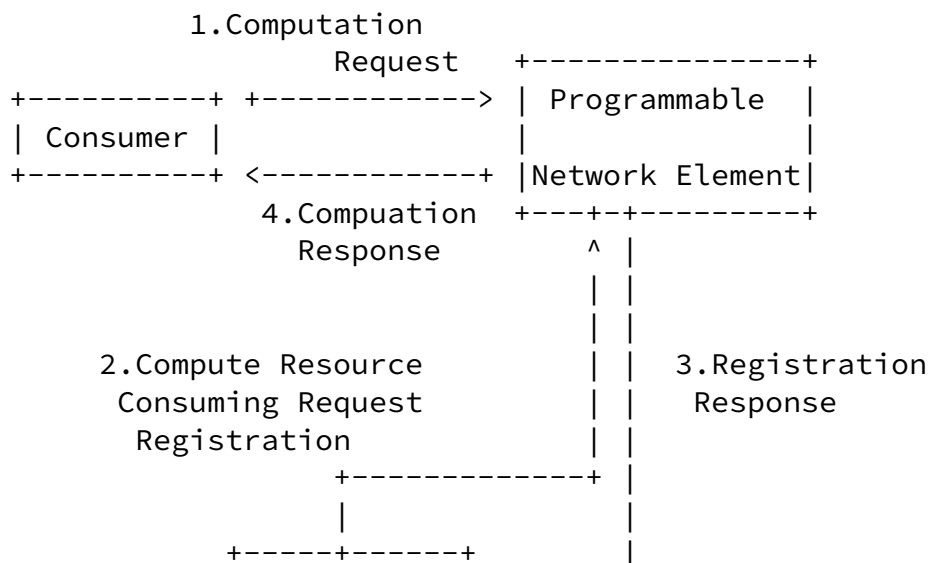
Figure 1: Figure 1: Network Topology

- Consumer: End node generating computing tasks which need to be done by compute resources
- Compute node: A network node that has the resources to finish computing tasks generated by consumers, e.g. a server or a cluster of servers.

- Programmable DPU: An unit that is connected to a compute node and a programmable element, responsible for the lifetime management of compute node and the communication with programmable element.
- Programmable network element: A network device which communicates with customers and programmable DPU, forwarding messages bidirectionally including requests for computing resources, activating or deactivating specific compute resource, and other routing messages.
- Computation management node: A network node that has the full view of the computation resources in the network, dynamically managing these resources and generate consuming receipt.

[3.2.](#) Mechanism Statement

In this section, the detailed procedure of the communication between the consumer and the compute management node which passes through programmable DPU, programmable network element, and compute node will be declared step by step .



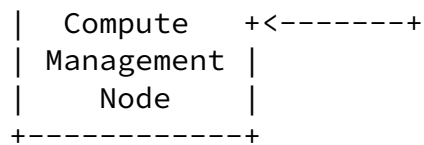


Figure 2: Figure 2: Computation Request Procedure

* Step1: computation request registration. When a consumer wants to do some computing tasks, e.g. machine learning model training, it first needs to send a request message to the compute management node for computation resource pre-allocation. The message is passed through programmable network element where some modification on the packet header can be done on the dataplane. Information like computation category, configuration template can be added into packet header, which could notify the compute management node that what kind of computation resource it needs to shedule,e.g. how many GPUs are needed in the task. Afterwards, The management node will send back a message in which the specific computation node IP address is inserted. If no such comptation node is available at the moment, the manament node will send back a refusal. And at last, the programmable network element will forward the message to the consumer.

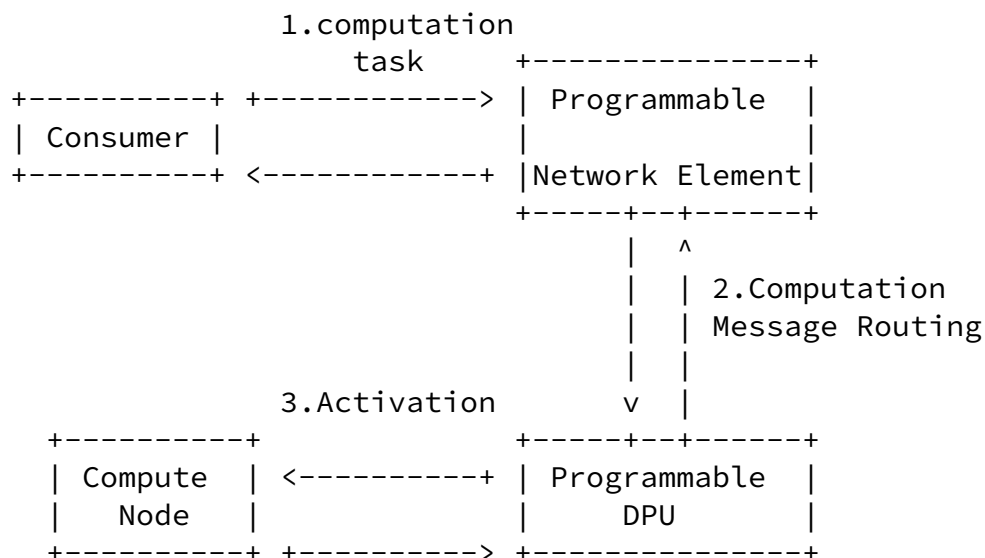
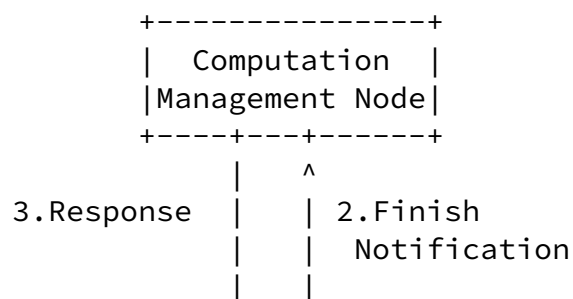


Figure 3: Figure 3: Computation Activation

* Step 2:Computation activation. Consumer will send the actual computation task to programmable network element which will do some modification on the packet. The activation message of the compute node will be encapsulated into the packet which could enable the lifetime management of the computation and the working progress of the compute node. And then, the message will be forwarded to the programmable DPU directly connected to the compute node where the decapsulation of the packet will be done. The DPU will tell the compute node to work and dynamically monitor the state of the compute node until the task is finished.



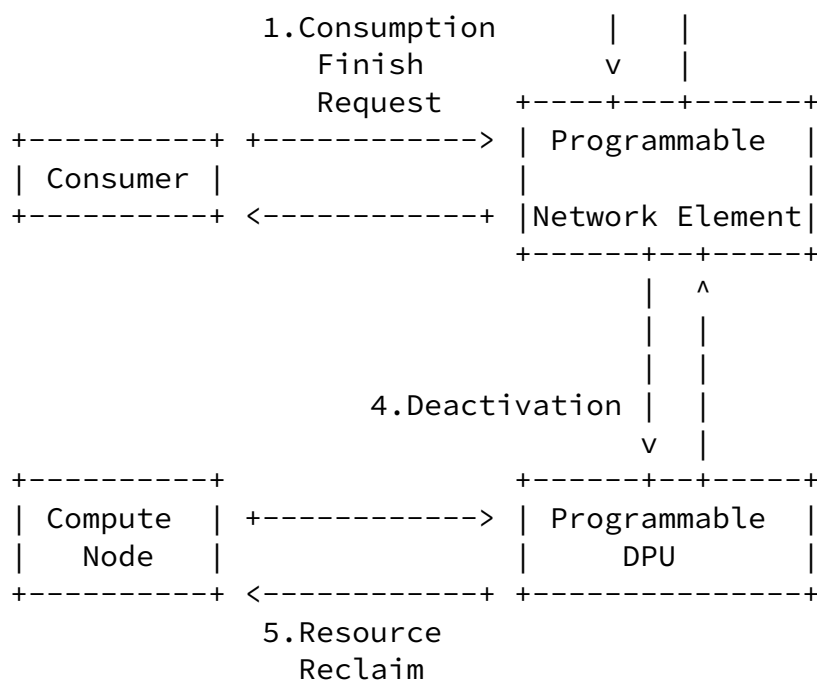


Figure 4: Figure 4: Consumption Finish

* Step 3: When the compute node notify the consumer that the task has been finished, the consumer will decide whether there is any waiting task, if not, the consumer will send a consumption finish request to the computation management node. Like computation request registration, the programmable network element will then insert information of the compute node and forward the notification message to the computation management node. when the programmable network element receives a response message, it will start deactivation procedure and tell the compute node to collect back the resource used for previous computation. This is the end the lifetime of computation of a single task.

4. Typical Way of Realization

The mechanism stated in above section can be realized by extending protocols like SRv6. The lifetime management message can be inserted dynamically in dataplane with the help of those programmable hardware. Such modification can be done flexibly and in line rate.

5. Security Considerations

TBD.

6. IANA Considerations

TBD.

7. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Authors' Addresses

Zhiqiang Li
China Mobile
Beijing
100053
China

Email: lizhiqiangyjy@chinamobile.com

Kehan Yao
China Mobile
Beijing
100053
China

Email: yaokehan@chinamobile.com

Yang Li
China Mobile
Beijing
100053
China

Email: liyangzn@chinamobile.com