Understanding RDMA Microarchitecture Resources for Performance Isolation

Xinhao Kong, Jingrong Chen, Wei Bai, Yechen Xu, Mahmoud Elhaddad,

Shachar Raindel, Jitendra Padhye, Alvin R. Lebeck, Danyang Zhuo

Duke
UNIVERSITY

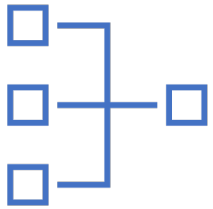
Rapid Adoption of RDMA in Clouds

Performance

Low latency

High throughput

High CPU efficiency



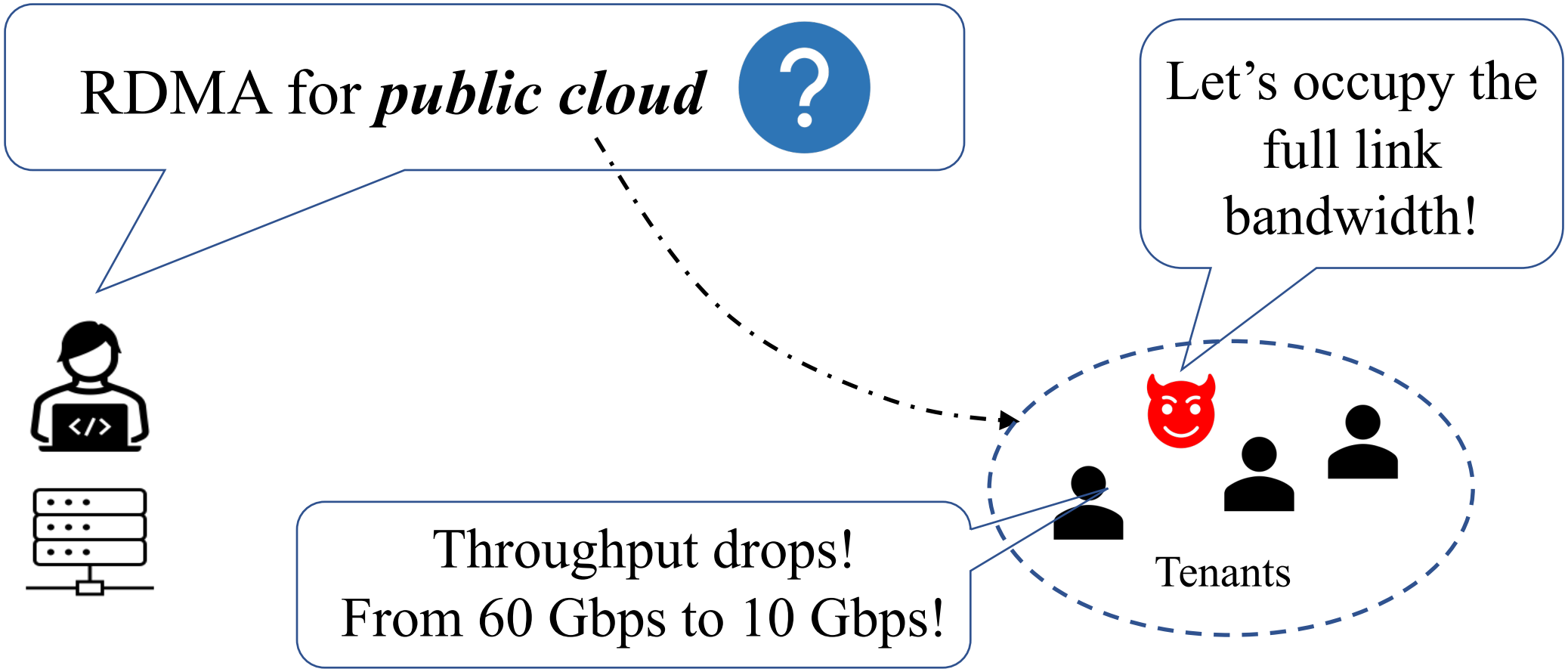
Various applications

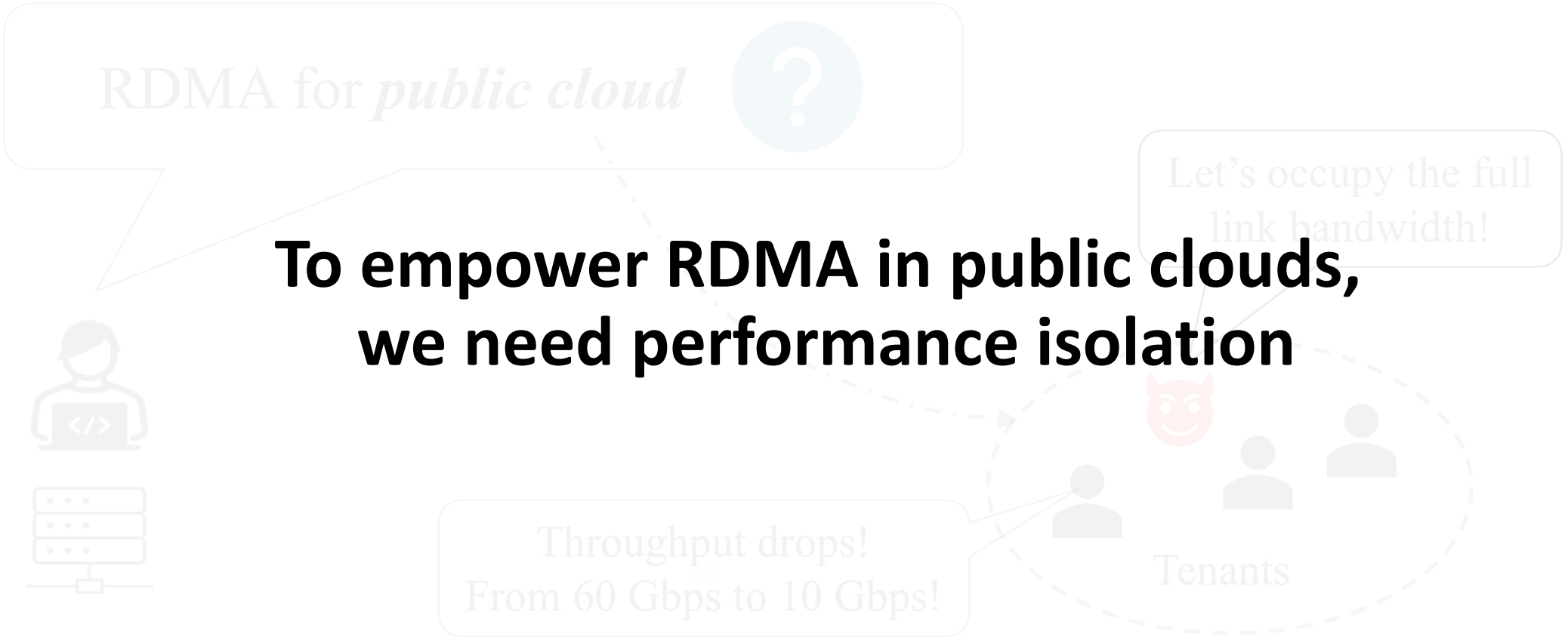
RDMA + RPC: mRPC (NSDI ’23), eRPC (NSDI ’19)

RDMA + Storage: iSCSI/NVMe over RDMA

RDMA + ML: GPU Direct RDMA

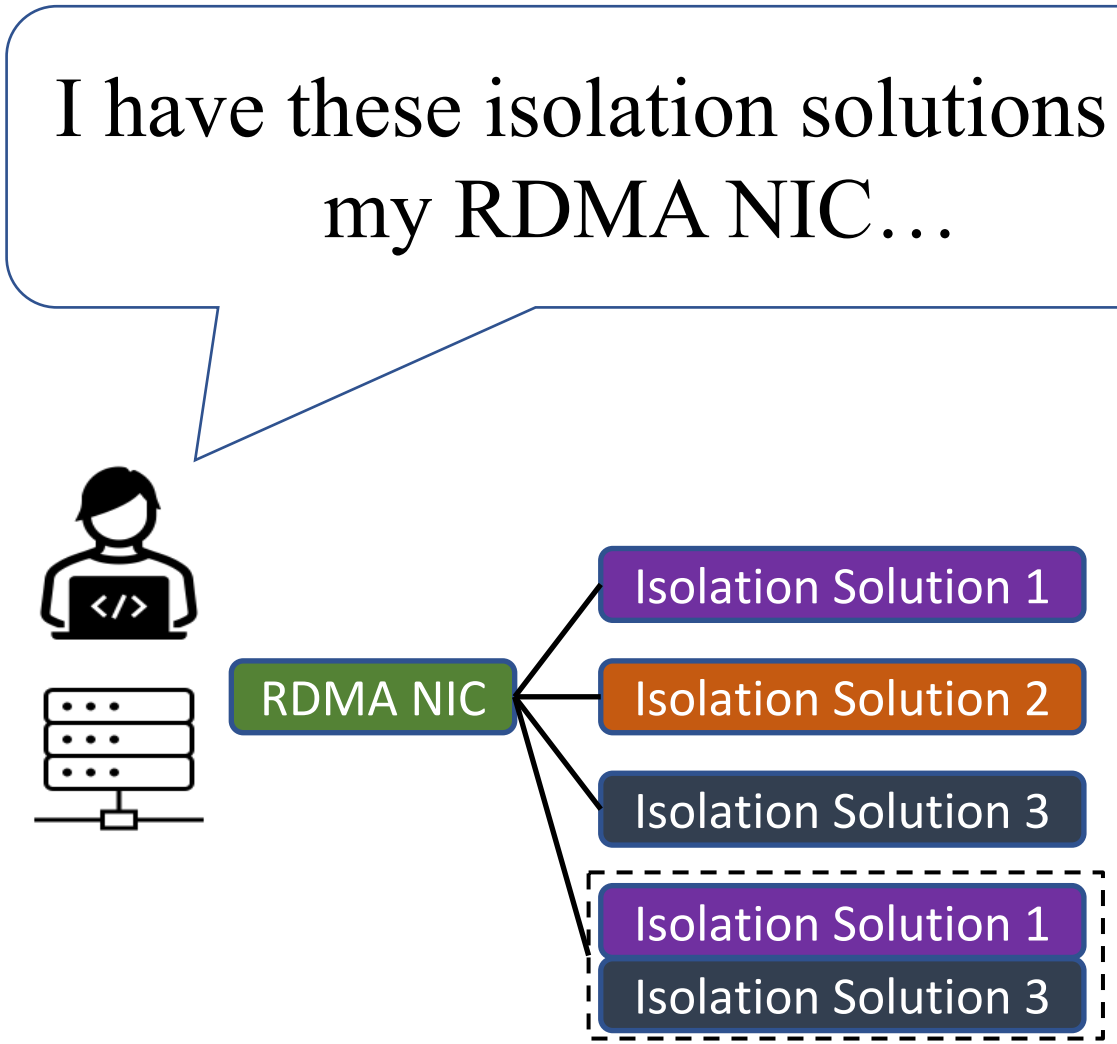
Deployment Challenges for RDMA in Public Clouds





To empower RDMA in public clouds, we need performance isolation

Existing Solutions for RDMA Performance Isolation



I want two tenants fairly share the 100 Gbps RDMA networks.

RDMA NIC — Isolation Solution 1 — RDMA NIC

How to Evaluate Isolation Solutions?

I want two tenants fairly share the 100 Gbps RDMA networks.

RDMA NIC — Isolation Solution 1 — RDMA NIC

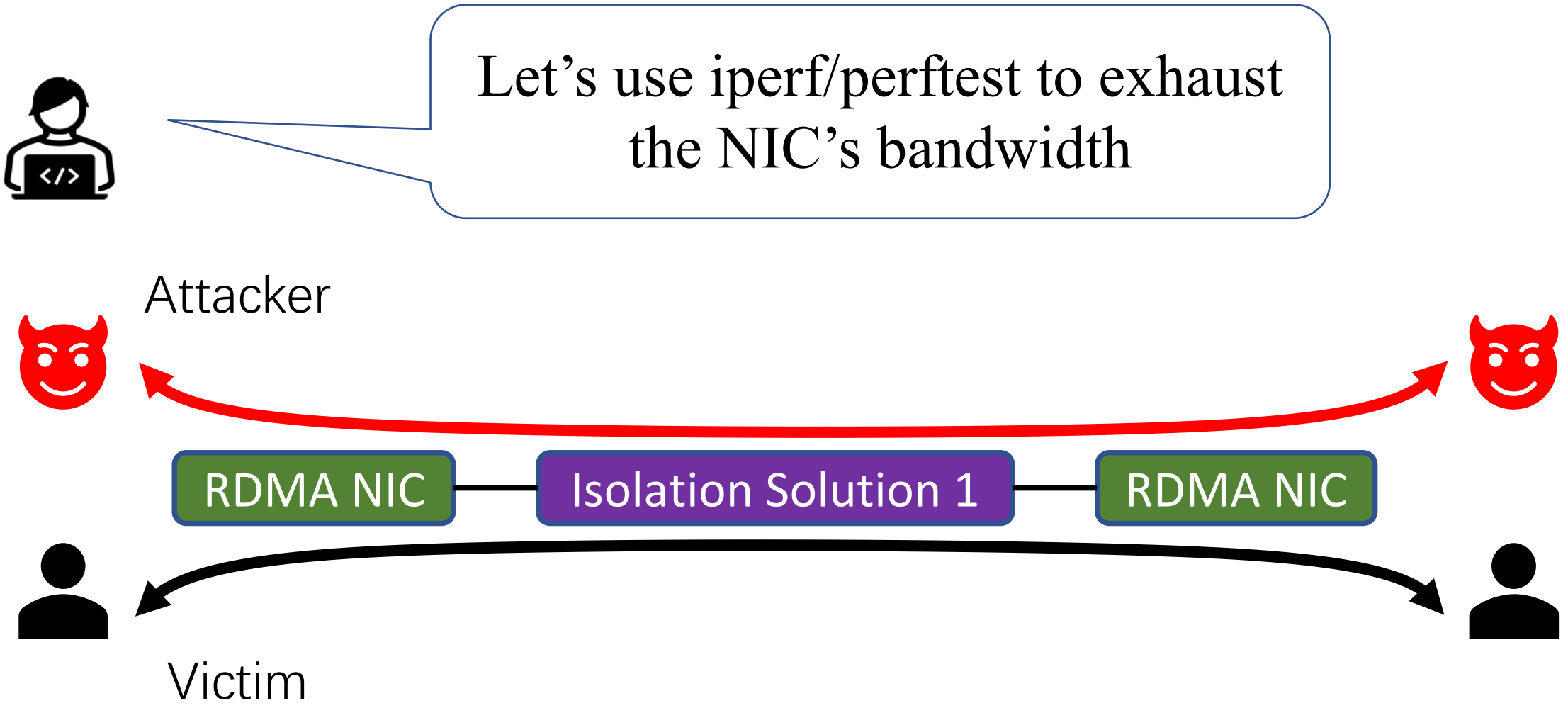
Evaluating RDMA Performance Isolation

Existing solutions uses traffic generators as attackers to stress the NIC

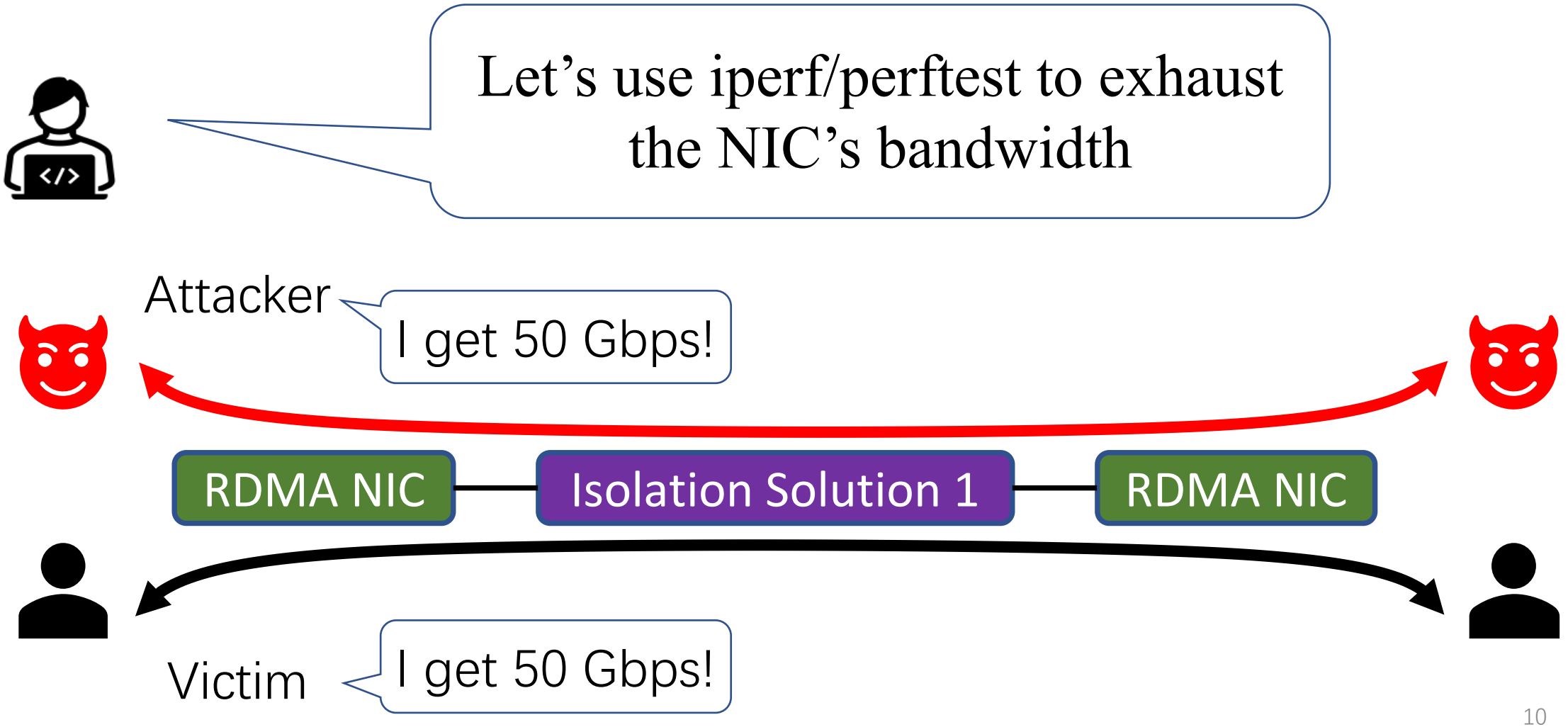
Attacker

Victim

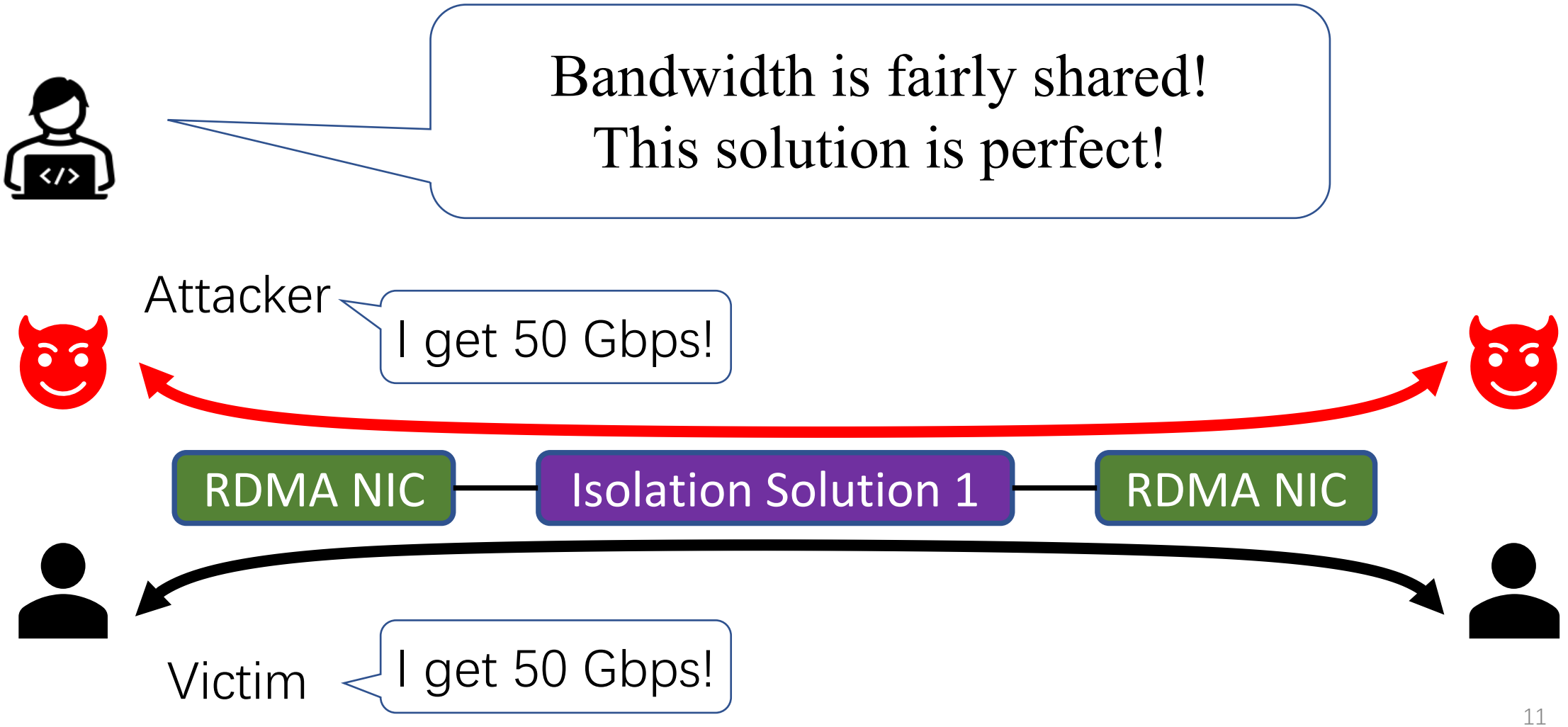
Evaluating RDMA Performance Isolation



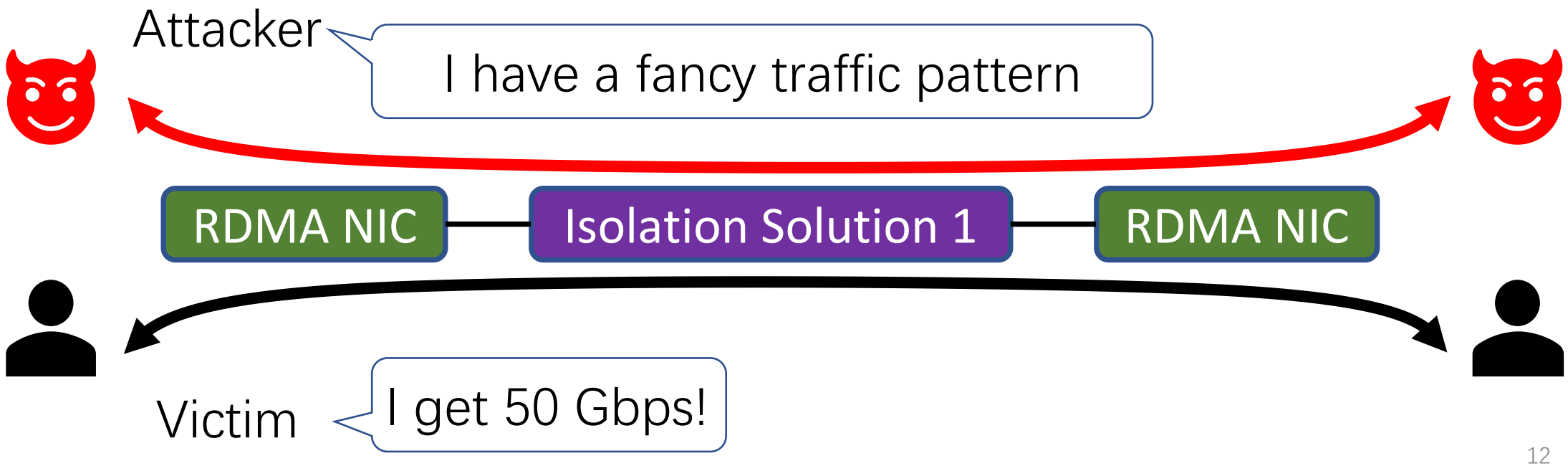
Evaluating RDMA Performance Isolation



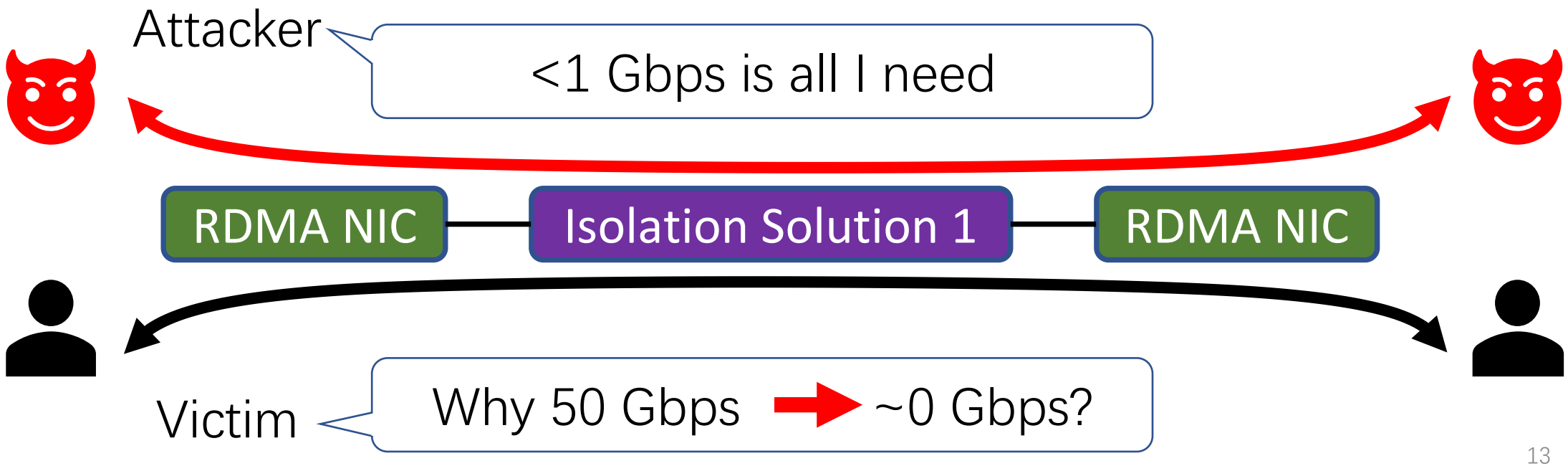
Evaluating RDMA Performance Isolation



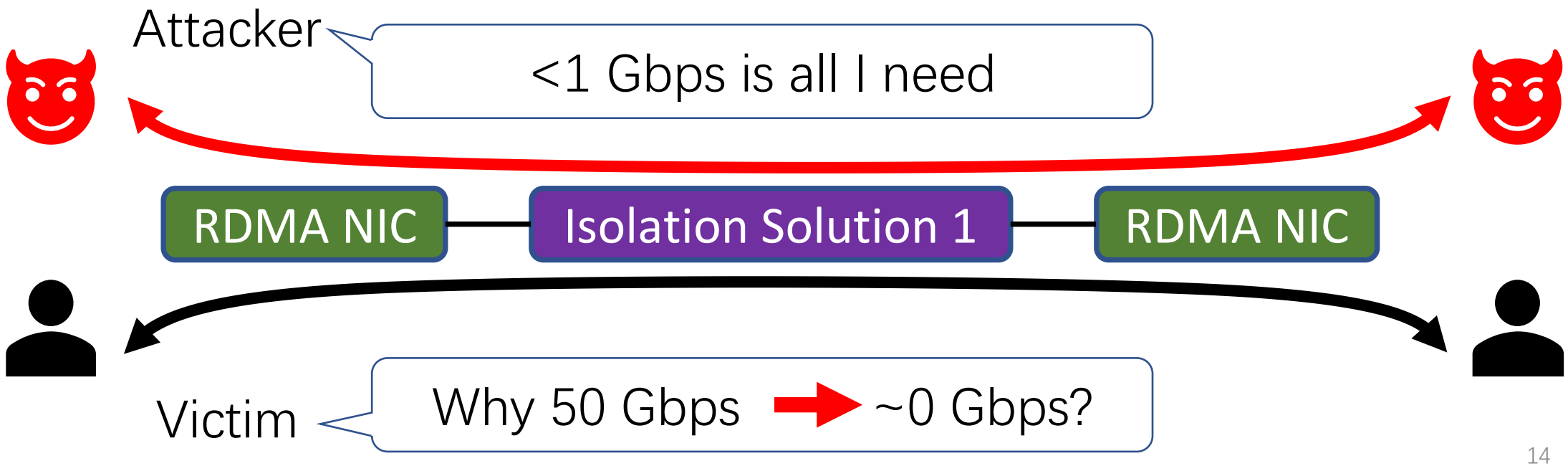
However…


Attacker
I have a fancy traffic pattern
RDMA NIC
Isolation Solution 1
RDMA NIC
Victim
I get 50 Gbps!

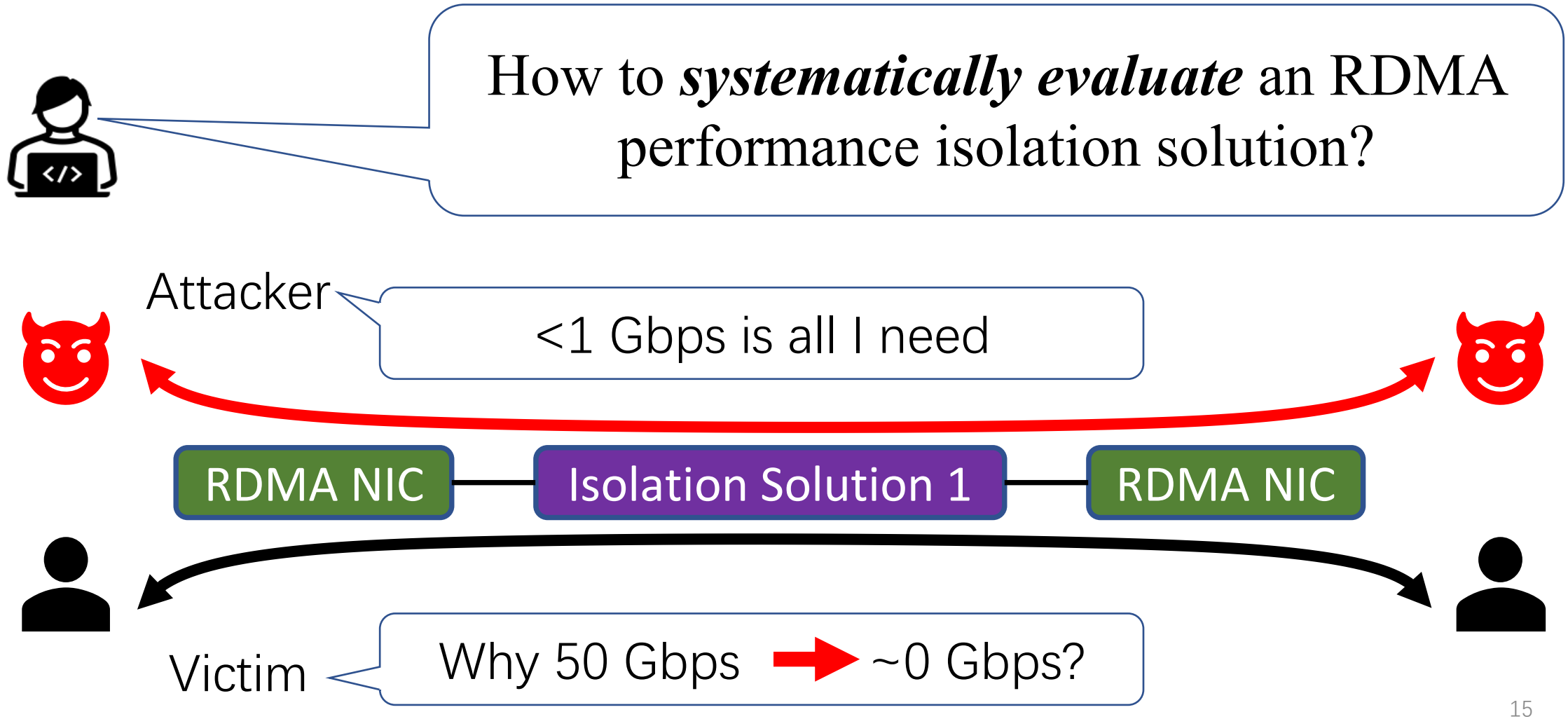
However…


Attacker
<1 Gbps is all I need
RDMA NIC
Isolation Solution 1
RDMA NIC
Victim
Why 50 Gbps
~0 Gbps?

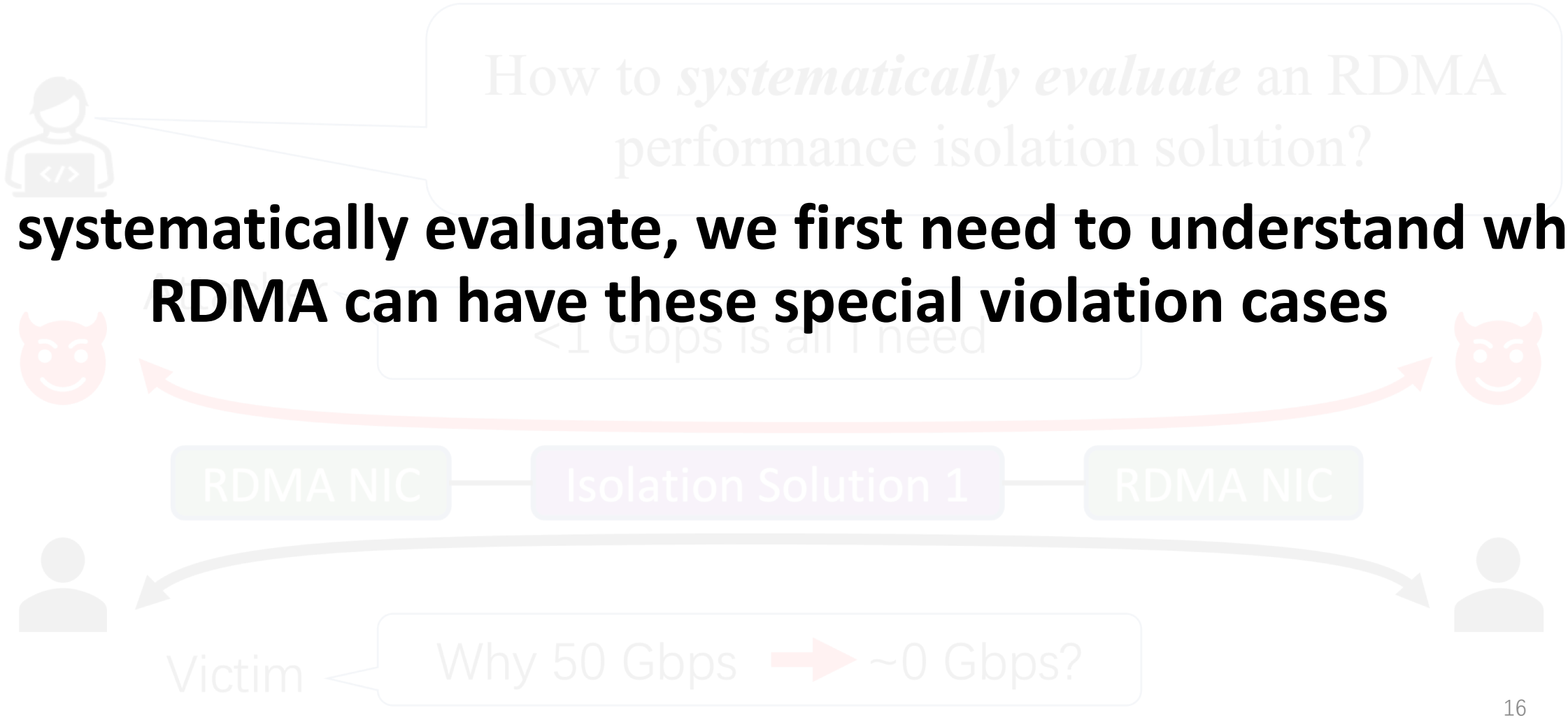
Existing Isolation Tests Have Limited Coverage



Existing Isolation Tests Have Limited Coverage

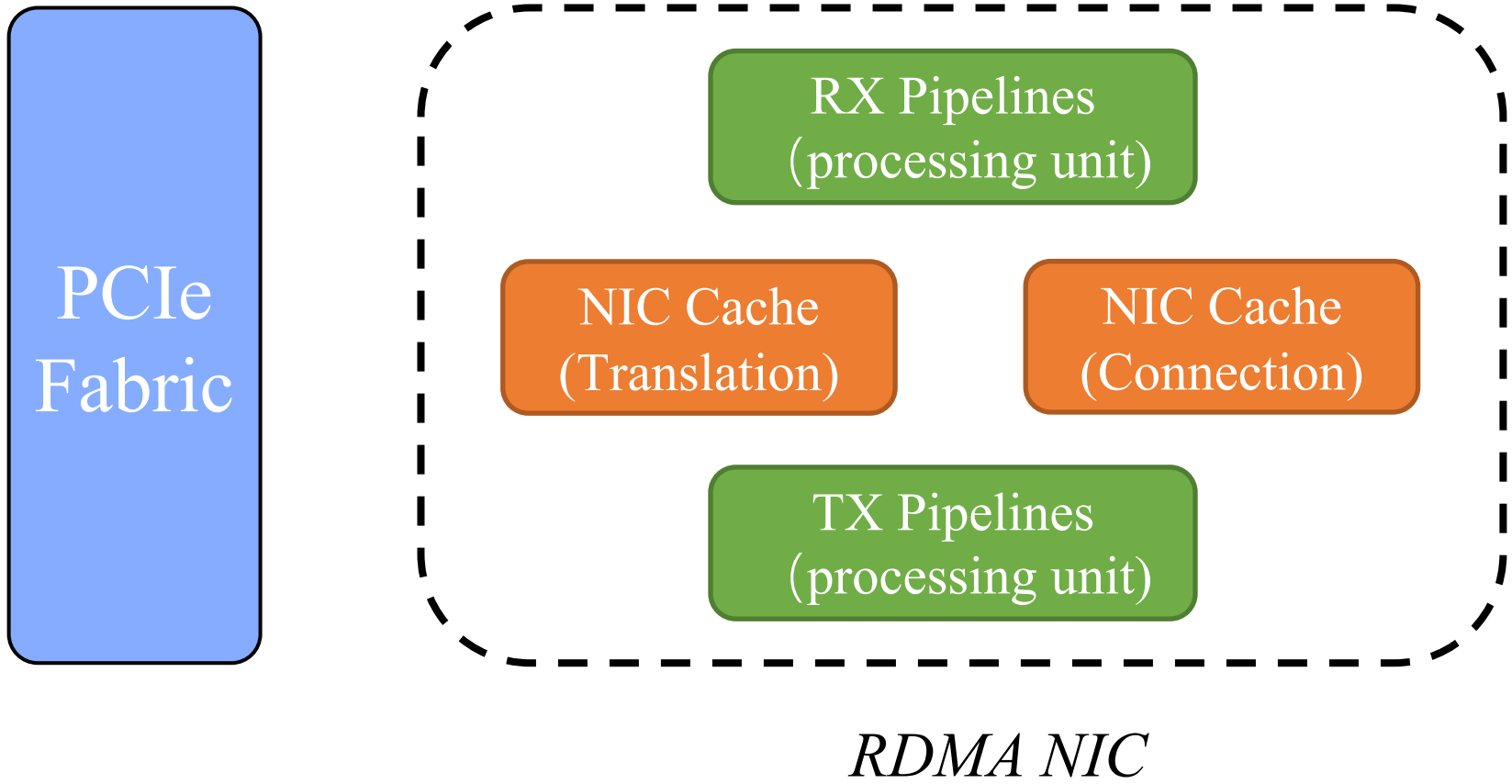


Existing Isolation Tests Have Limited Coverage

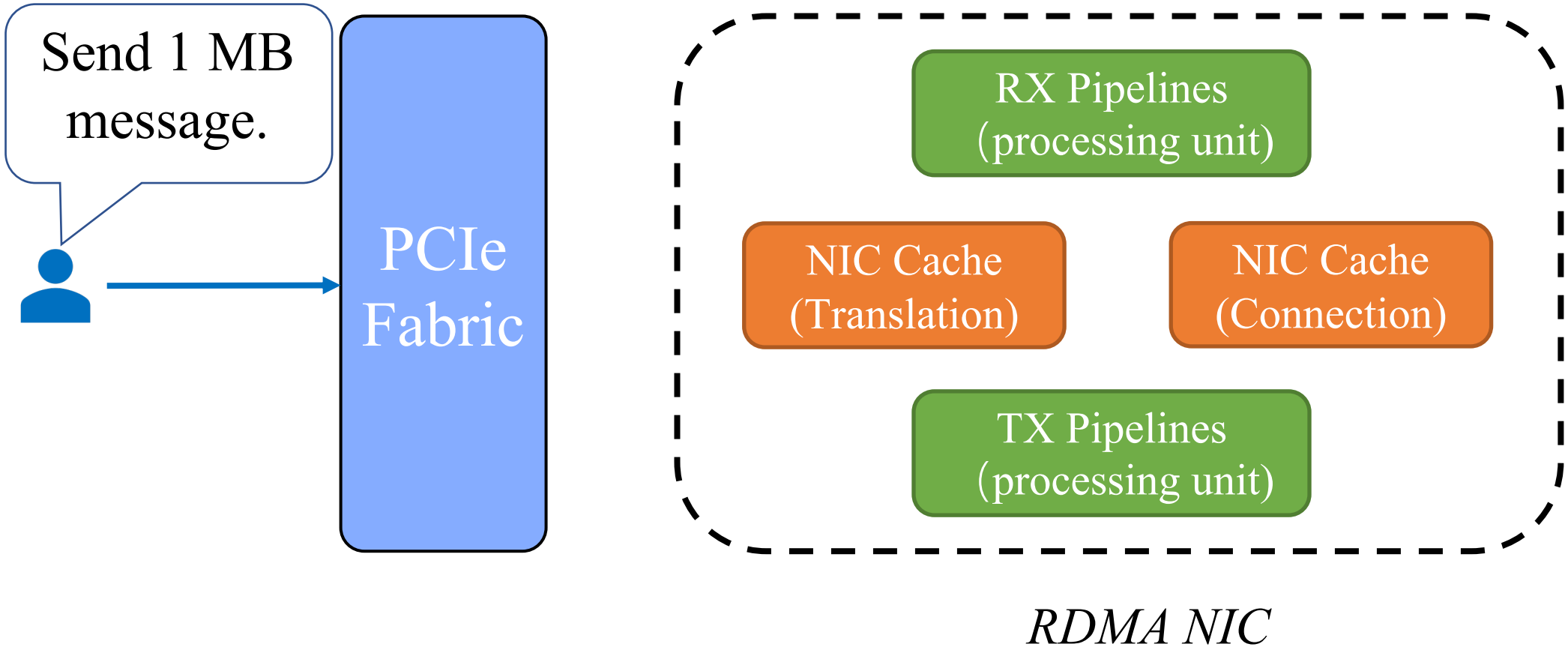


To systematically evaluate, we first need to understand why RDMA can have these special violation cases

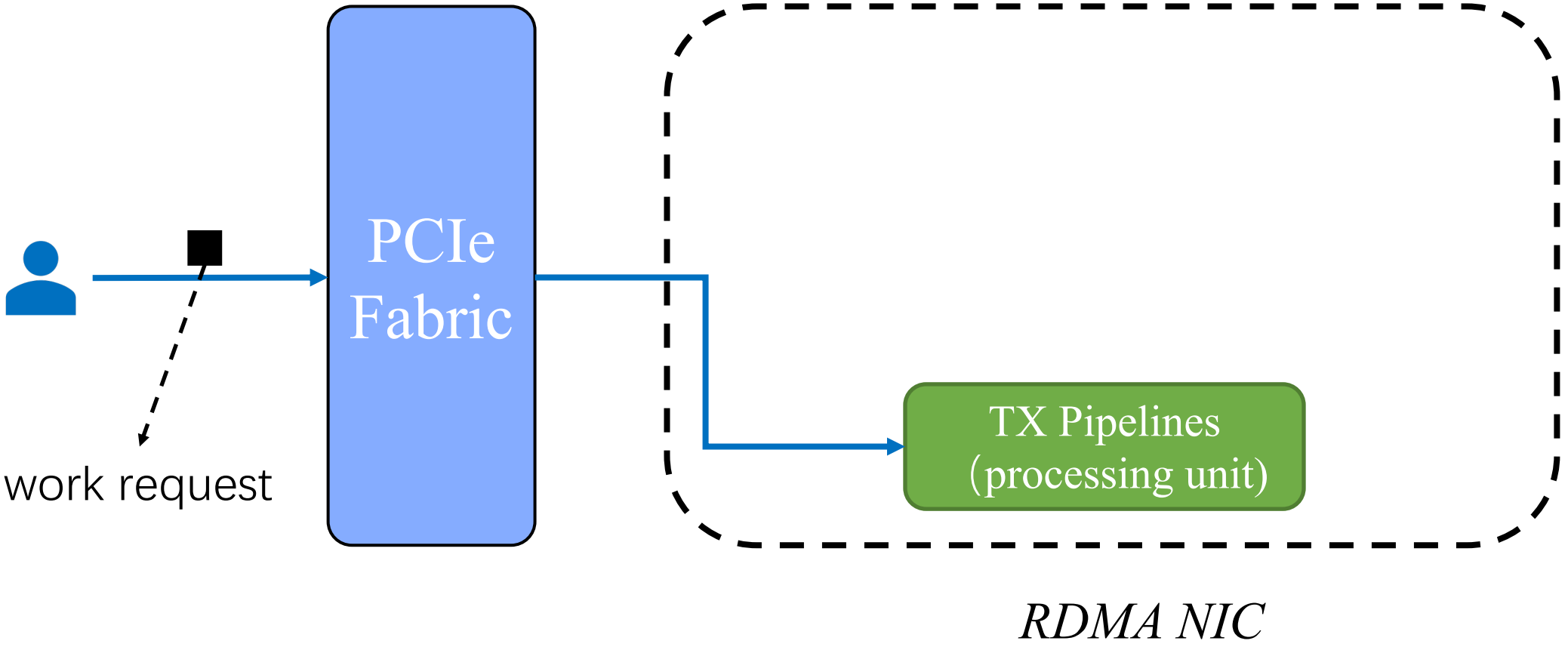
RDMA Microarchitecture Resource Contention



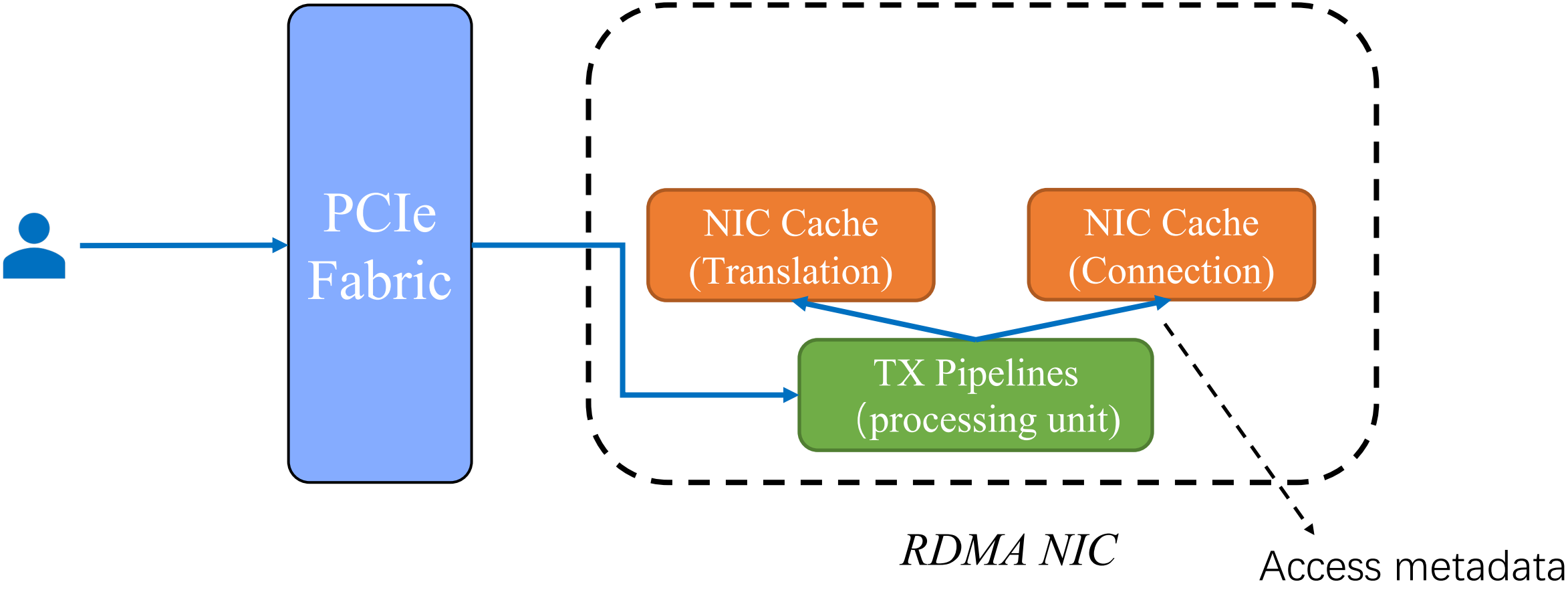
How RDMA Microarchitecture Resources Are Involved?



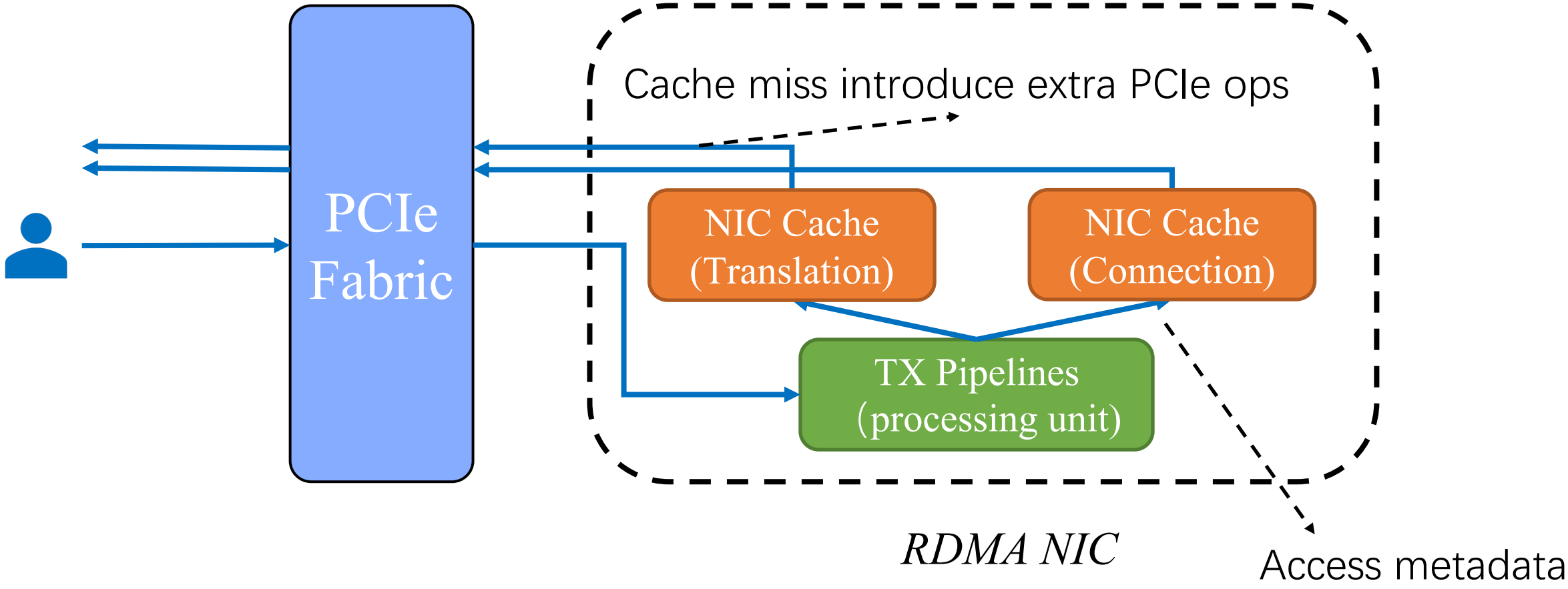
An ibv_post_send Example



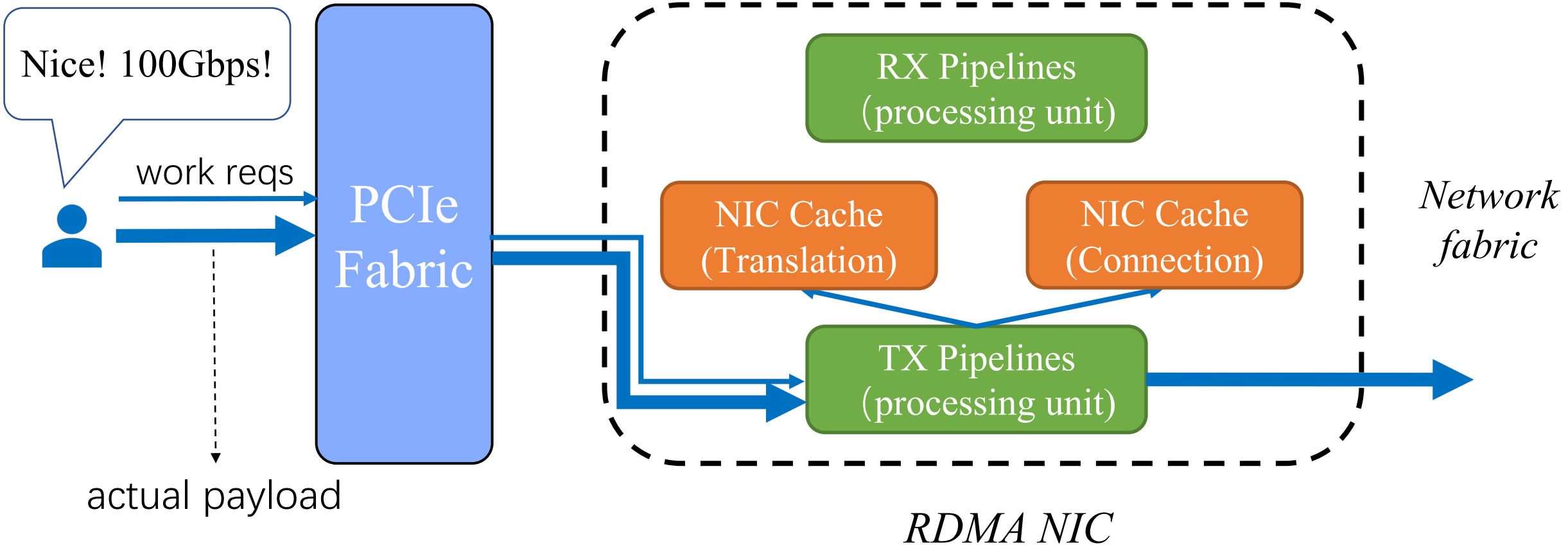
An ibv_post_send Example



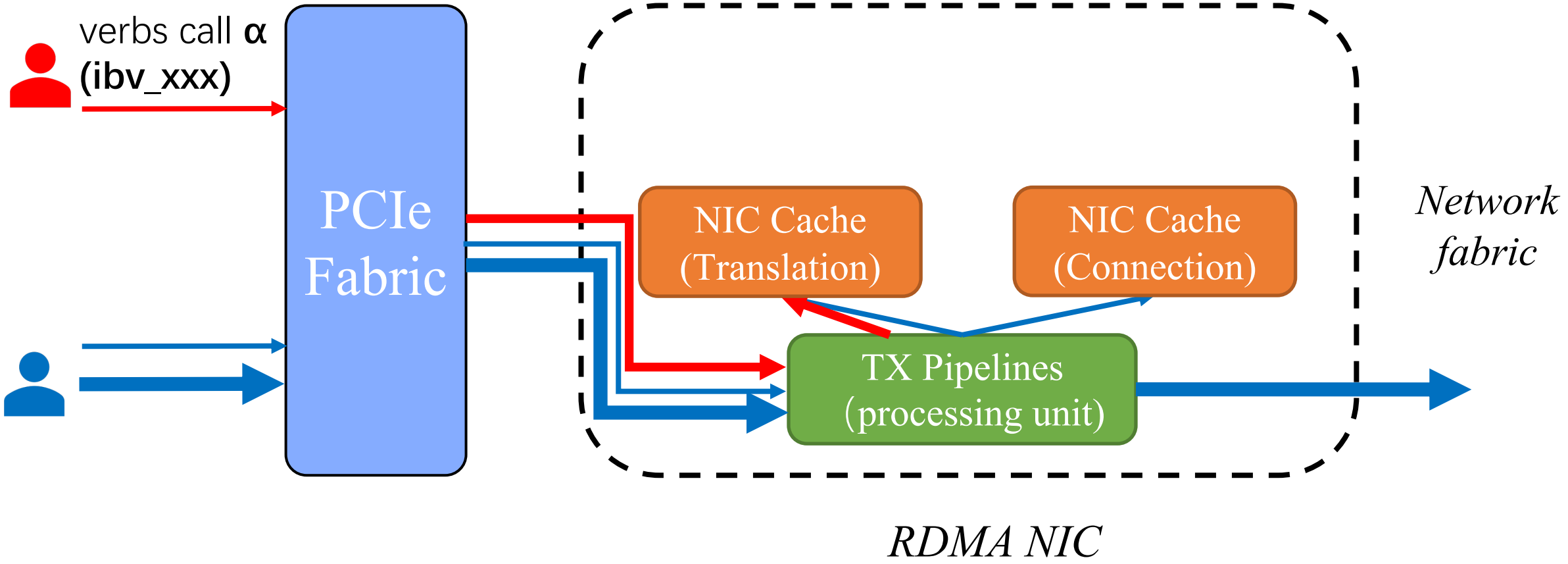
An ibv_post_send Example



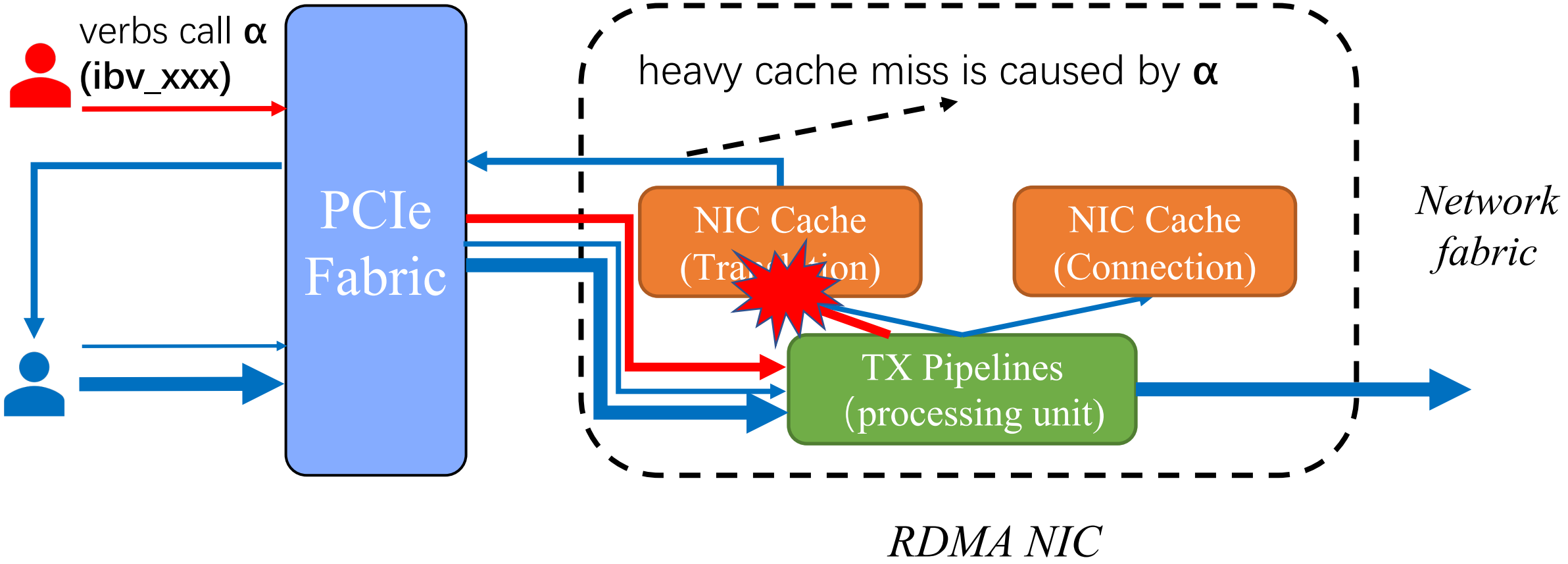
An ibv_post_send Example



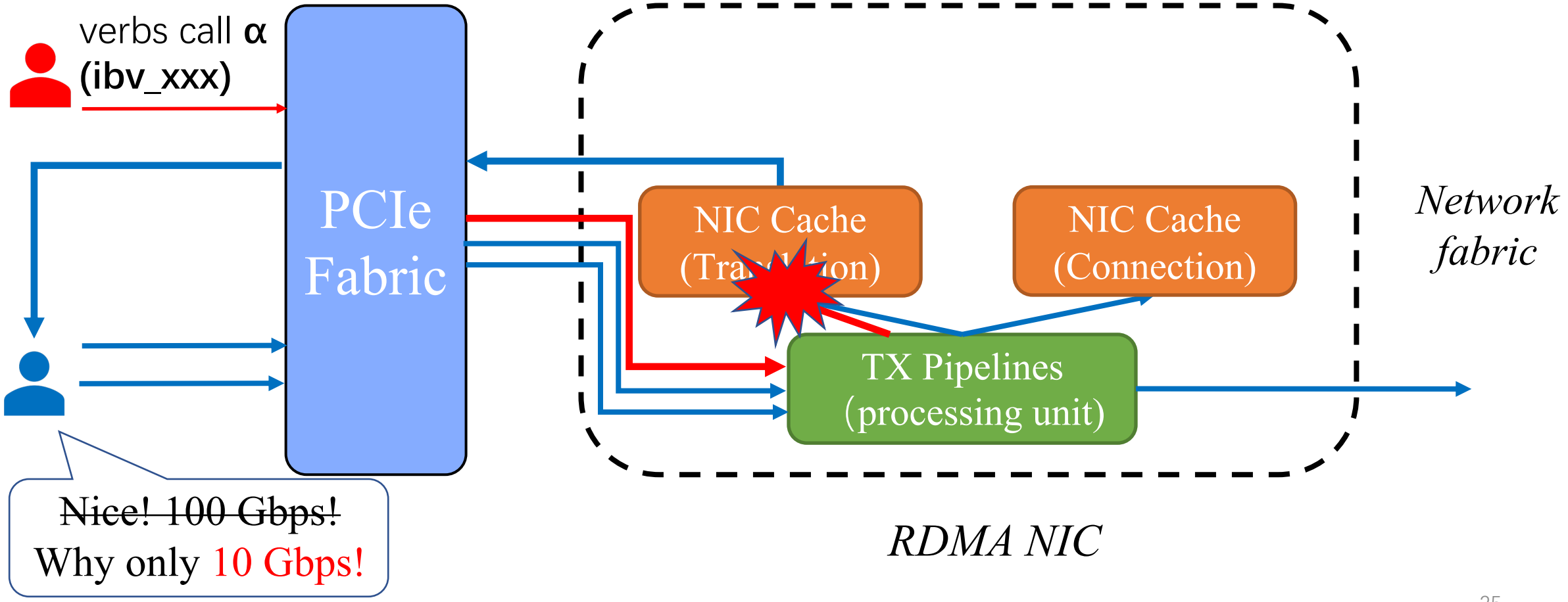
Microarchitecture Resource Contention



Microarchitecture Resource Contention



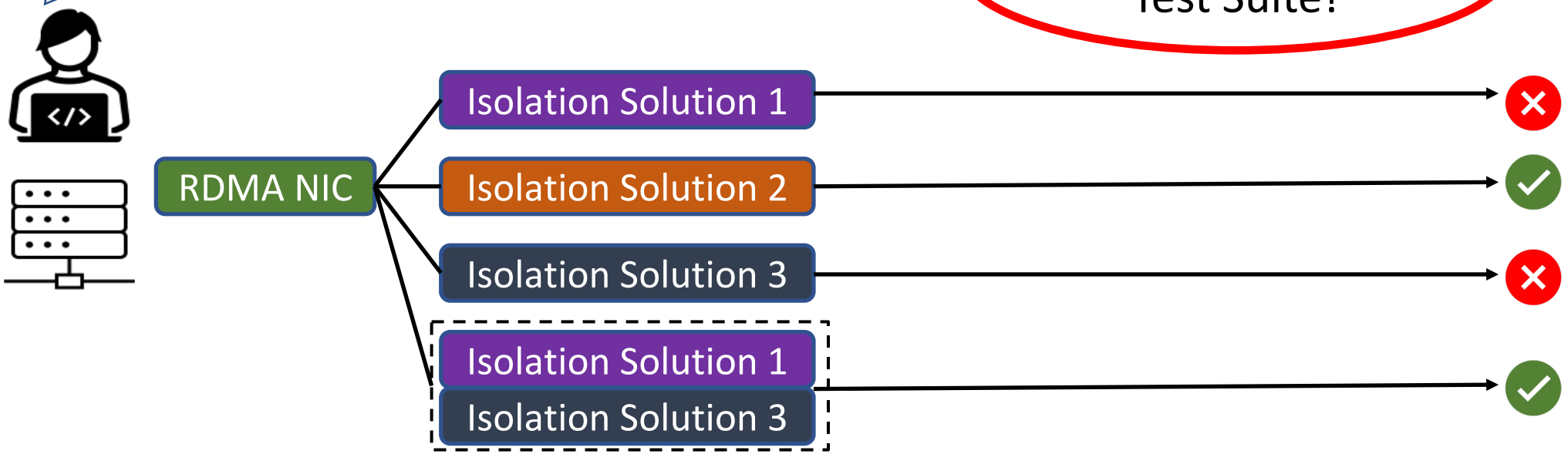
Microarchitecture Resource Contention



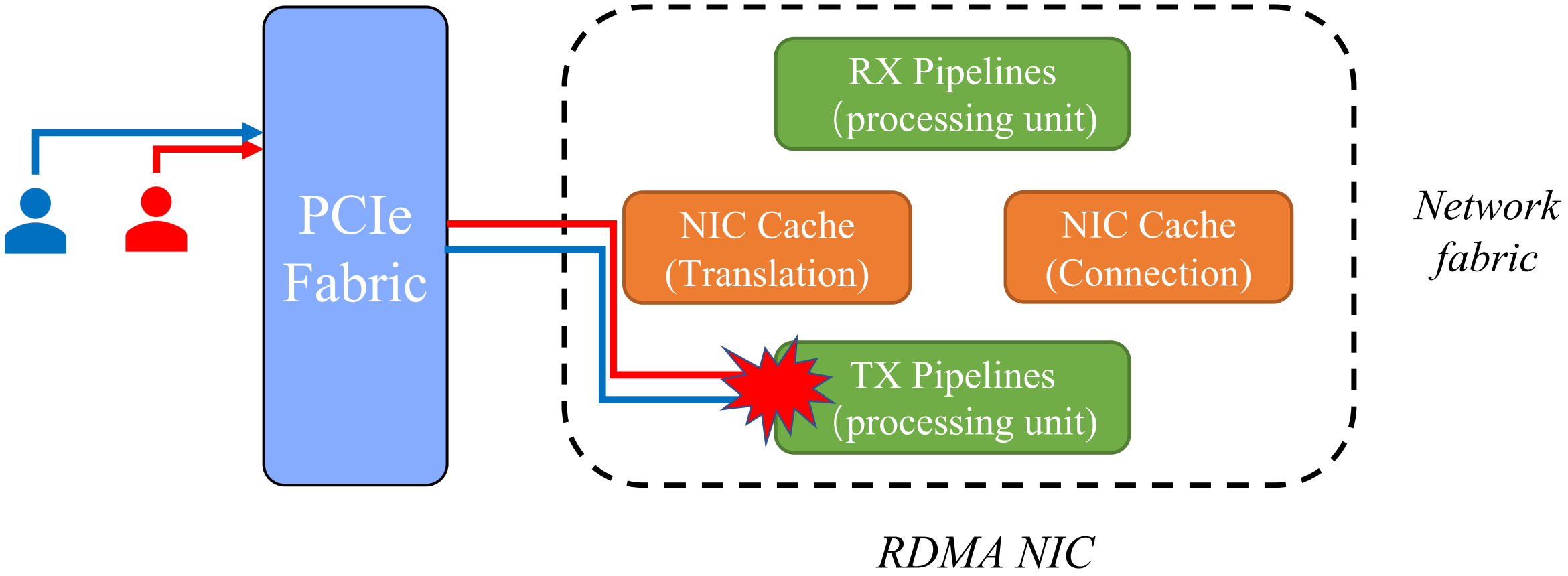
Goal - Systematic Test Suite

We need to build a systematic test suite for performance isolation, considering ***microarchitecture resource contention***!

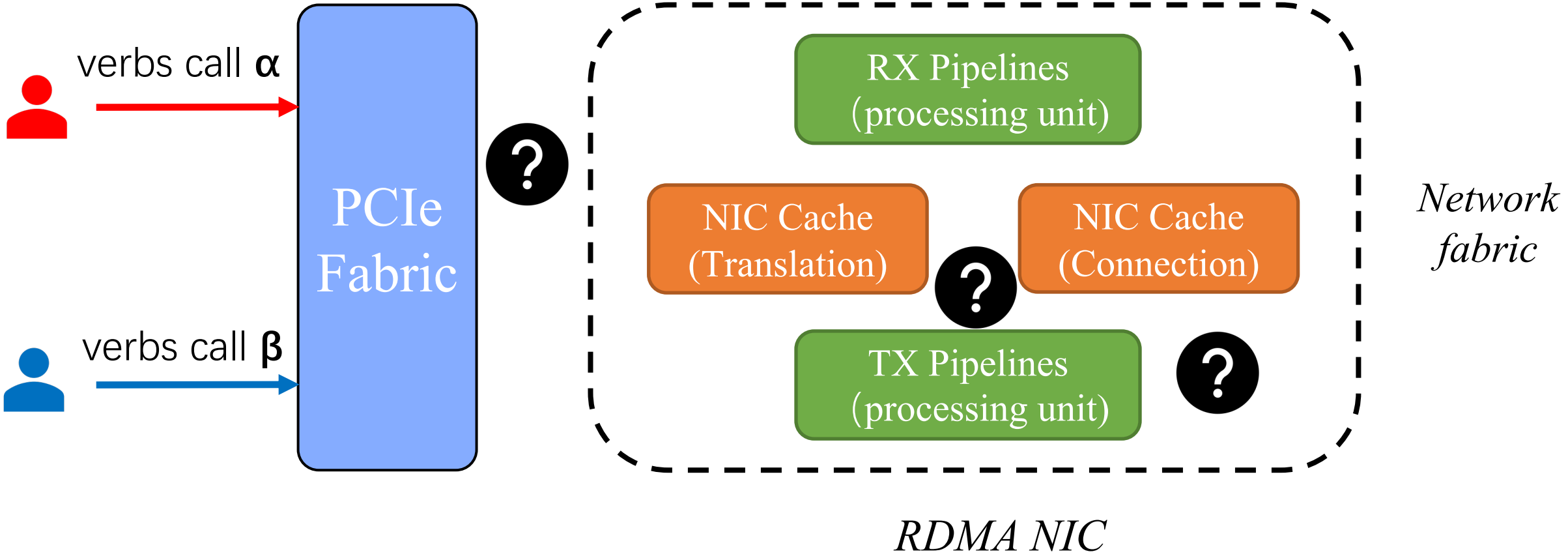
Systematic Test Suite!



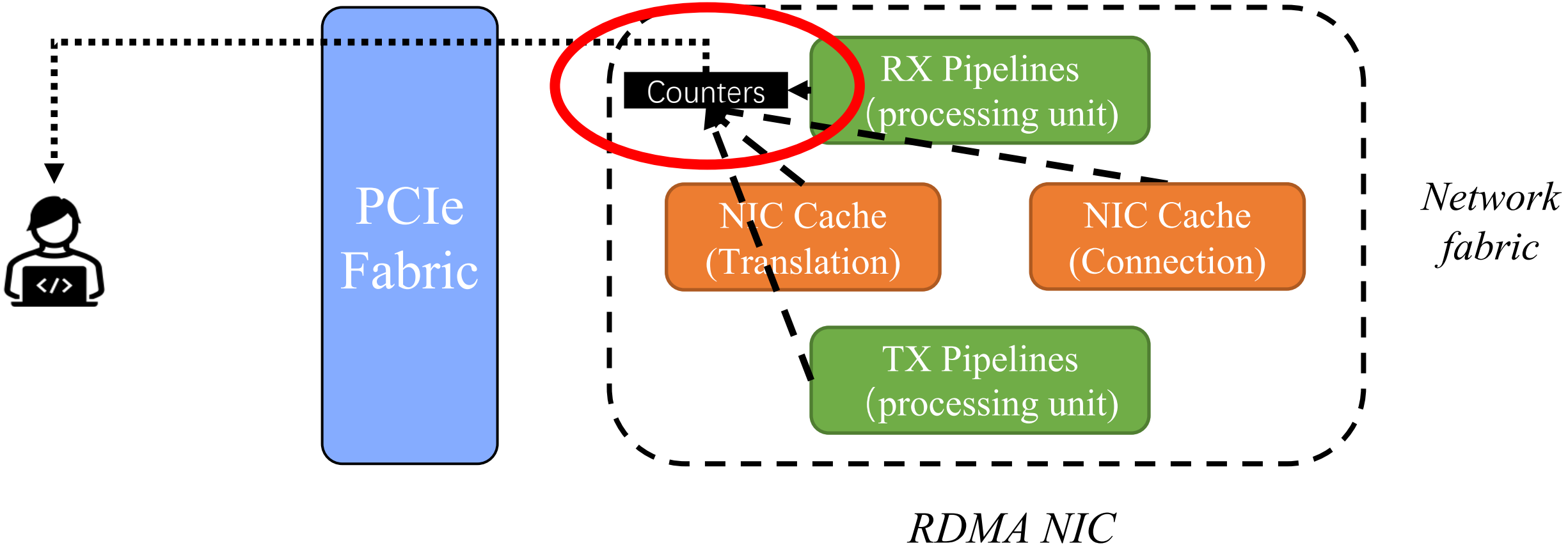
Methodology – Contention on Individual Resources



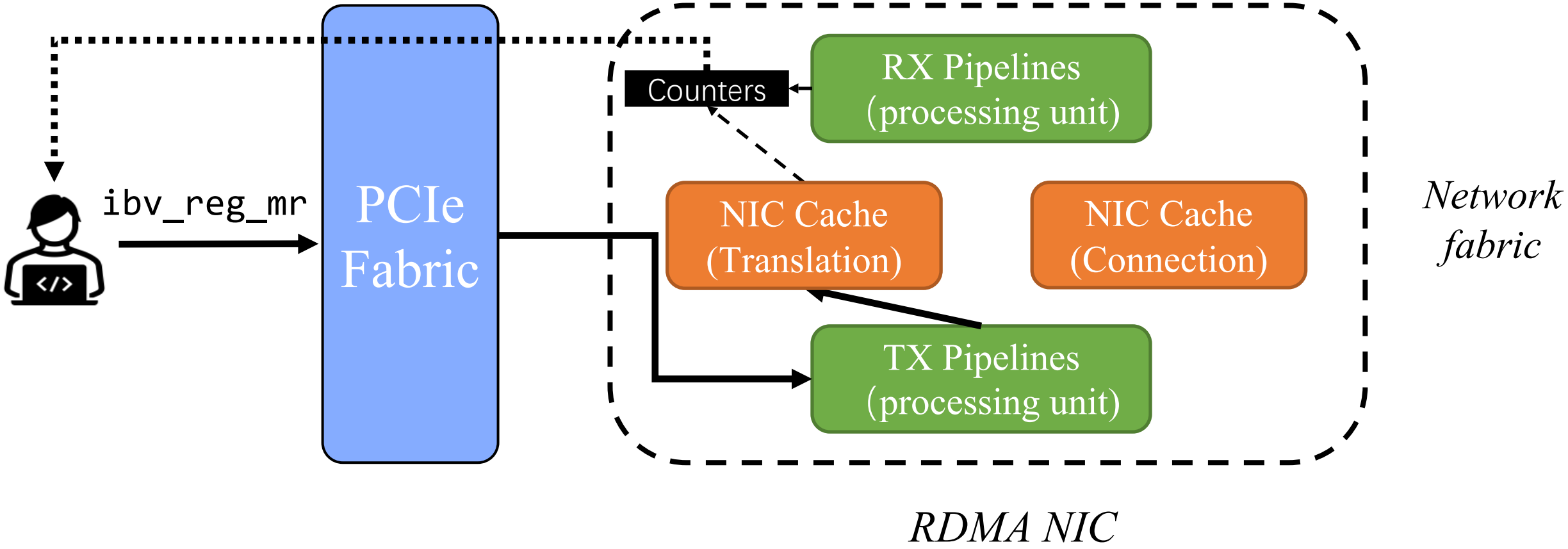
Step 1 - Identify Relationship of *verbs <-> resources*



Leverage Modern RNIC's Features!

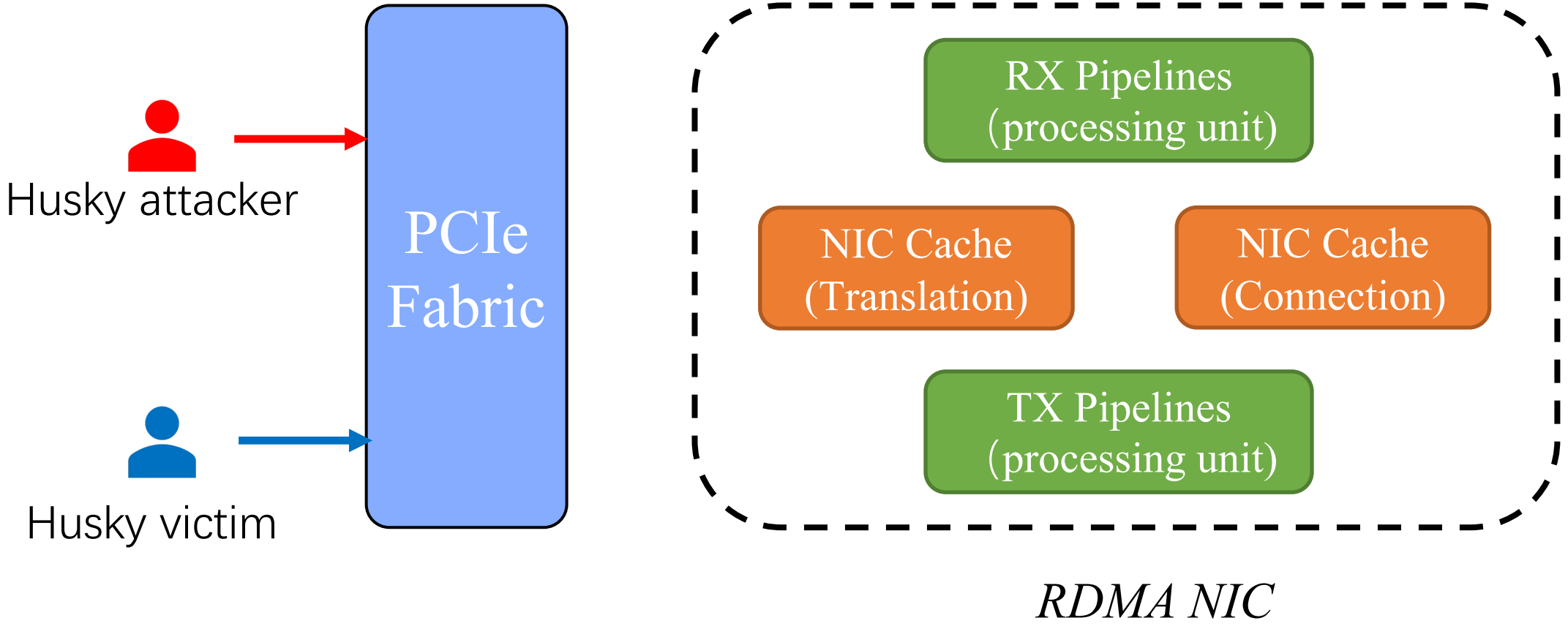


Construct A Qualitative Model of *verbs <-> resources*

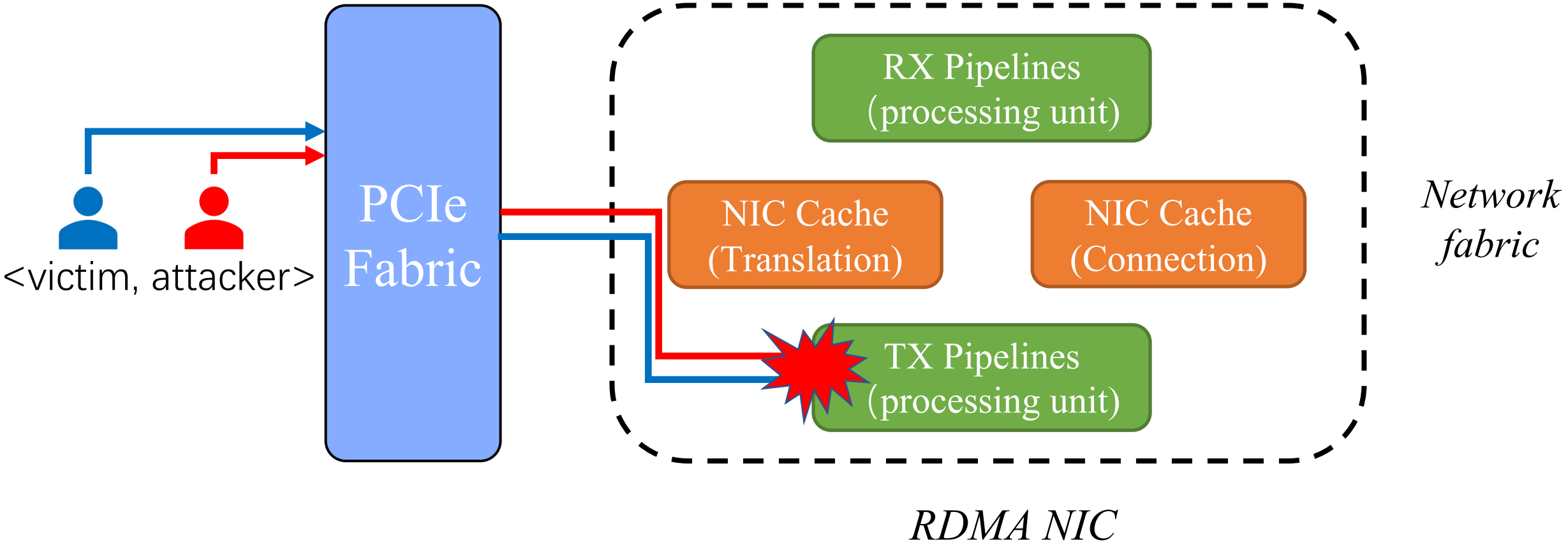


A qualitative relationship: `ibv_reg_mr` consumes NIC Cache (Translation)

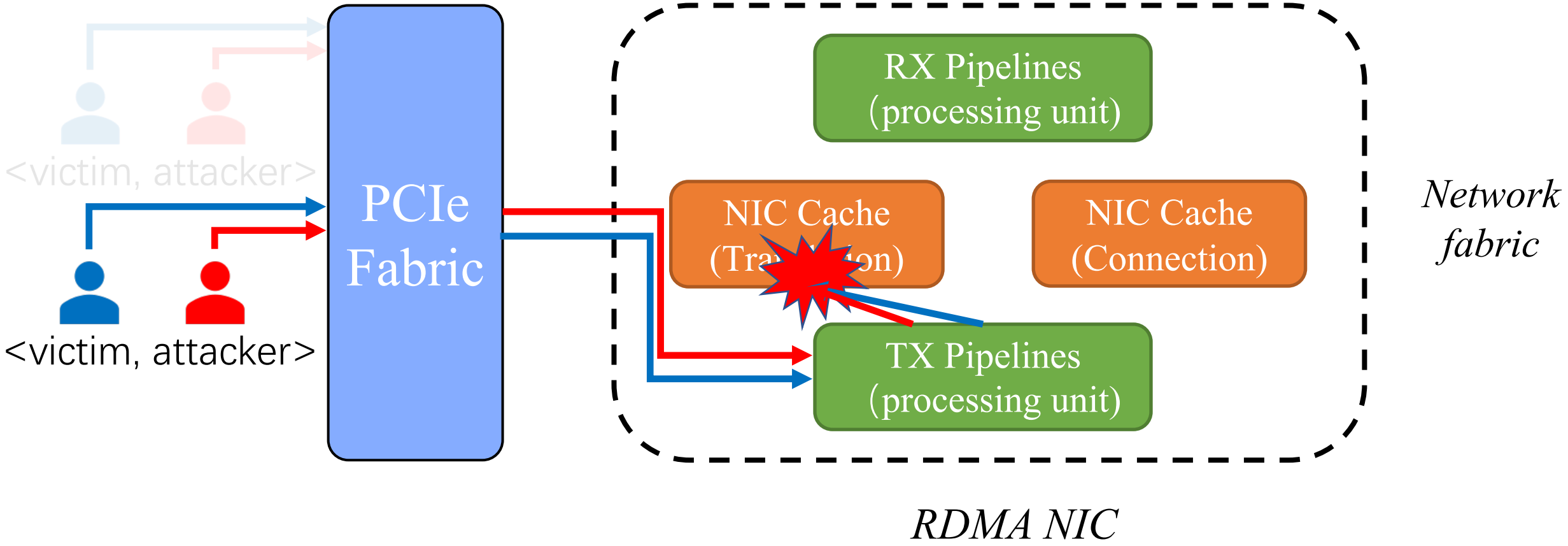
Step 2 - <victim, attacker> Suite to Exhaust Resources



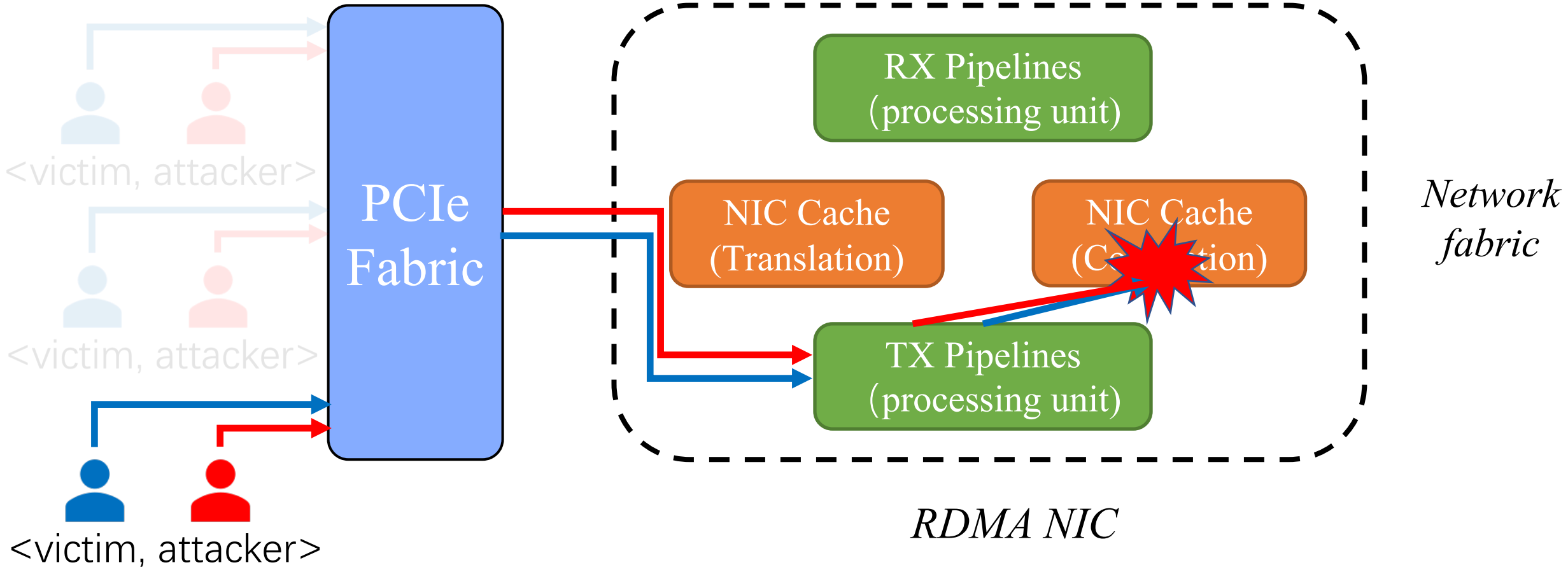
<v, a> Cause Resource Contention on TX Pipelines



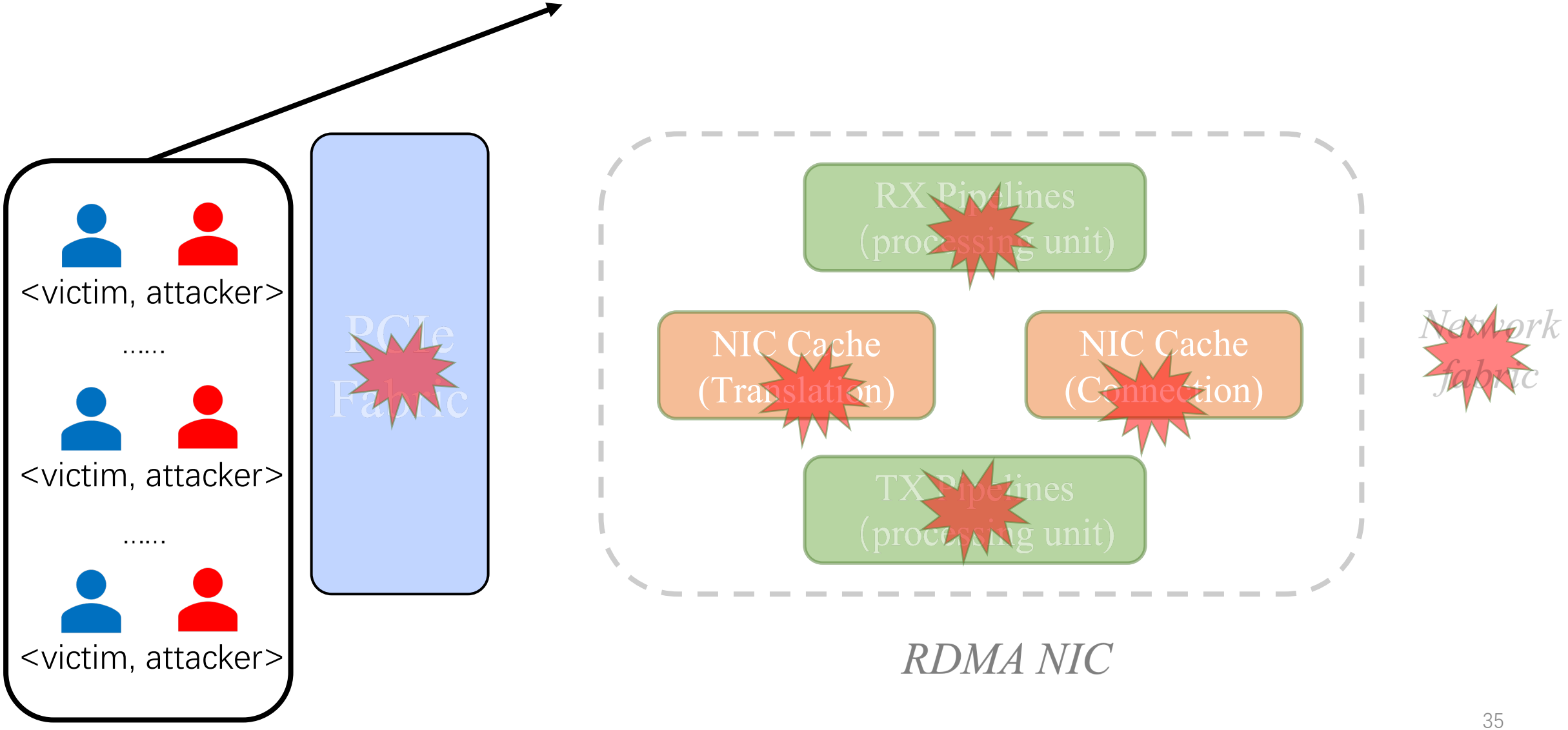
<v, a> Cause Resource Contention on NIC Cache



<v, a> Cause Resource Contention on NIC Cache



A Test Suite to Exhaust Each Resource Separately



Testbed Description

- RDMA NICs
  - Mellanox ConnectX-5 100Gbps
  - Chelsio T62100-LP-CR 100Gbps
  - Intel E810 100 Gbps
- CPU
  - Intel Xeon Gold 5215
- PCIe 3.0 x 16 (8GT x16)

Findings

- Key finding #1: control verbs can cause excessive cache misses and a drastic performance reduction.
- Key finding #2: performance interference between different data verbs depends on the complexity of verbs.
- Key finding #3: error handling can stall RNIC processing units and hang all the applications.
- Key finding #4: PCIe bandwidth will only become the bottleneck when the request size is in a specific range.

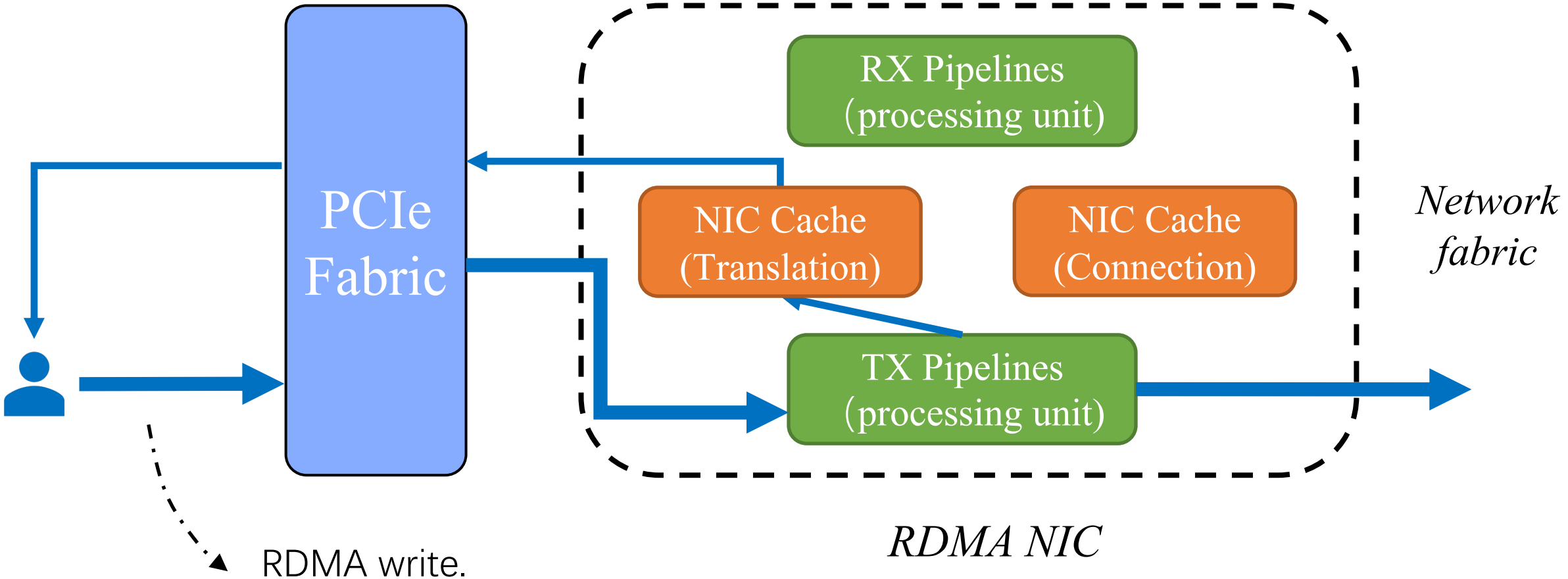
- Other finding #1: Data verbs contend for different RNIC caches
- Other finding #2: Wide range access across many objects (QP, CQ, MR) causes ICM cache misses.
- Other finding #3: The impact of control verbs is restricted by its kernel involvement.

Findings

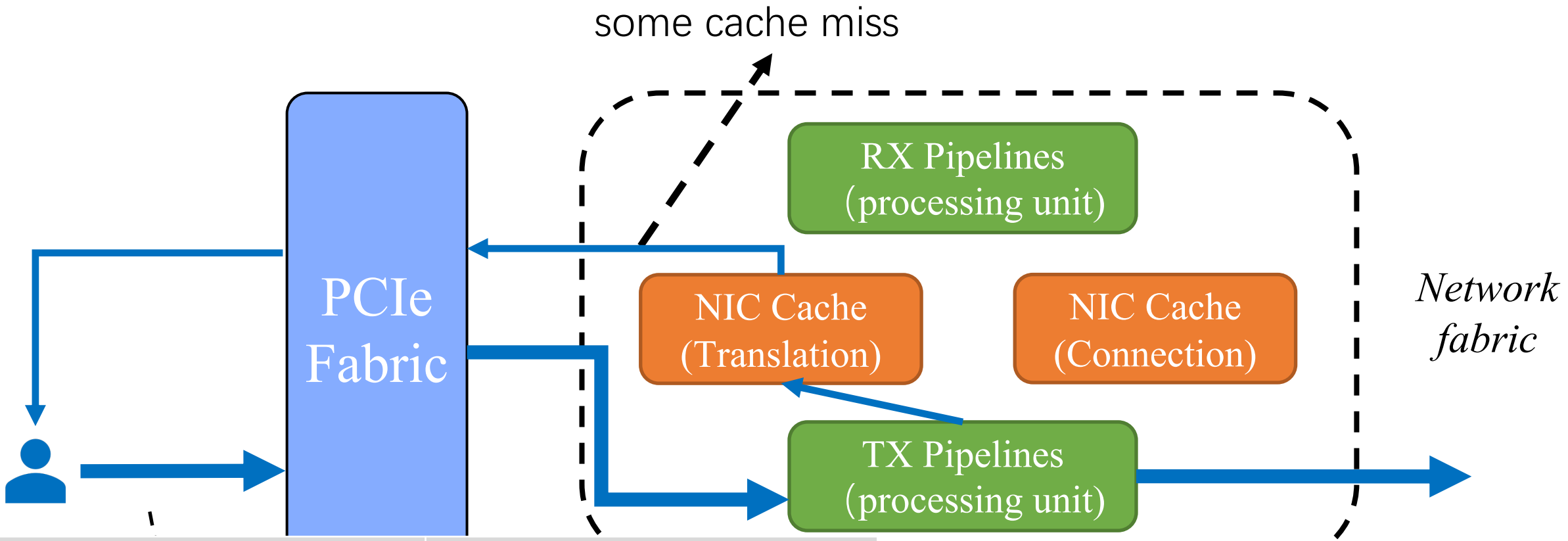
- **Key finding #1: control verbs can cause excessive cache misses and a drastic performance reduction.**
- Key finding #2: performance interference between different data verbs depends on the complexity of verbs.
- **Key finding #3: error handling can stall RNIC processing units and hang all the applications.**
- Key finding #4: PCIe bandwidth will only become the bottleneck when the request size is in a specific range.

- Other finding #1: Data verbs contend for different RNIC caches
- Other finding #2: Wide range access across many objects (QP, CQ, MR) causes ICM cache misses.
- Other finding #3: The impact of control verbs is restricted by its kernel involvement.

Memory Registration Cause Severe Cache Contention

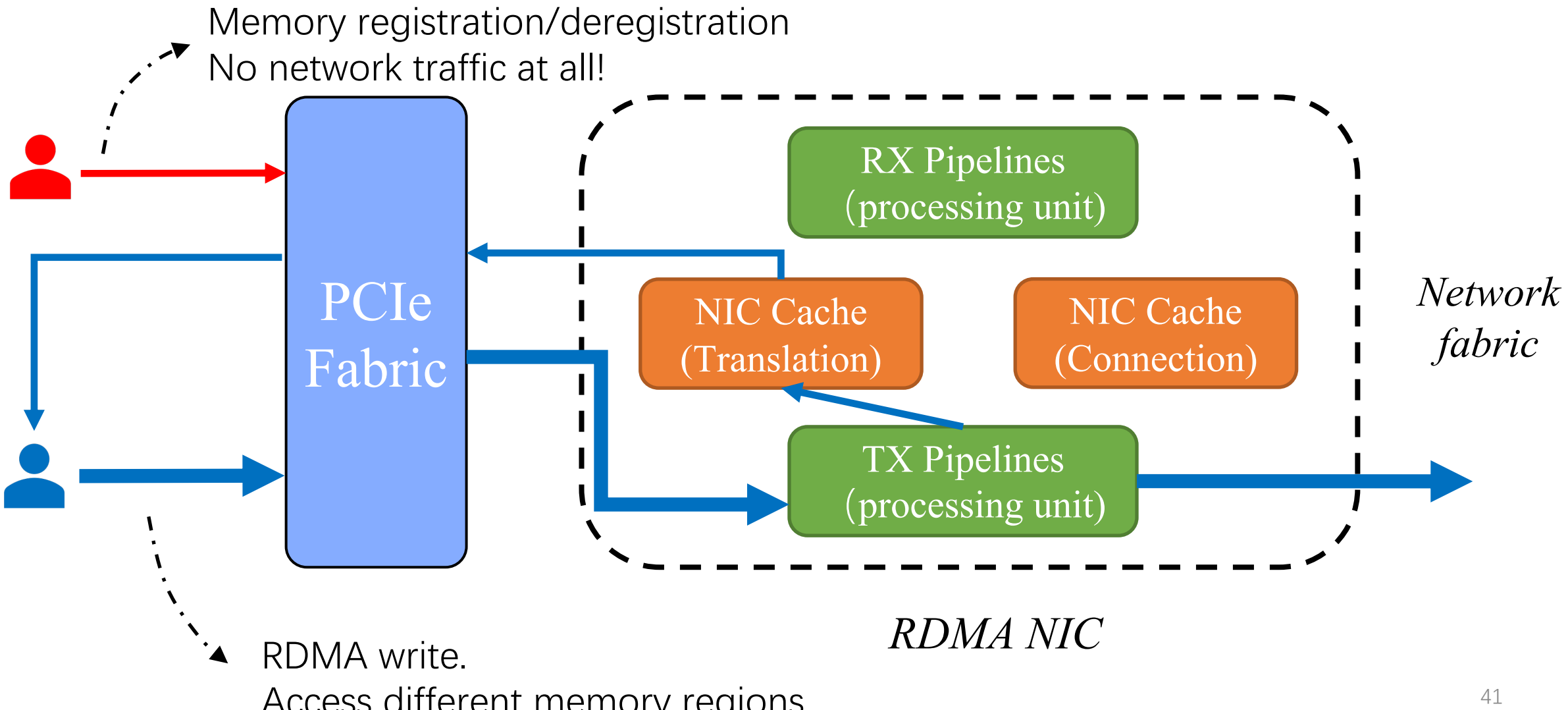


Memory Registration Cause Severe Cache Contention

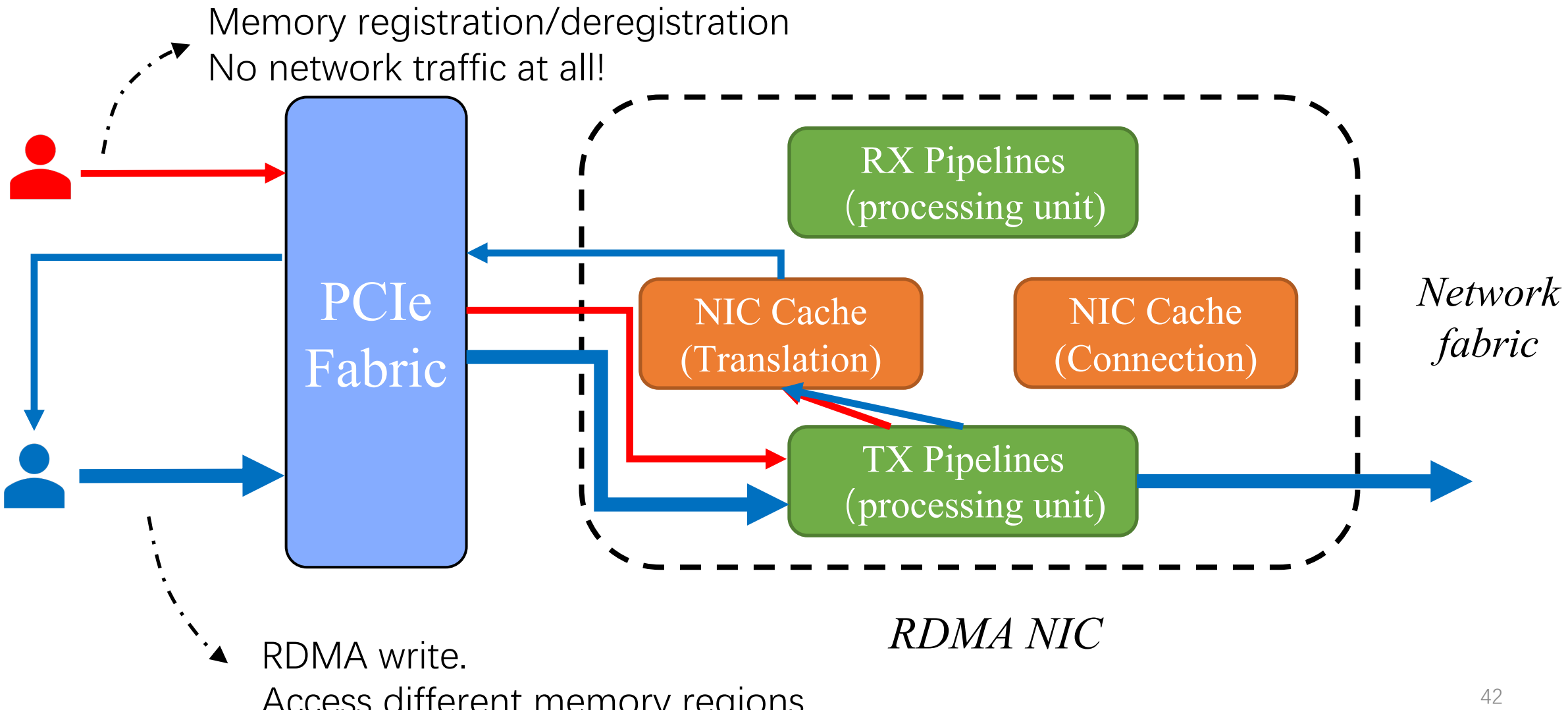


	Victim (blue) Only
BW / Gbps	96.6
Cache Miss Rate	17.2%

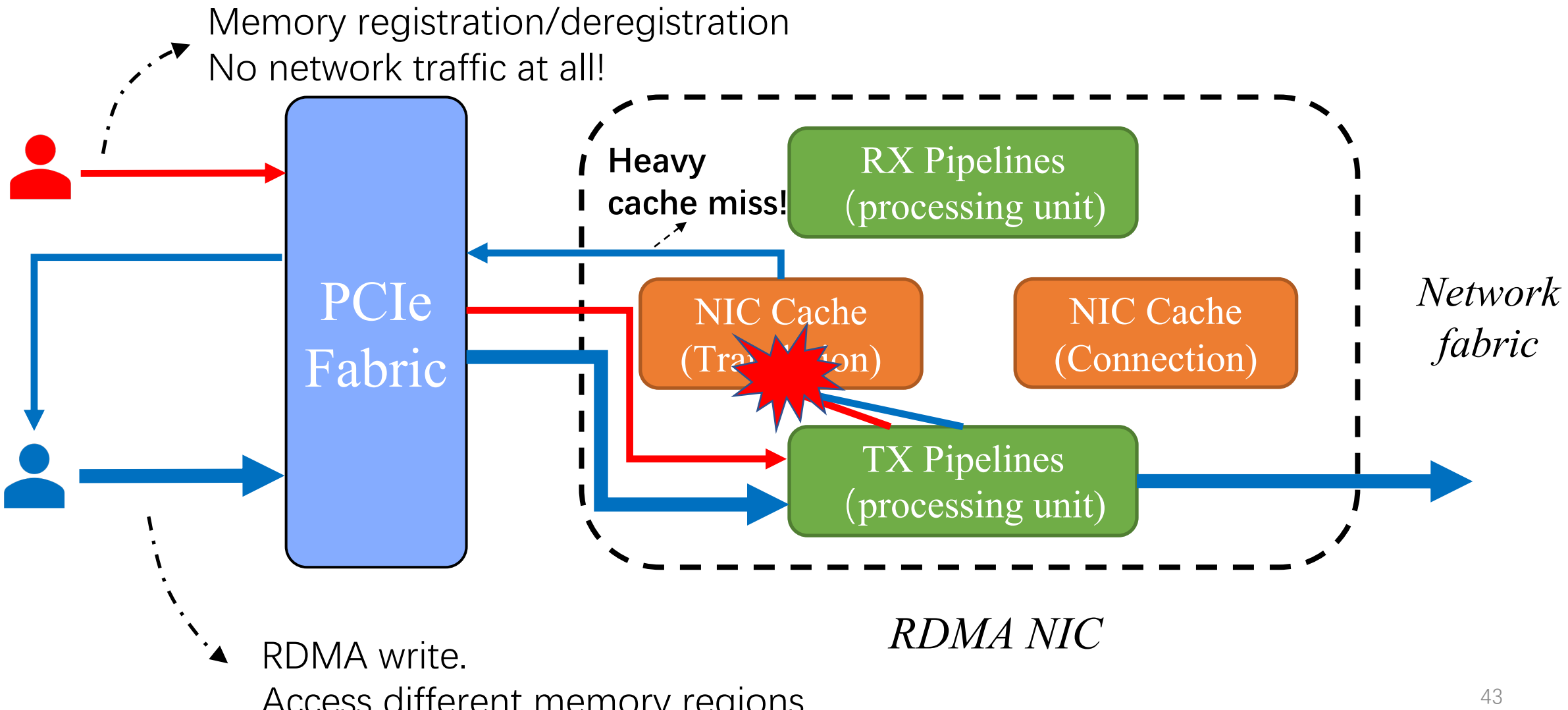
Memory Registration Cause Severe Cache Contention



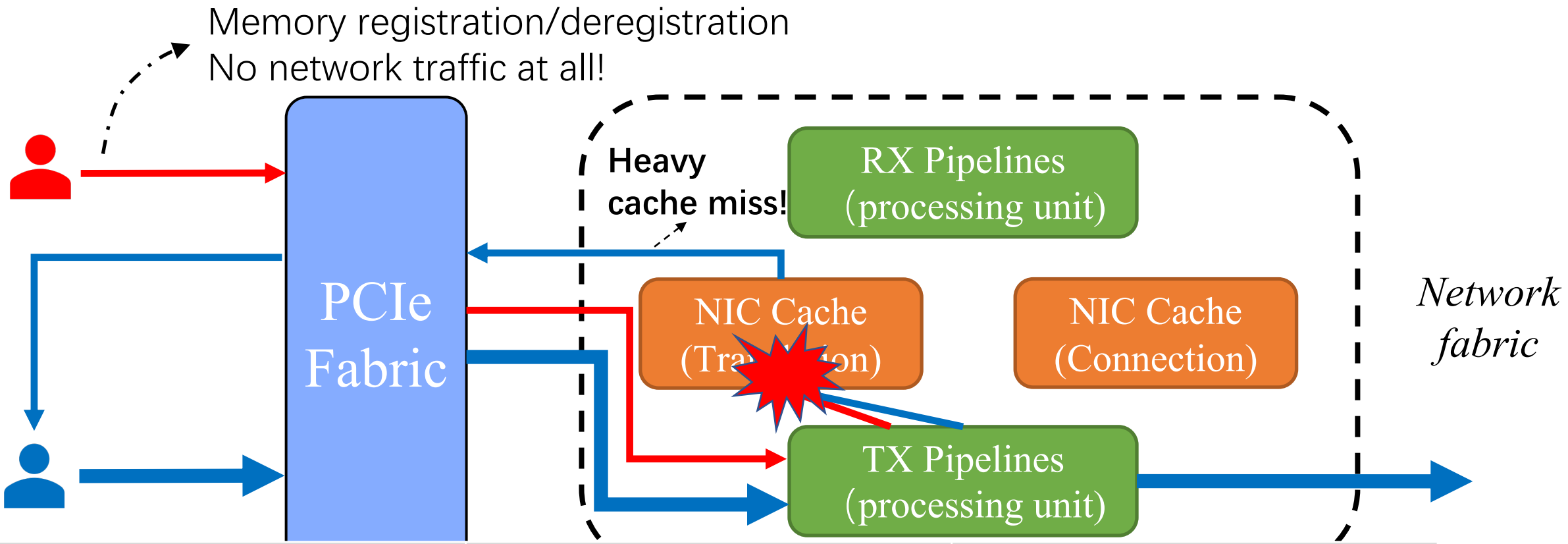
Memory Registration Cause Severe Cache Contention



Memory Registration Cause Severe Cache Contention

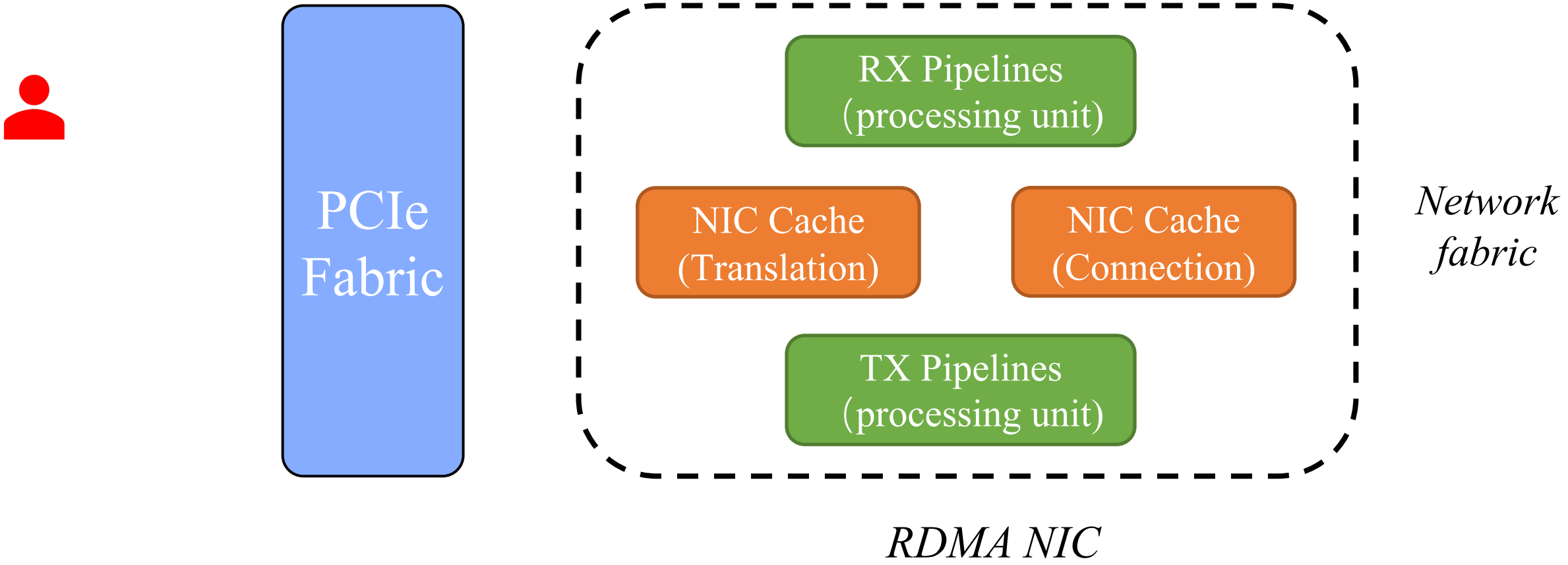


Memory Registration Cause Severe Cache Contention

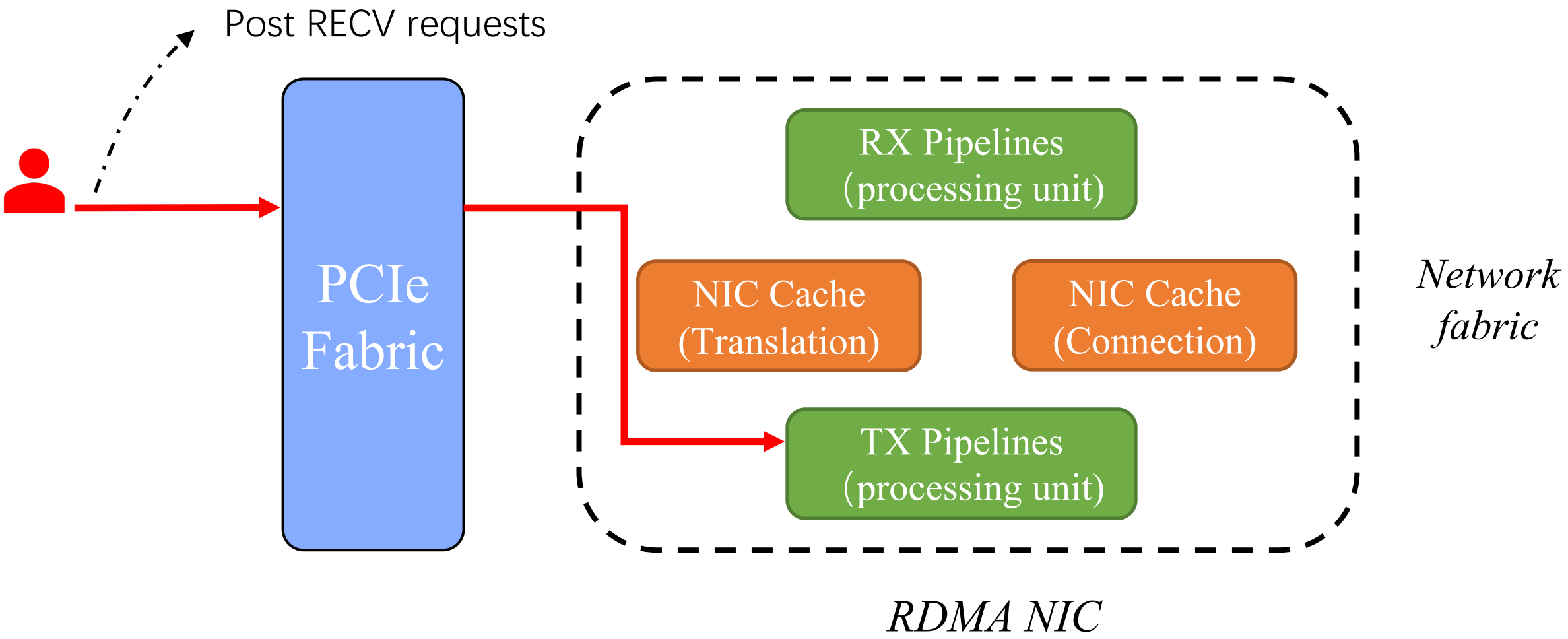


	Victim (blue) Only	w/ Attacker (red)
BW / Gbps	96.6	48.0
Cache Miss Rate	17.2%	49.1%

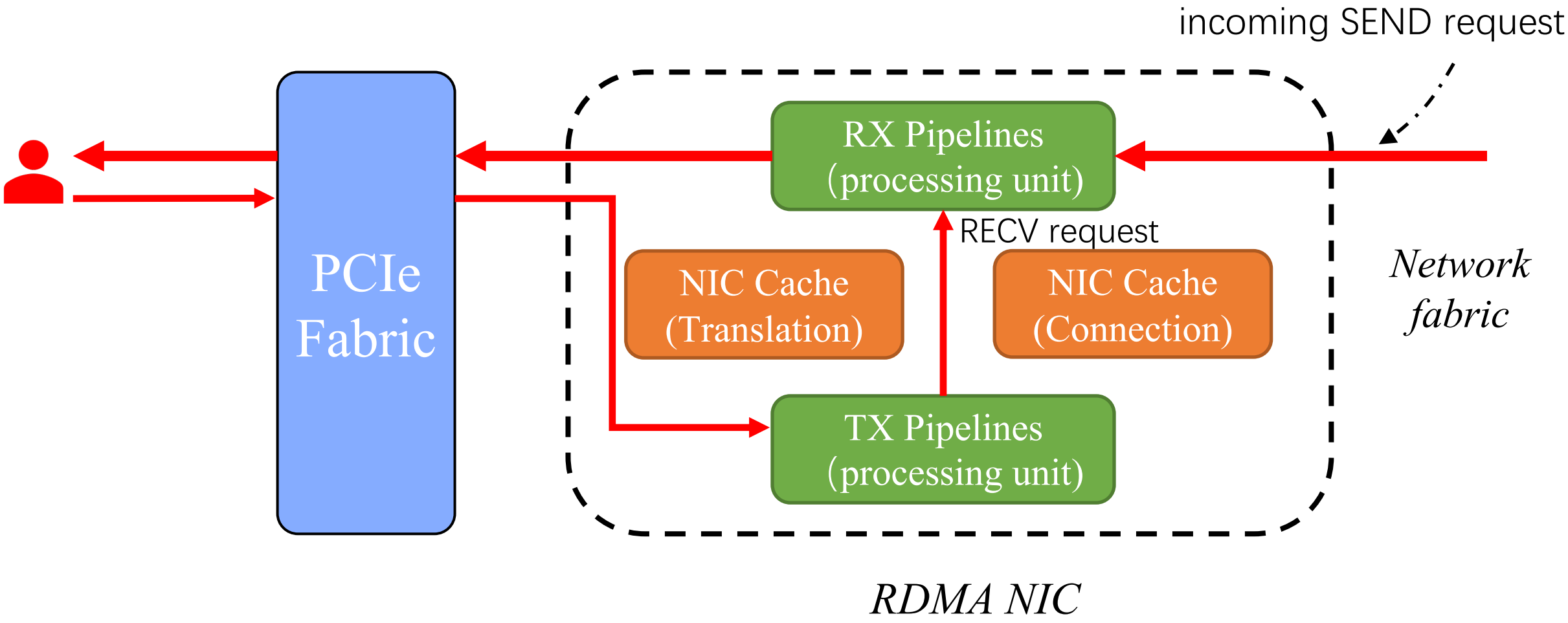
Error Handling Exhaust RX Pipelines



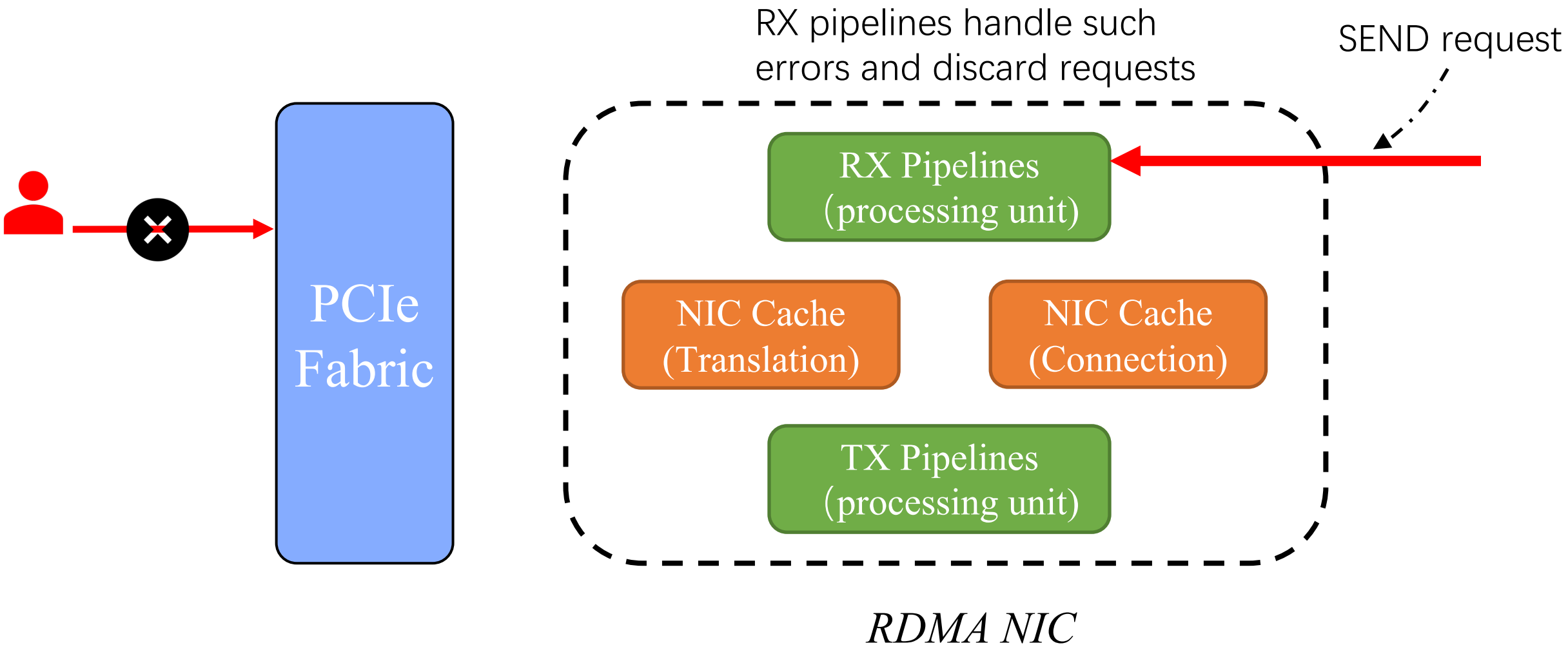
SEND/RECV Needs Prepared Receive Requests



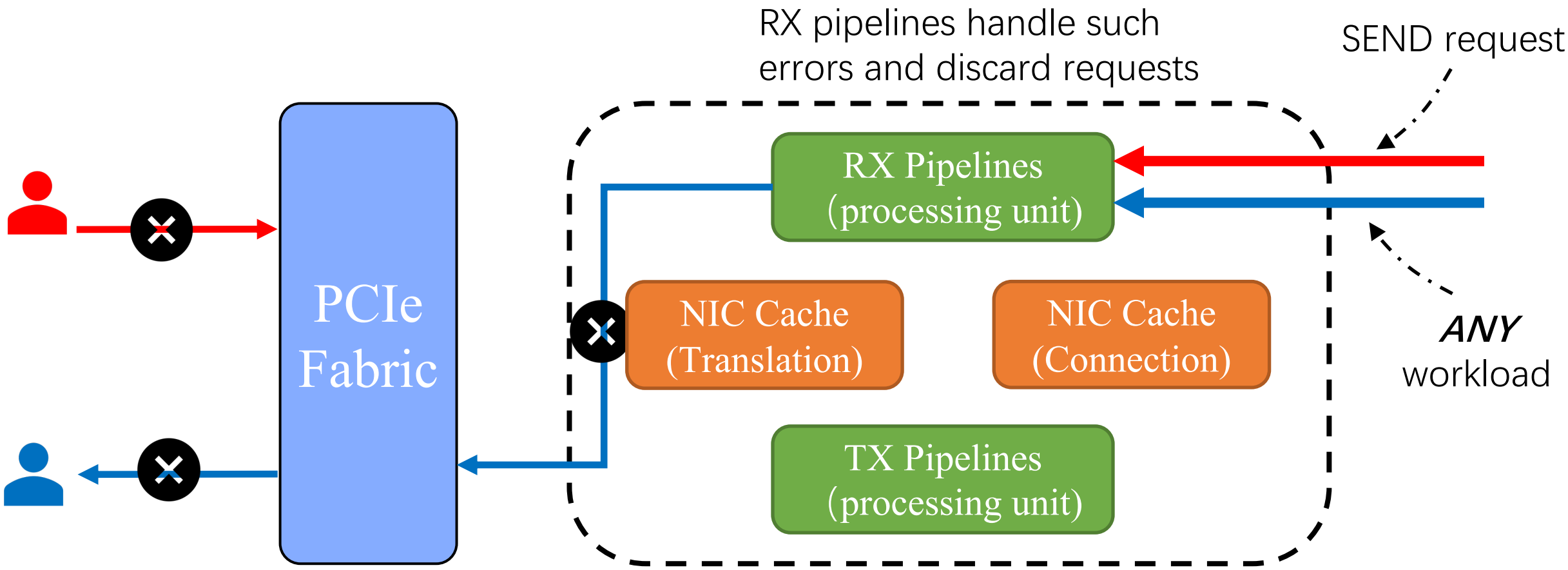
SEND/RECV Needs Prepared Receive Requests



Receive-Not-Ready (RNR) Consumes RX pipelines



RNR Cause Severe RX Pipelines Contention



	Victim Bandwidth	Attacker Bandwidth
w/o RNR error	97.07 Gbps	\
w/ RNR error	0.018 Gbps	0 Gbps

Husky: A Test Suite for RDMA Performance Isolation

- Husky attacker – synthetic workloads that
  - 6 exhaust NIC bandwidth
  - 4 exhaust NIC PCIe bandwidth
  - 14 exhaust NIC Processing Units
  - 28 exhaust NIC cache

- Husky victim
  - synthetic workloads (20) that are sensitive to different microarchitecture resources

Evaluating Existing Solutions

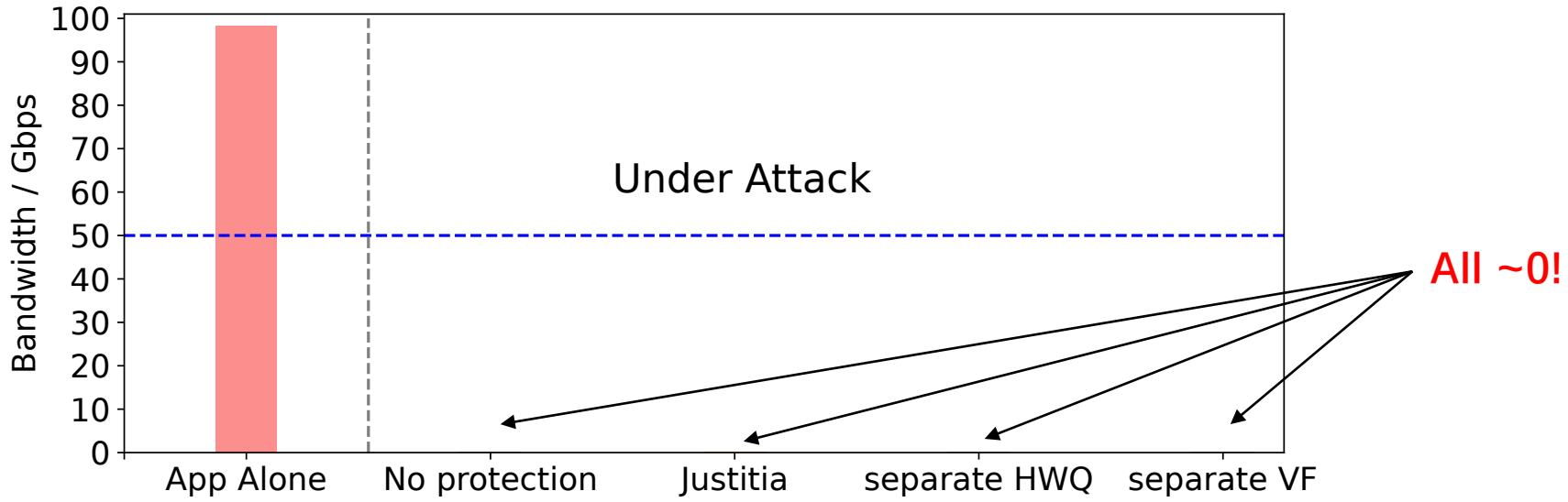
- Separate Hardware (HW) Traffic Class (TC)
  - Separate bandwidth and packet buffers
- SR-IOV
  - Separate virtual function (VF) with separate resources
- Justitia (NSDI ‘22)
  - sender admission control with data verbs pacer

Example - Synthetic Victim

- Attacker – Receive-Not-Ready PU attack
- Victim – RDMA WRITE benchmark
- Configure all approaches to enable ***fairly share*** (100G infra).
  - Each tenant is expected to have 50 Gbps capacity.

Example - Synthetic Victim

- Attacker – Receive-Not-Ready PU attack
- Victim – RDMA WRITE benchmark
- Configure all approaches to enable ***fairly share*** (100G infra).



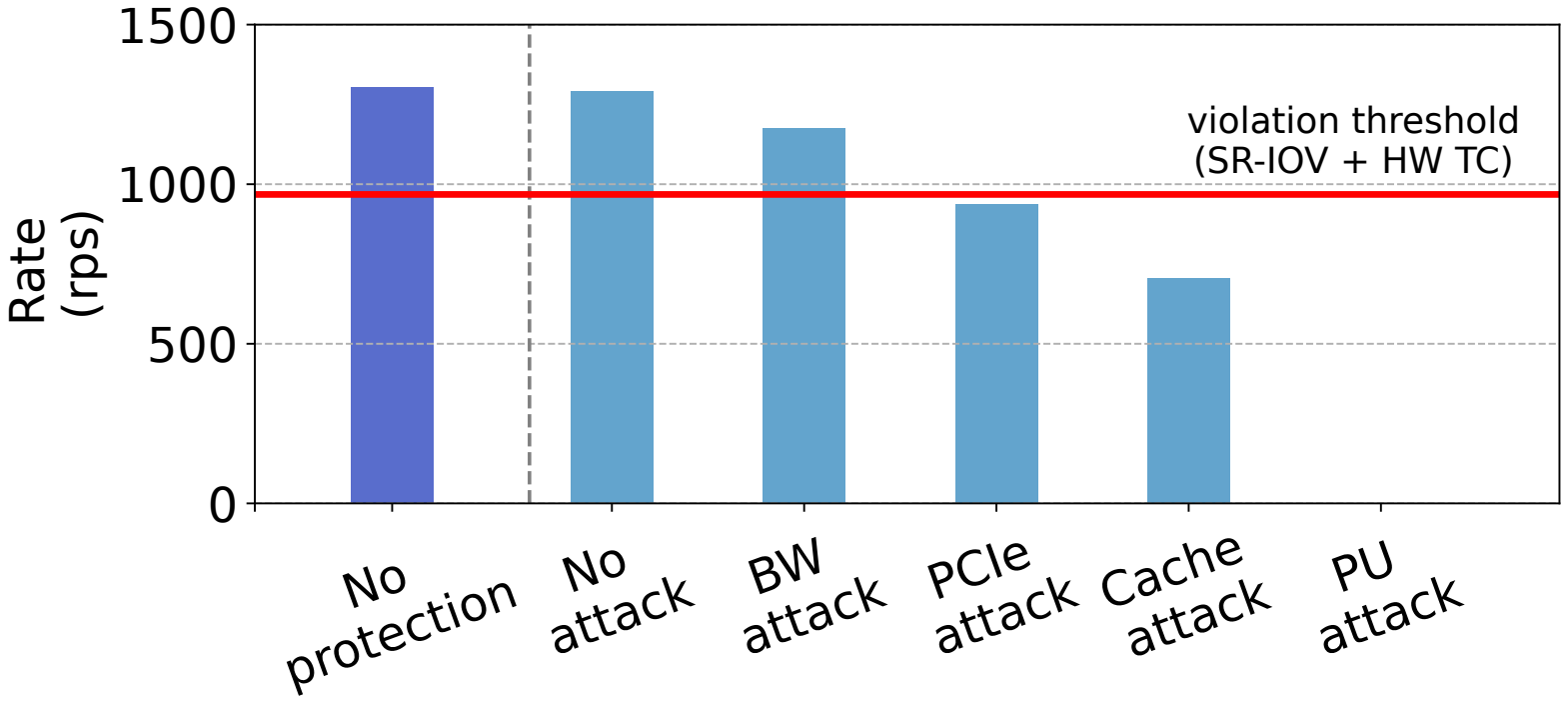
Example – Real Application + Combined Isolation

- Real applications – RDMA-based Allreduce [1]
  - Less than 50Gbps bandwidth -> performance should not be reduced.

- Isolation configuration
  - SR-IOV + HW TC
  - Victim uses TC 0 + VF 0
  - Attacker uses TC 1 + VF 1

[1] OSU benchmarks (https://mvapich.cse.ohio-state.edu/benchmarks/)

Example – Real Application + Combined Isolation



Calculated threshold based on isolation settings

[1] OSU benchmarks (https://mvapich.cse.ohio-state.edu/benchmarks/)

Lessons and Implications

- RDMA performance isolation is more than isolating only bandwidth and packet rate

- Targeting ***microarchitecture resources*** violates isolation easily
  - Less than ~1Gbps bandwidth
  - Or even no bandwidth at all!

- Application sensitivity for resource contention is different
  - Write RDMA app as a public cloud tenant?

Impacts

- For hardware vendors
  - Our findings are all acknowledged and reproduced.
  - We are working with vendors
    - ***Nvidia has already rolled out multiple firmware upgrades.***
- For NIC users
  - Cloud providers are using our test suite.

- Our tool is general – please try it if you want to test your RNIC!

Conclusions

- Understanding microarchitecture resources is important for RDMA performance isolation.

- Husky: first test suite that evaluates performance isolation solutions by looking into microarchitecture resource contention.
  - Include sets of benchmarks target microarchitecture resources contention
  - Evaluate existing solutions with synthetic workloads and real applications
  - *Sets a higher bar for RDMA solution – public cloud deployable*
  - https://github.com/host-bench/rdma-bench