Signaling In-Network Computing operations (SINC)

draft-zhou-sfc-sinc-00

David Lou, Luigi Iannone, Yujing Zhou, Zhangcuimin

RTGWG WG

IETF-115 : Nov 2022, London+Online

Motivation*

- Recent research has shown that **network devices undertaking some computing tasks** can greatly improve the overall network and application performance in some scenarios

- Their implementation is mainly based on the **programmable** network devices by using P4 or other languages.

- Also, for complex network topologies, such as DC, **traffic steering** is needed to route the packet to the programmable network devices.

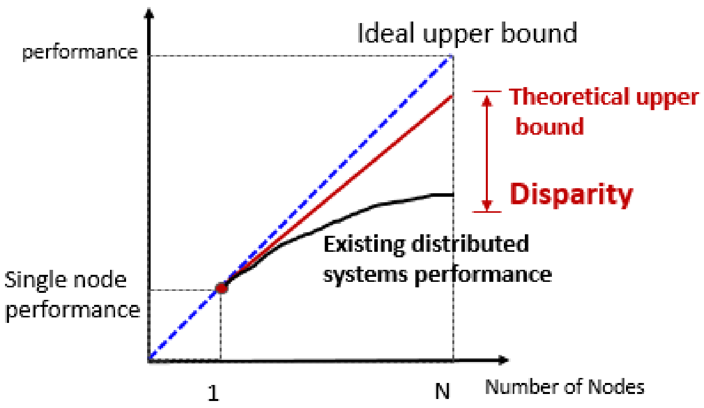
- We argue that for some use cases, it is useful to **provide an explicit and general way** in the data/control planes **to signal the in network computation.**

*More extensive overview of use cases presented at COINRG interim Sep. 2022
https://datatracker.ietf.org/doc/agenda-interim-2022-coinrg-03-coinrg-01/

SINC Use Case

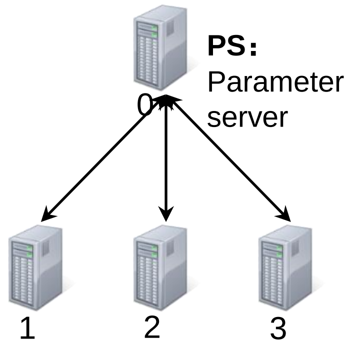
- Existing Weak points of distributed system:

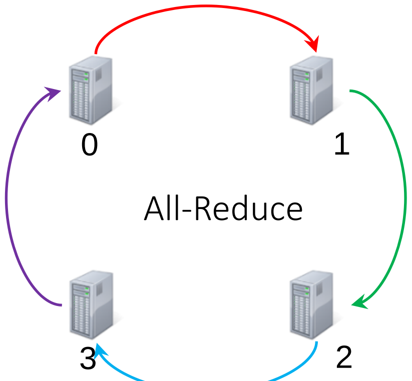
1) The nodes among distributed nodes will generate more traffic in order to reach consensus
2) CPU and GPU chip throughput is insufficient, therefore packet loss is caused



The increase in the number of servers does not lead to a linear increase in server performance.

Traditional way:

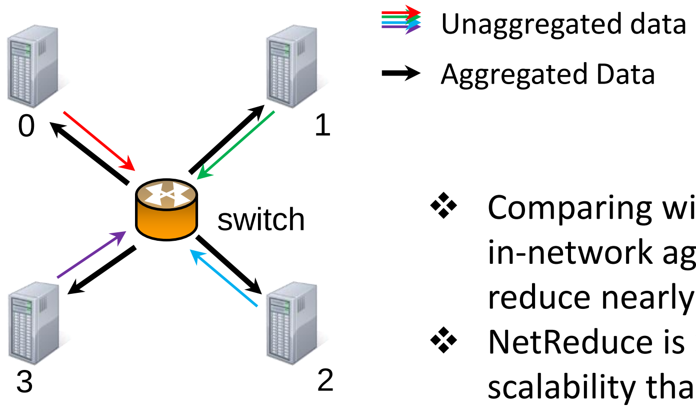




- ❖ PS aggregate the gradients, thus PS will suffer from in-cast issue.
- ❖ PS can easily become a bottleneck

- ❖ The overall data transfer is increasing;
- ❖ The communication pattern involved may lead to higher network latency

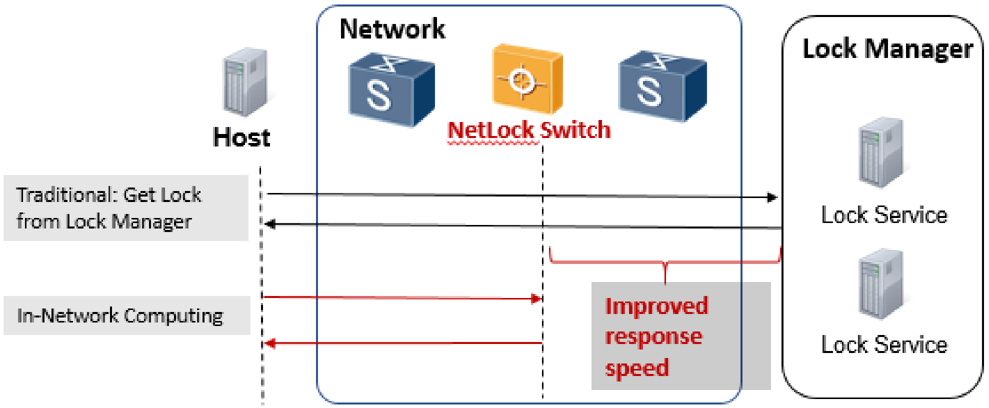
NetReduce:



- ❖ Comparing with the host oriented solutions, in-network aggregation could potentially reduce nearly half the aggregation data
- ❖ NetReduce is >1.5x faster, and has better scalability than ring all-reduce.

SINC Use Case

NetLock:

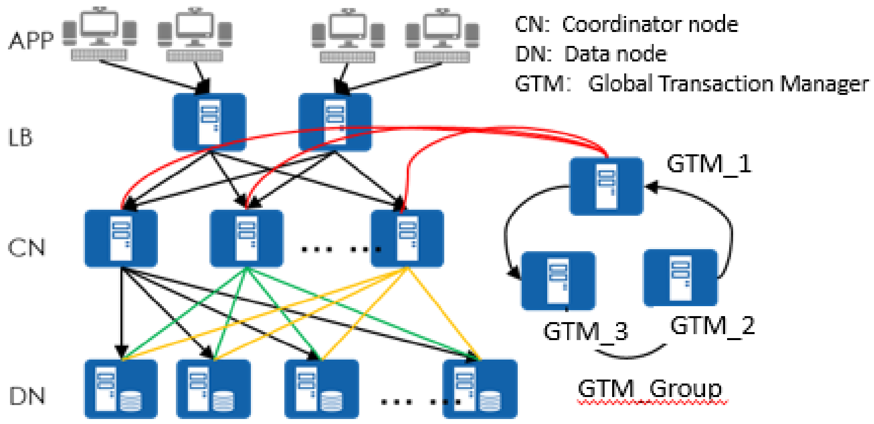


❖ For SINC :

The lock manager can be abstracted as **Compare And Swap (CAS) or Fetch-and-Add (FA) operations**.

The test results in NetLock[1] show that the lock manager running on a switch is able to answer 100 million requests per second, **nearly 10 times more than** what a lock server can do.

NetSequencer:



❖ For SINC:

Switches could realize the sequencer[2] by using a **"Fetch-and-Add (FA)" operation**.

Compared with Gbps-level throughput of servers, network devices have **Tbps-level throughput** and **line-rate processing capabilities**

[1] Yu Z, Zhang Y, Braverman V, et al. Netlock: Fast, centralized lock management using programmable switches[C]//Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. 2020: 126-138.
[2] Design Guidelines for High Performance RDMA Systems, https://www.usenix.org/conference/atc16/technical-sessions/presentation/kalia

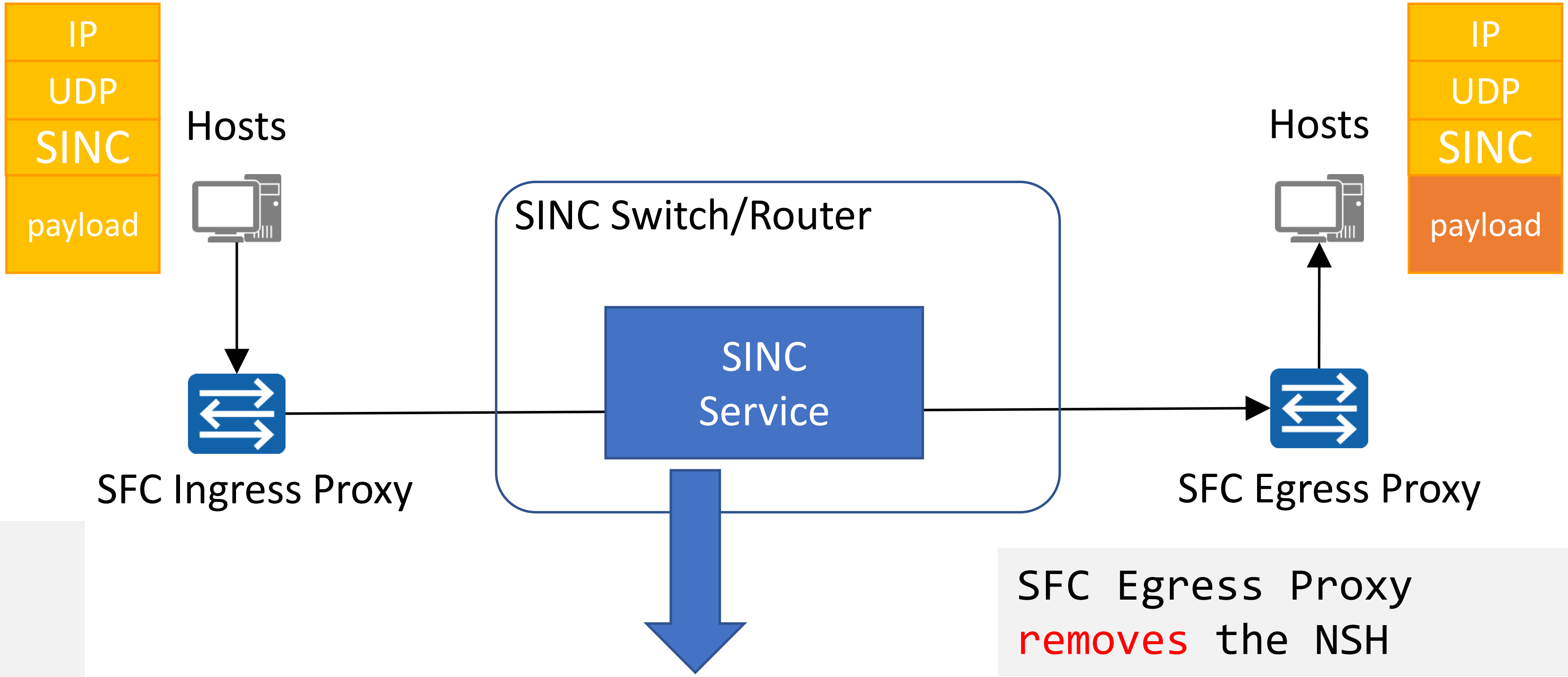
In-Network Operations and Data

- ❖ The core idea of SINC is to offload “bottleneck” computing operations to the network devices in order to improve the system performance
- ❖ The network devices executing computing operations should not affect the forwarding performance of data plane
- ❖ Generic "simple and basic" operators are desired to support different scenarios
- ❖ An explicit and general mechanism is required to tell the switch what, where and how

Use Case	Operation	Description
NetReduce	Sum value (SUM)	The network device sums the collected parameters together and outputs the result
NetLock	Compare And Swap or Fetch-and-Add (CAS or FA)	By comparing the request value with the status of its own lock, the network device sends out whether the host has the acquired lock. Through the CAS and FA, host can implement shared and exclusive lo cks.
NetSequencer	Fetch-and-Add (FA)	The network device offers a counter service and provides a monotonically increasing sequence number for the host.

SINC Overview

A host sends out packets containing data operations to be executed in the network



SFC Ingress Proxy encapsulates the packet with NSH

SFC Egress Proxy removes the NSH

SINC-capable switches/routers executes all or part of the data computation during the transmission

SINC Header

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved    |L|                 Group ID                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     No. of Data Sources       |     Data Source ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            SeqNum                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Data Operation          |     Data Offset               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

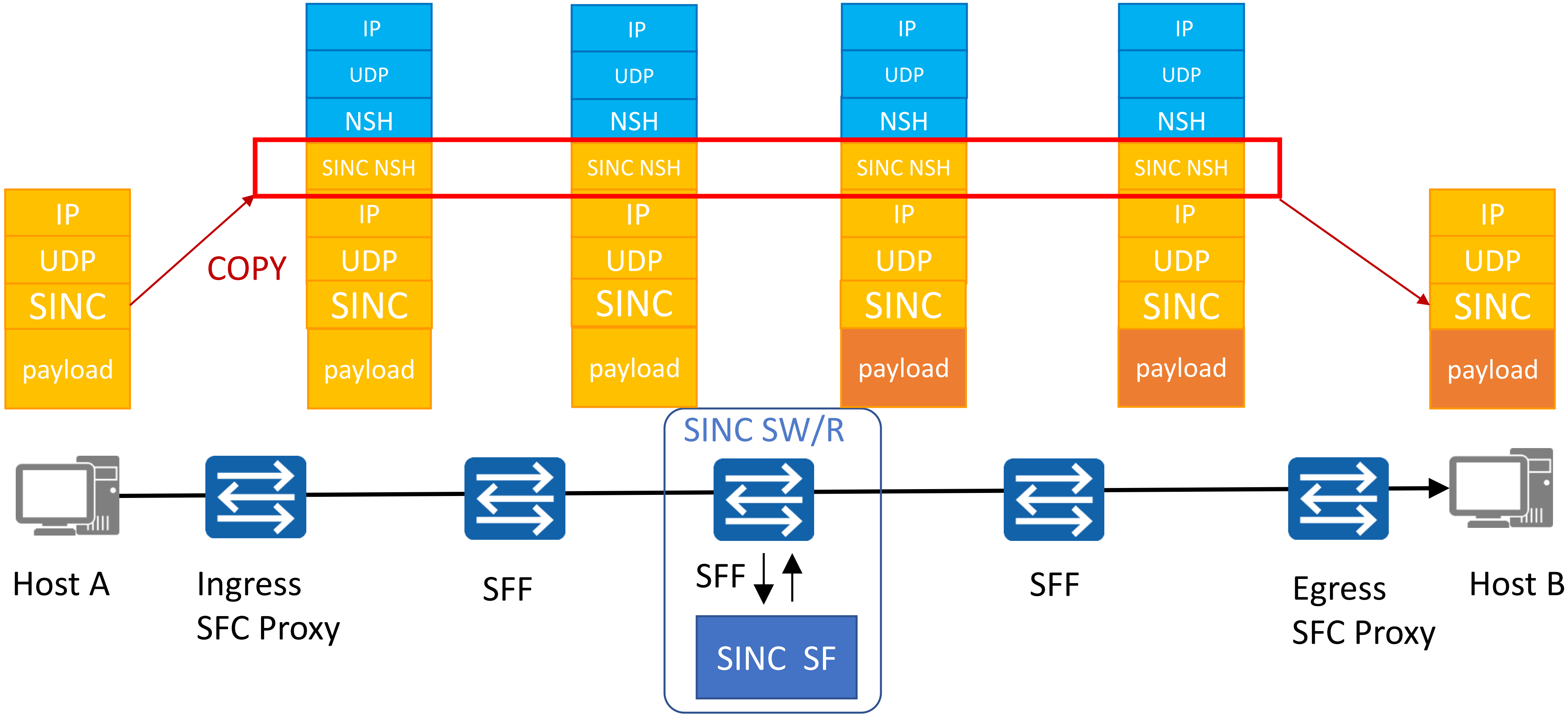
- **Loopback flag (L):**
  - Zero (0) -> be sent to the destination.
  - One (1) -> be sent back to the source node.
- **Group ID:** Identifies different groups
- **Number of Data Sources:** Total number of data source nodes that are part of the group.
- **Data Source ID:** Unique identifier of the data source node of the packet.
- **Sequence Number (SeqNum):**
  The SeqNum is used to identify different requests within one group.
- **Data Operation:** The operation to be performed, like ADD, SUM, MAX, MIN
- **Data Offset:** The in-packet offset from the SINC context header to the data required by the operation.

To **associate** and **forward** the message with to the right computing service

SFC for Signal In-Network Computing

SFC is one possible way to steer traffic to the right in-network SINC-capable switch where SINC is a Service Function (SF)

Case 1: hosts do support SINC



SINC NSH Encapsulation

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|ver|O|U|    TTL     |   Length  |U|U|U|U|MD Type| Next Protocol |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Service Path Identifier (SPI)        | Service Index |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          SINC Header                                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

NSH Base Header
MD type = 0x1

NSH Service Path Header

SINC Context Header

- **NSH Base Header:**
  - Use the NSH Meta Data (MD) fixed-length context headers to carry the data operation information
  - MD type = 0x4 was used in the draft because the size of the original design of the SINC header is not 16 bytes. It will be updated in the next version of the draft.
- **NSH Service Path Header:** as defined in RFC 8300.
- **SINC Context Header:** as defined SINC Header. SFC Proxy copy these information in SFC header.

Next Step

- Encourage discussion on the mailing lists of the RTG WG and SFC WG
- Update the draft based on comments and remarks
- Design the control plane with a possible separate draft
- Welcome to contributions and co-authors