Enabling Memory-intensive Network Functions on Programmable Switches

Daehyeok Kim

Zaoxing Liu, Yibo Zhu, Changhoon Kim, Jeongkeun Lee, Antonin Bas

Vyas Sekar, Srini Seshan

Carnegie Mellon University

Microsoft

BAREFOOT NETWORKS

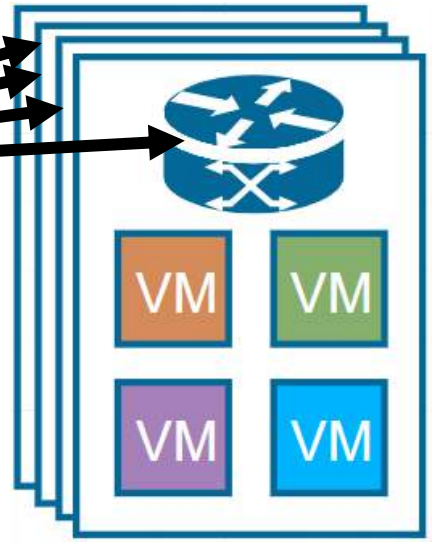
Example: Enabling DC Scale Virtual Switching on ToR Switch

Multi-million Entries
≫ SRAM size!

(Tenant, VM IP)	Host IP
(1, 20.0.0.1)	10.0.0.1
(1, 20.0.0.2)	10.0.1.1
(1, 20.0.0.3)	10.0.2.1
…	…

Move virtual switch to ToR switch

Programmable Switch



Customers' Bare-metal servers

Cannot install virtual switches on the servers

Limited SRAM space is bottleneck for memory-intensive applications!

Status Quo

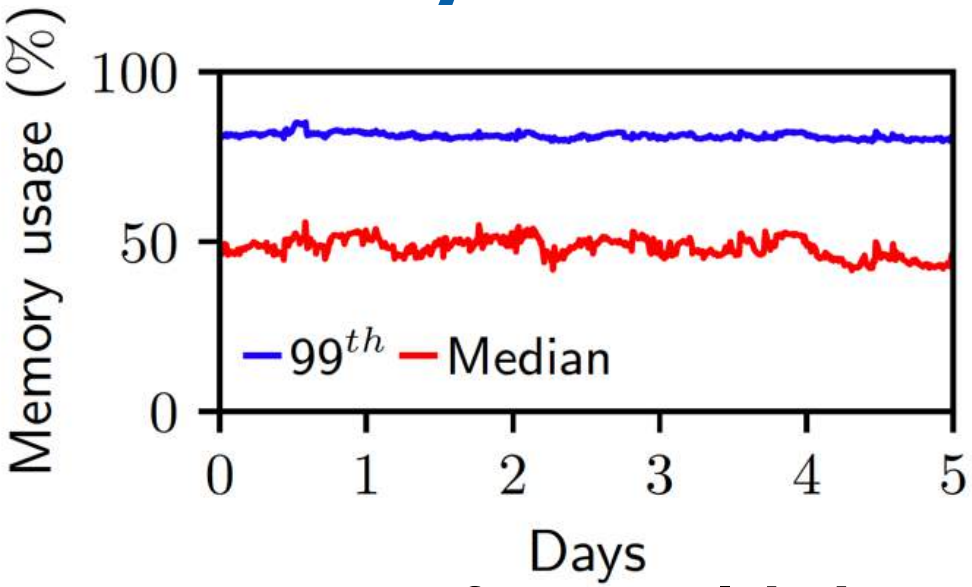
- ***Fixed-function* switch chips built with *fixed-function* external memory**

- **These aren't very useful**
  - **Inflexible**: Usage fixed at design time
  - **Fixed and small scale**: Memory size and bandwidth fixed at design time
  - **Expensive**: Chip getting larger and complex

Is programmable switch chip + general-purpose memory possible?

Opportunities

1. Underutilized servers' memory

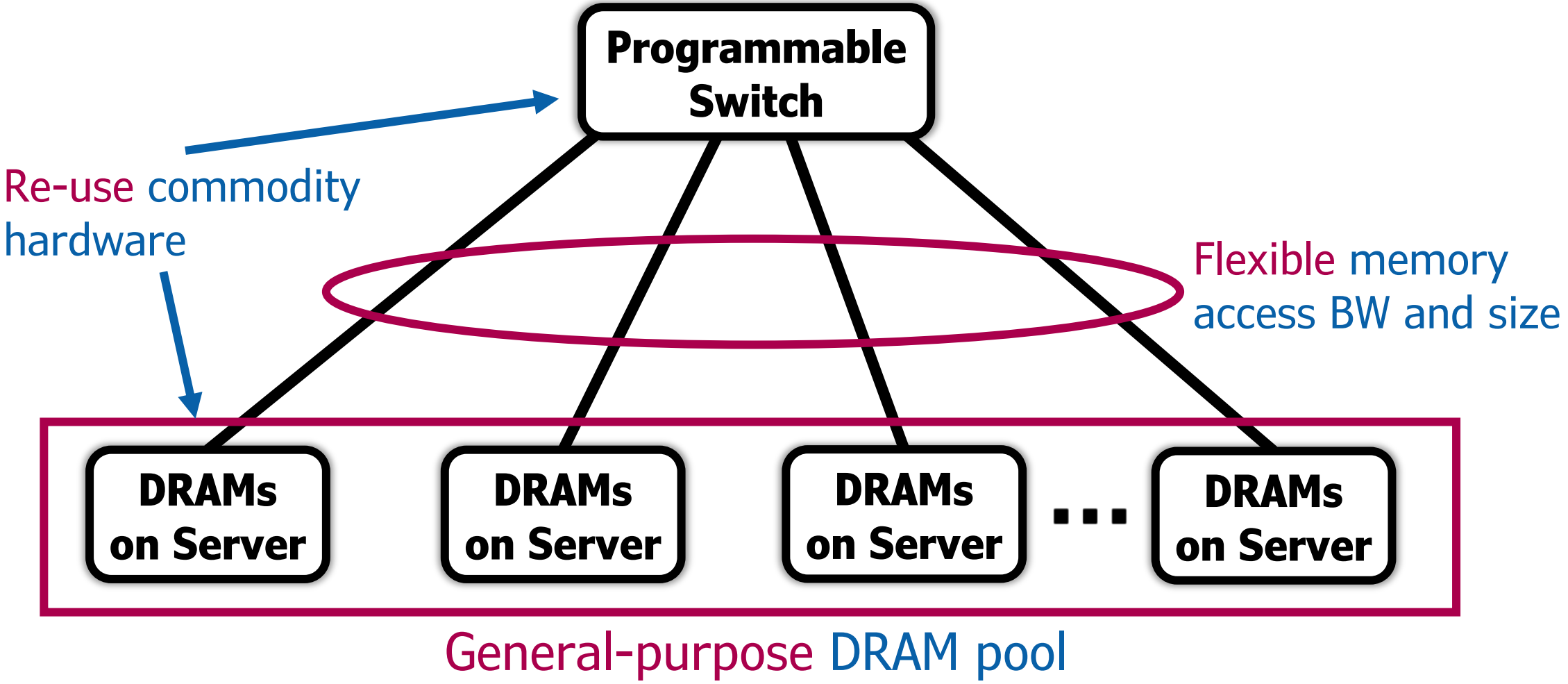


Memory usage trace from Google's data centers

2. Underutilized network bandwidth

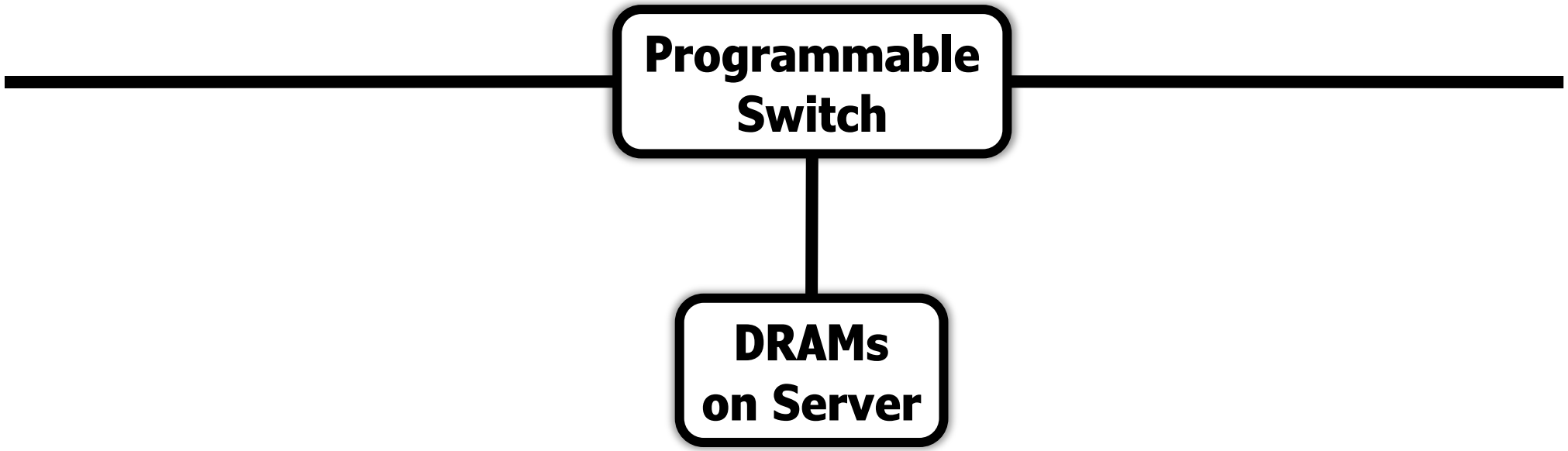
- In a production data center network, ToR switch-server link utilization is ~1% on average and ~10% in mostly loaded case.

Our Work: Generic External Memory (GEM)

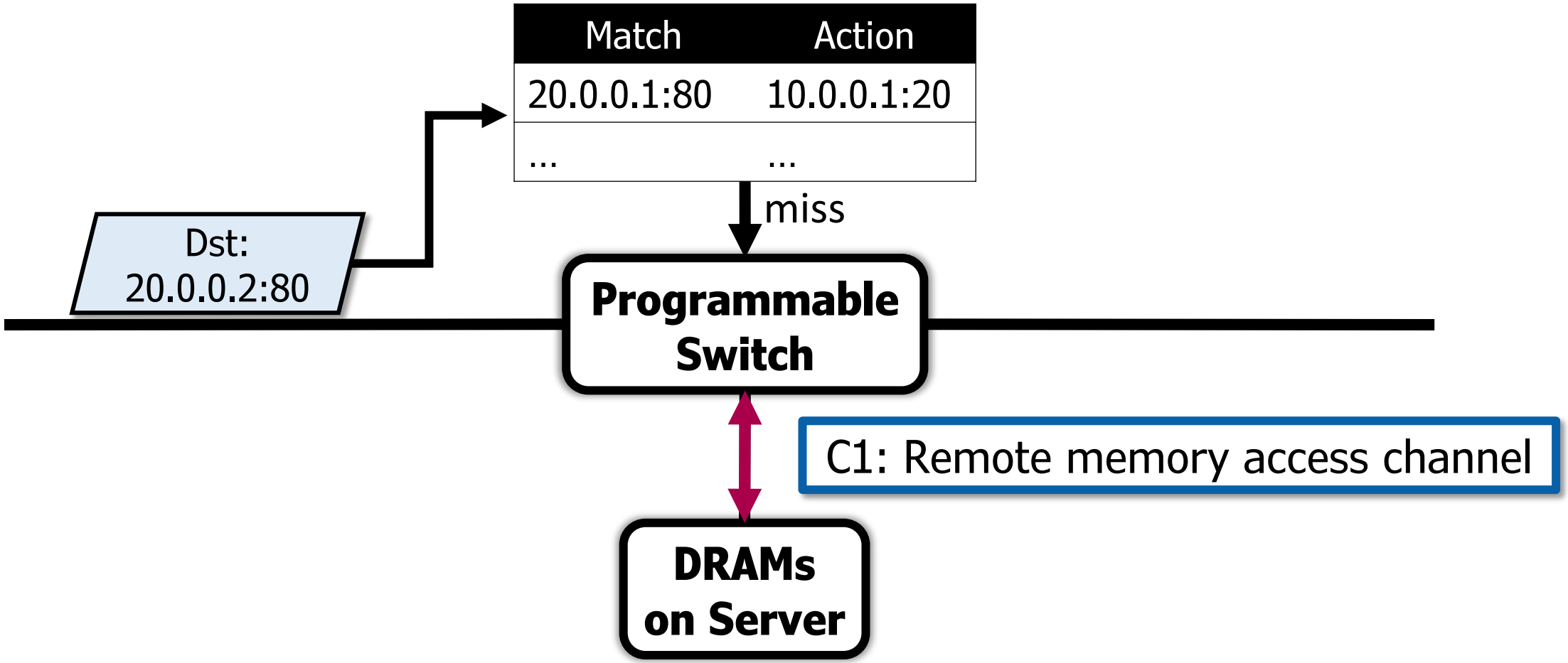


Key Challenges

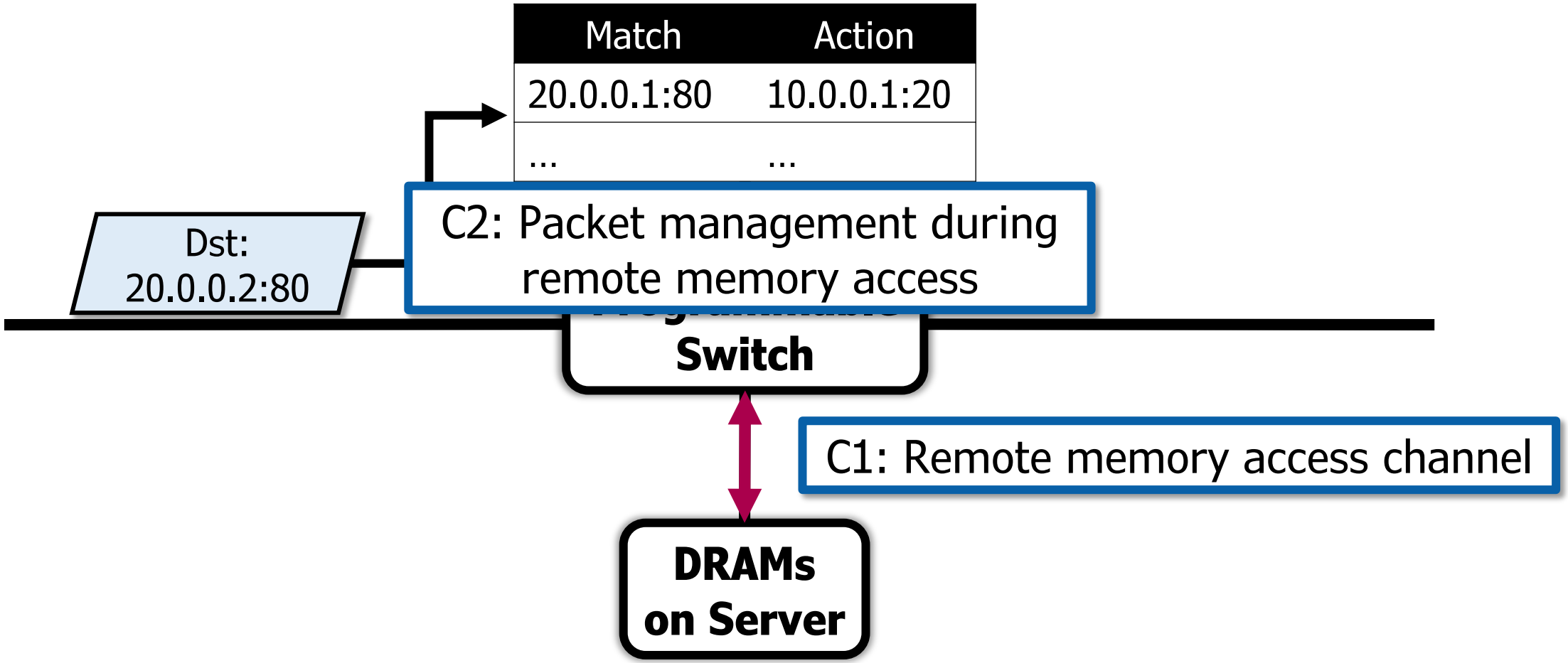
Match	Action
20.0.0.1:80	10.0.0.1:20
…	…



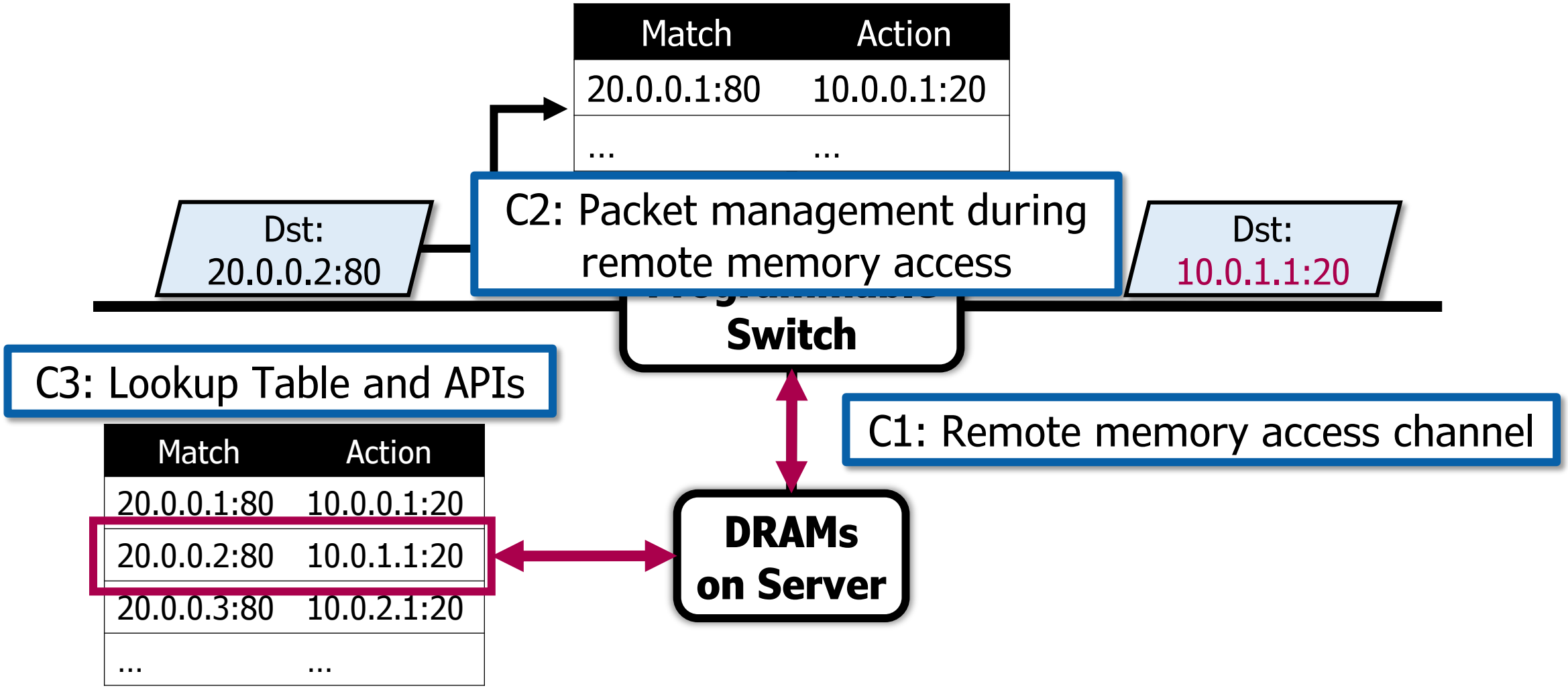
Key Challenges



Key Challenges



Key Challenges



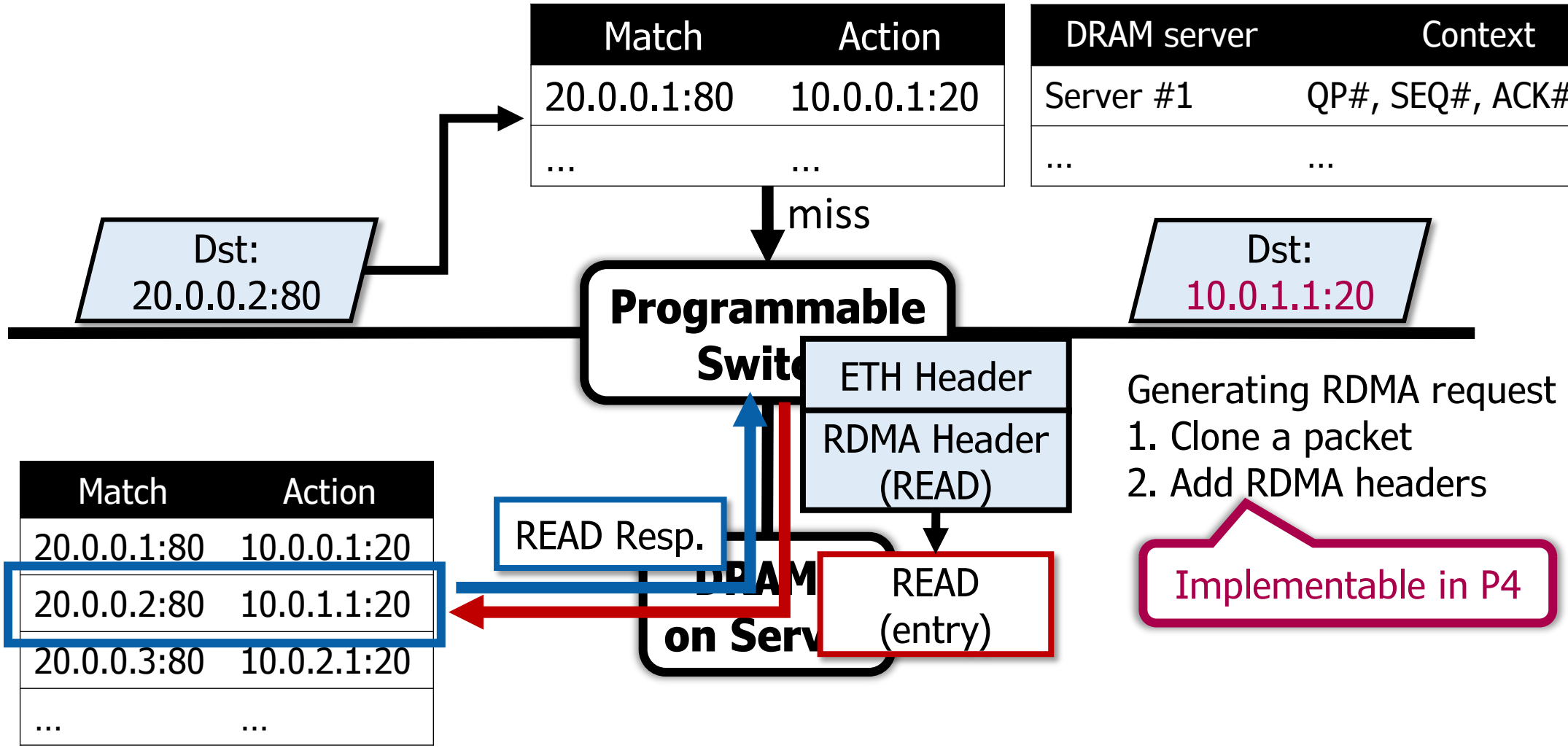
C1: Remote Memory Access Channel

- **Goal: Enable programmable switch chip to directly access memory**
  - **Purely access DRAM:** No impact to the server's existing compute and networking workloads
  - **Minimal latency** between the chip and memory

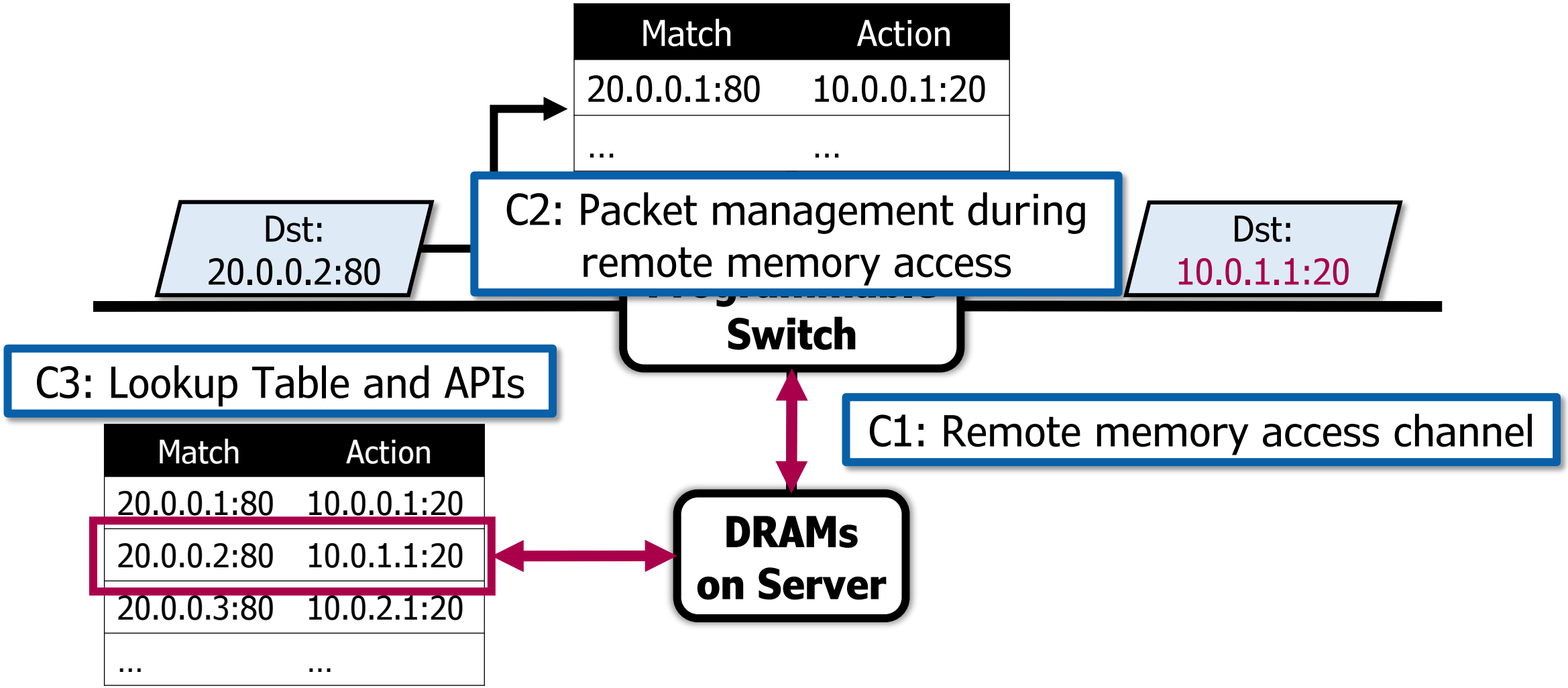
Leverage RDMA!

- **Challenge: How to generate RDMA requests from the data plane?**

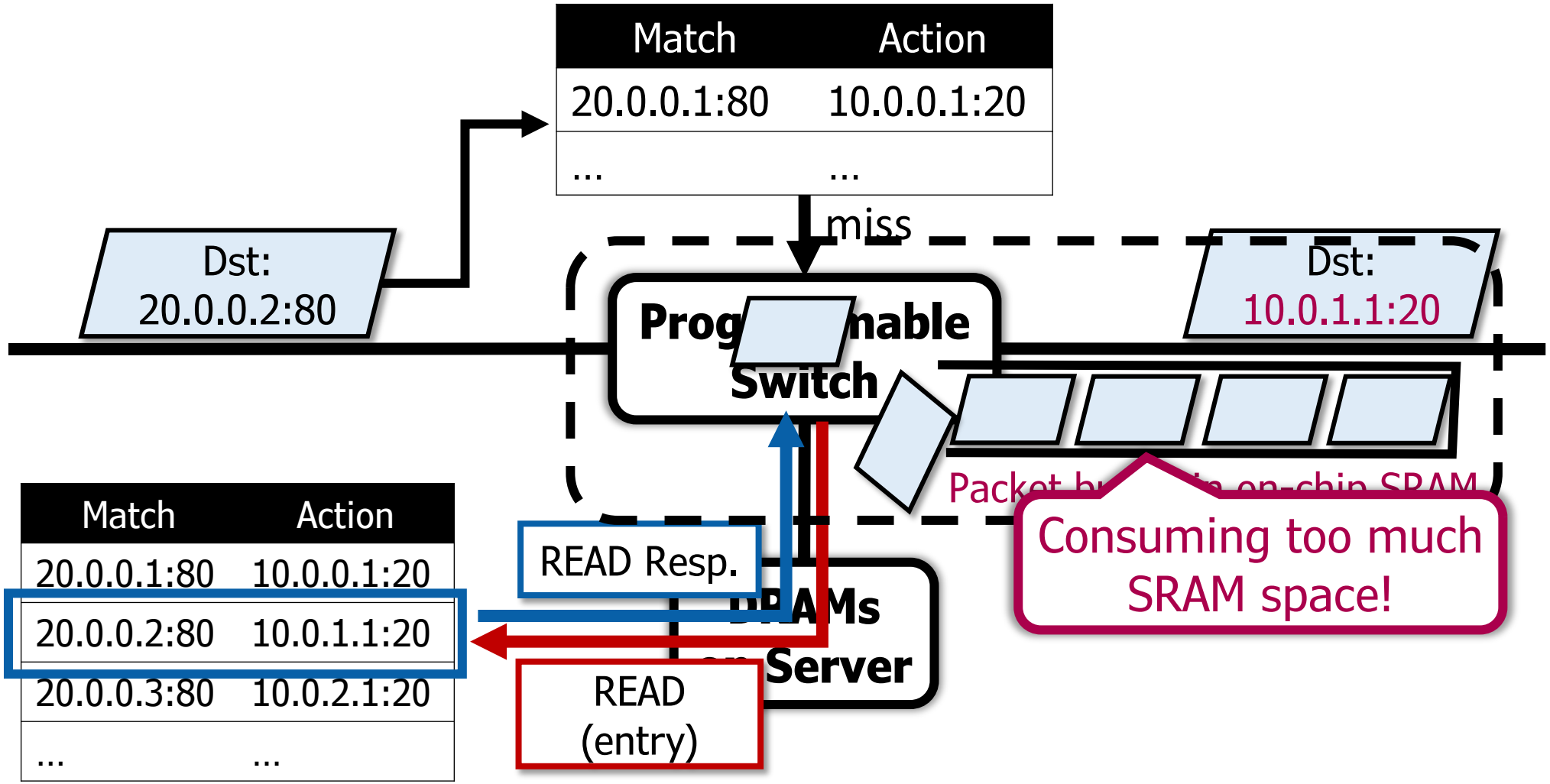
Accessing Remote Memory via RDMA



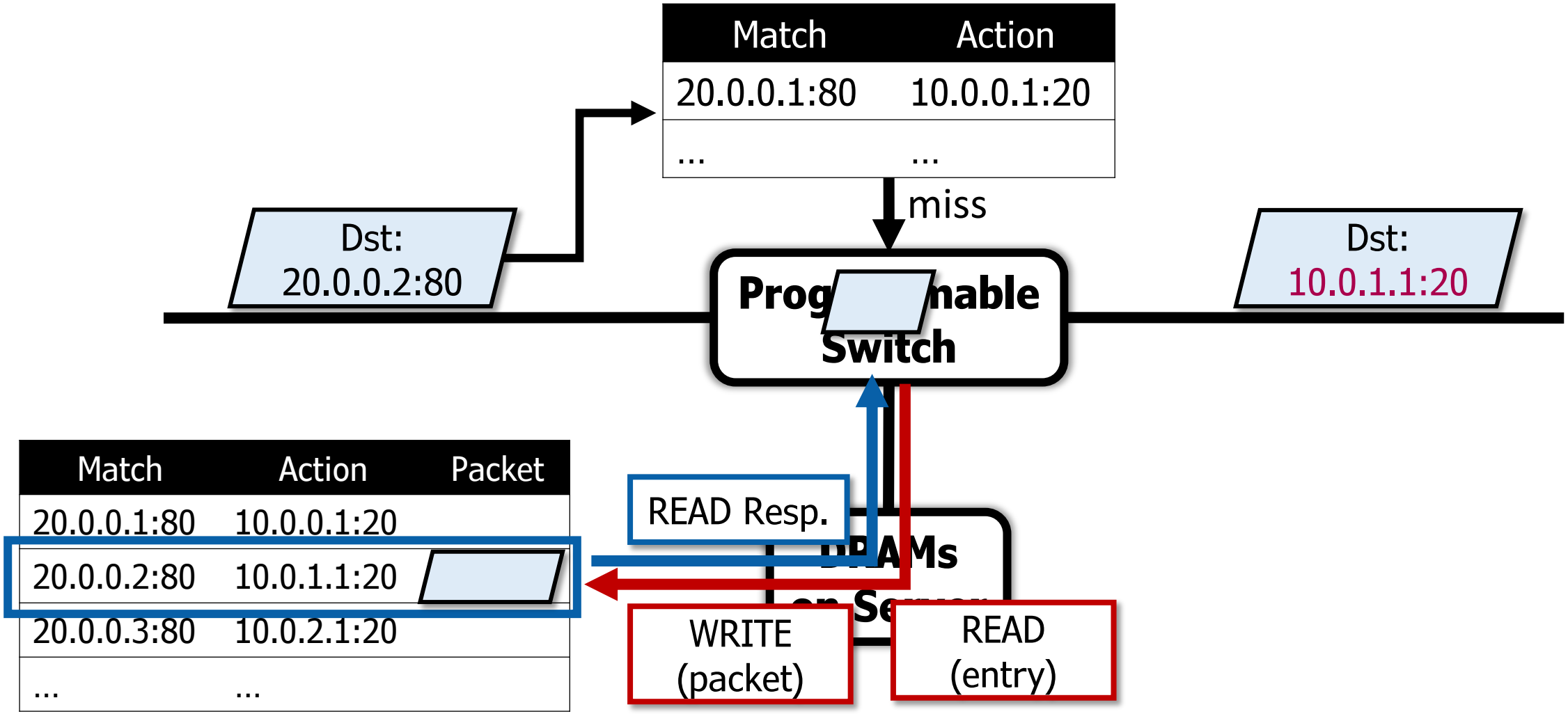
Key Challenges



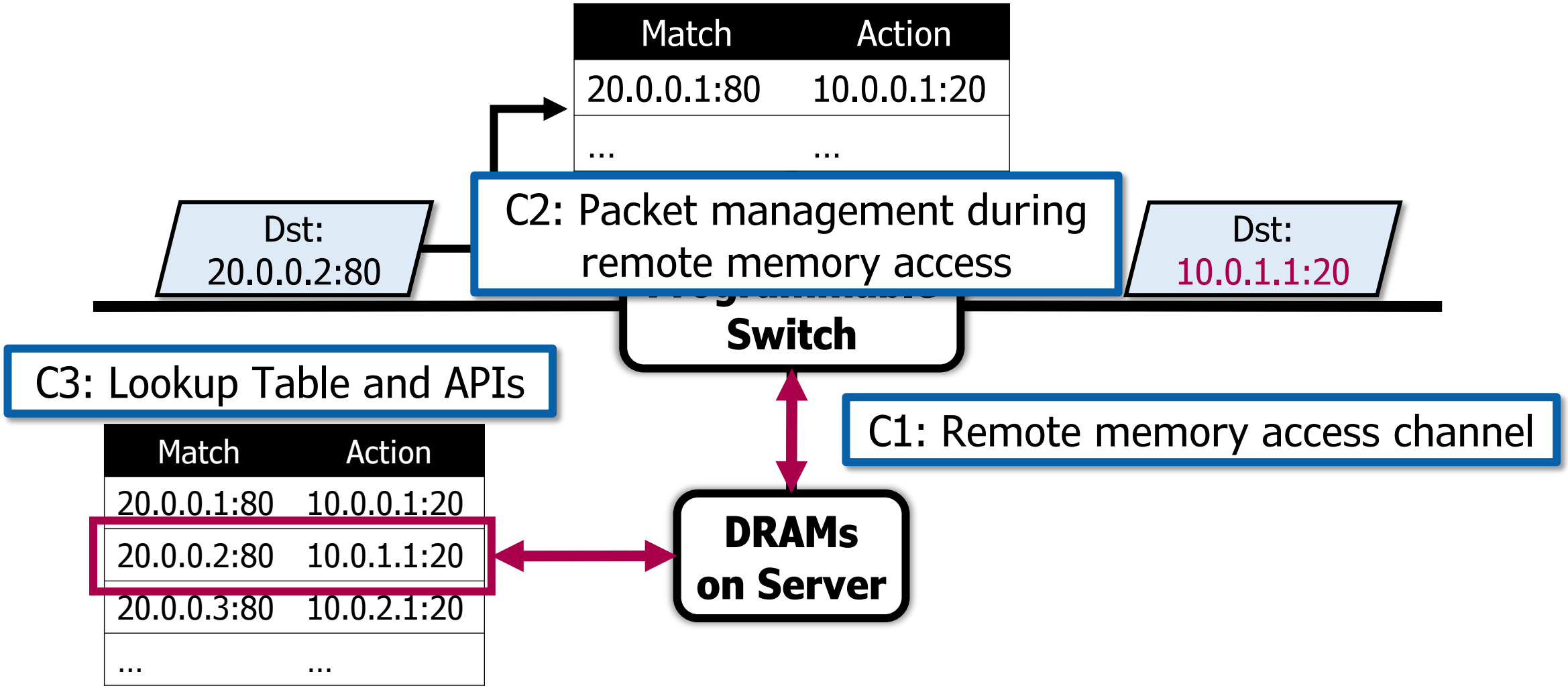
C2: Packet Management during Remote Memory Access



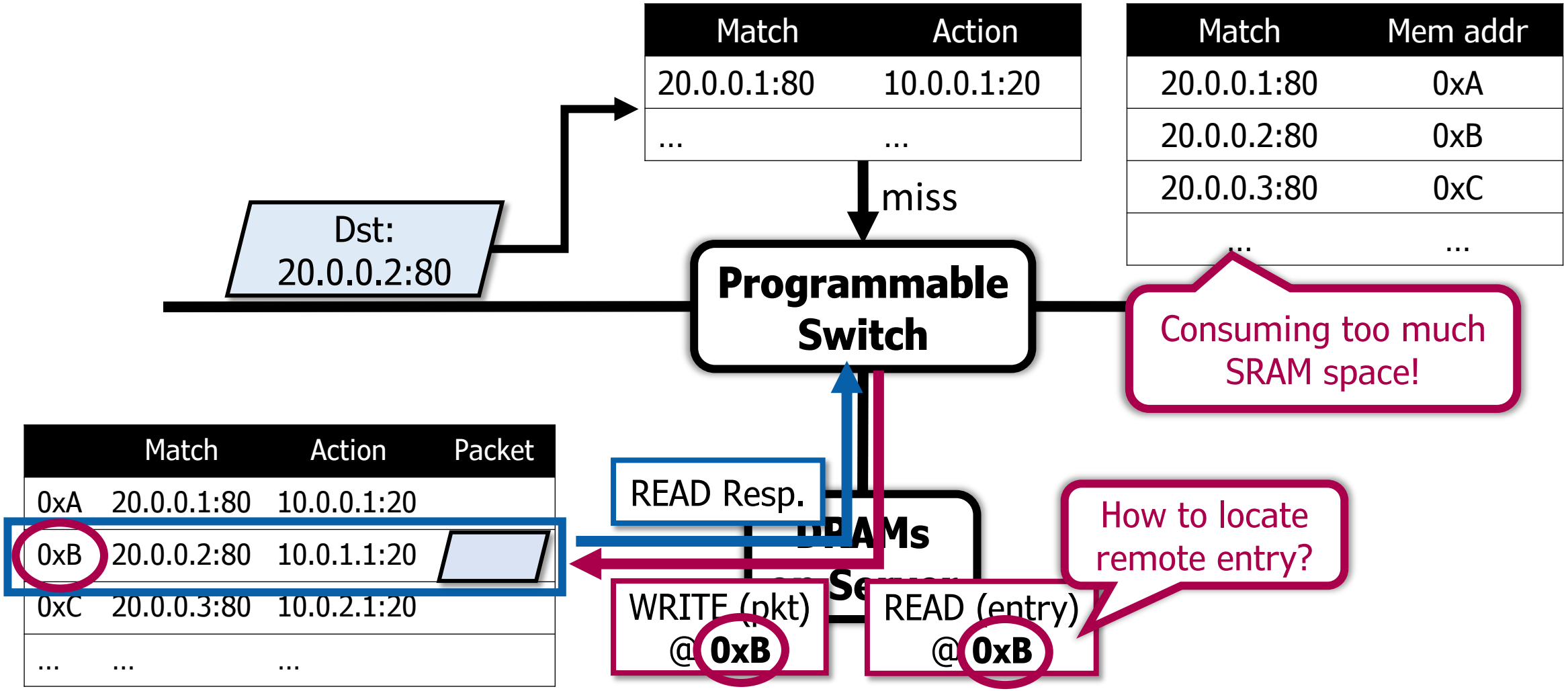
Depositing Packets on Remote Buffer



Key Challenges

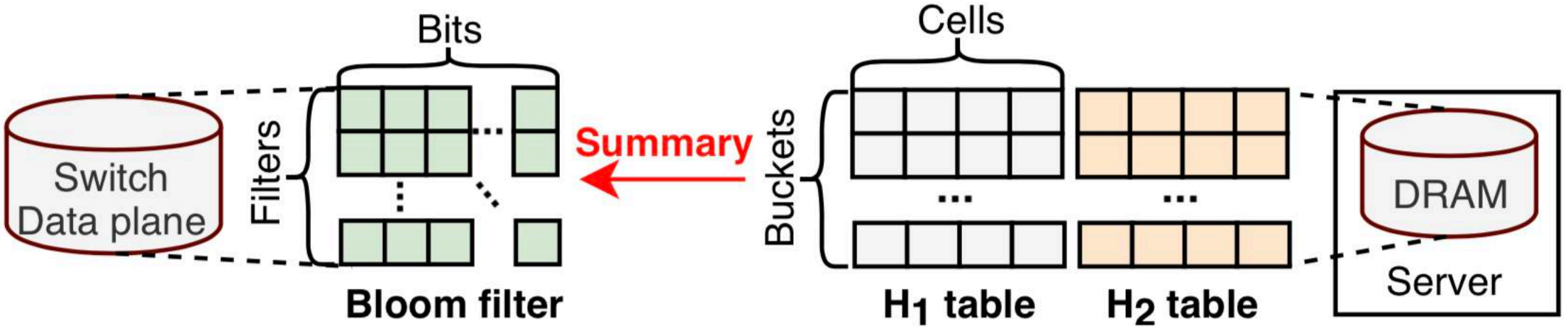


C3: Remote Lookup Table and APIs

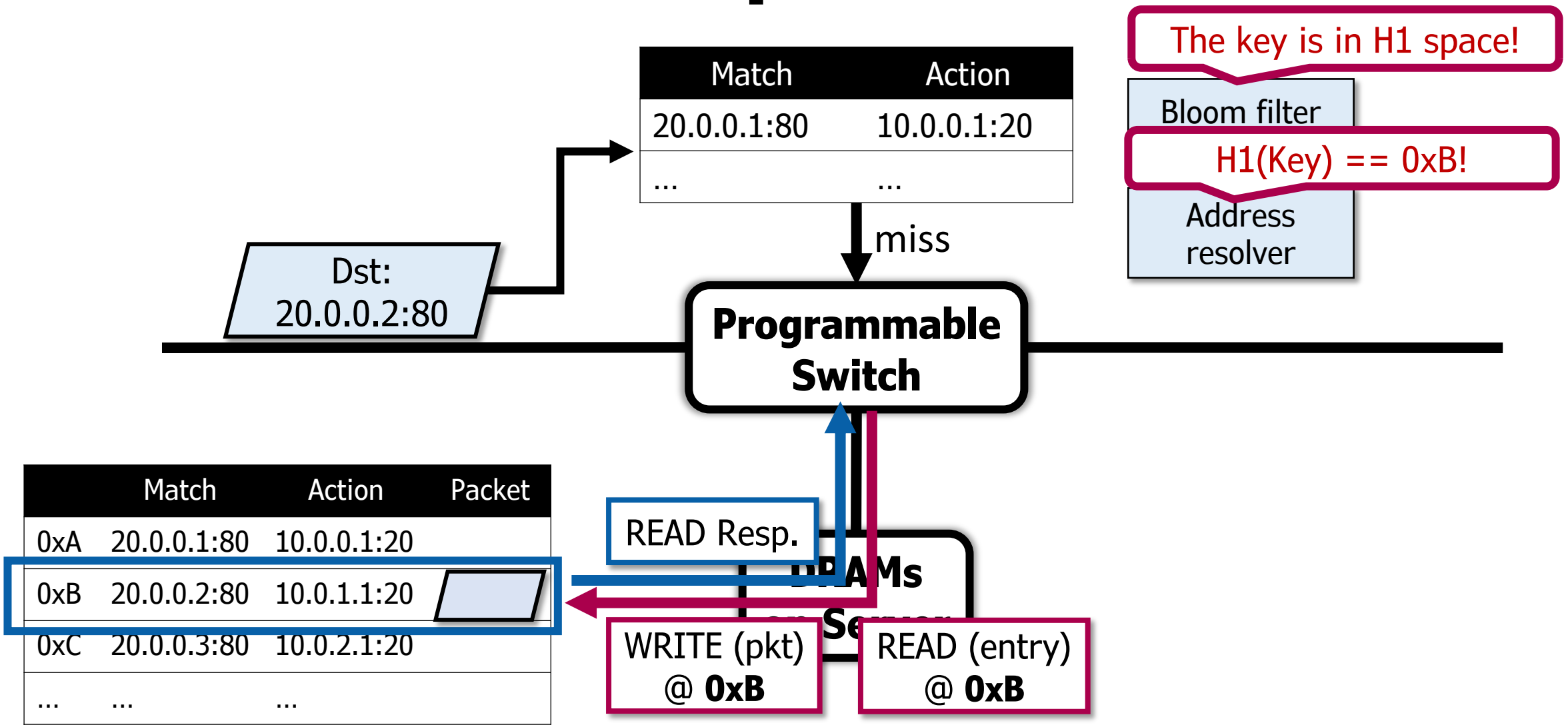


GEM Lookup Table

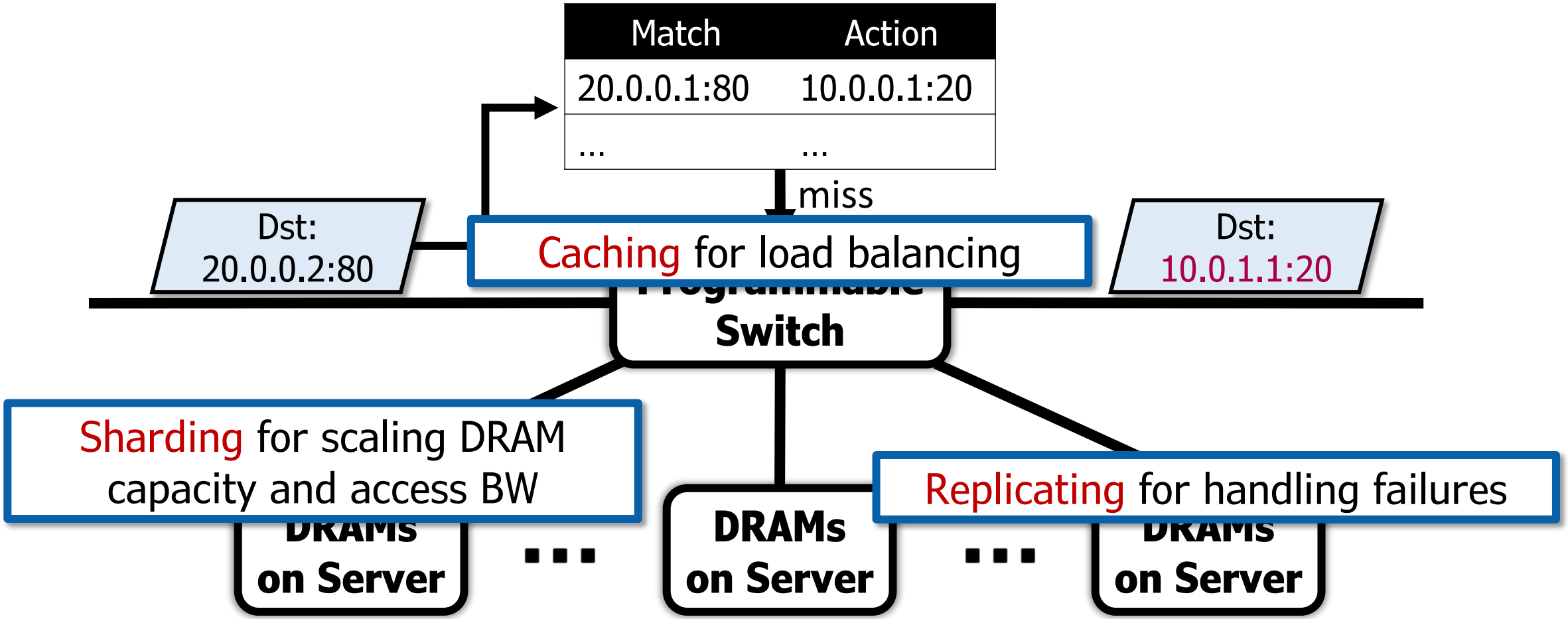
- **Two-way cuckoo hash table guaranteeing one memory access per lookup**
- **Idea:** Maintain a "summary" of which table the entries are stored in.
  - Summary(key) tells you which table (*H1* or *H2*) contains the key.



GEM Lookup Table in Action



GEM with Multiple Servers



Implementation and Results

- **Prototype implementation**
  - Tofino-based programmable switch + 8 servers with RDMA NICs
- **End-to-end latency overhead**
  - 2.4 – 2.75 µs added latency (in NAT case) due to external memory lookup
- **Lookup throughput**
  - Single server with caching: 20 millions lookup per second
  - Scalability: Achieving near line-rate (i.e., traffic rate to NFs) with four servers

Summary

- **Limited memory space on programmable switches is problematic.**

- **Our solution:** **G**eneric **E**xternal **M**emory for Programmable Switches
  - Allows programmable switches to access external DRAM
  - Provides an *external memory abstraction* to developers
  - Can be applied to existing NF implementations

- **Result:** NFs can utilize lookup tables on external DRAM without much performance degradation (~0.55% throughput reduction).