# Cuttlefish: A Fair, Predictable Execution Environment for Cloud-hosted Financial Exchange

***Liangcheng (LC) Yu***, Prateesh Goyal, Ilias Marinos, and Vincent Liu

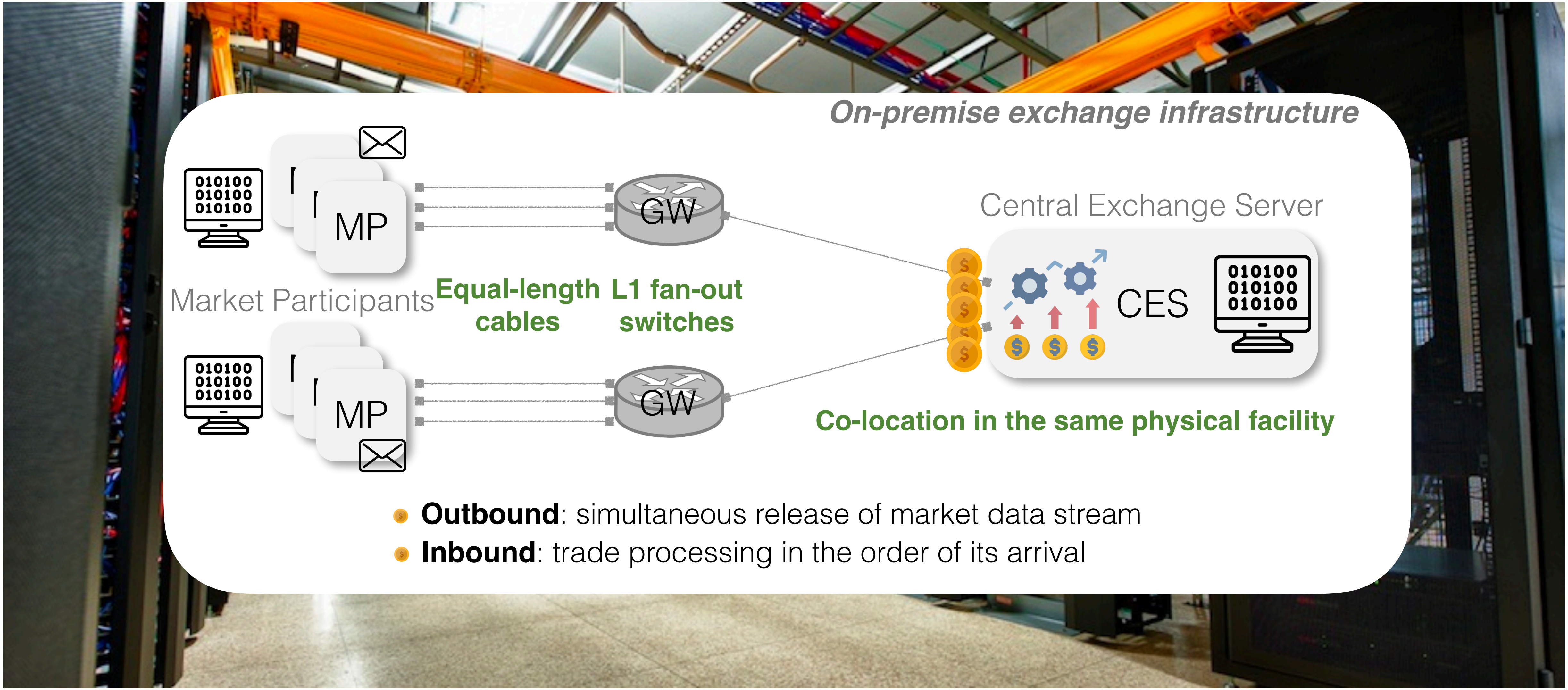
*Advances in Financial Technologies (AFT) 2025*

Microsoft Research

NVIDIA

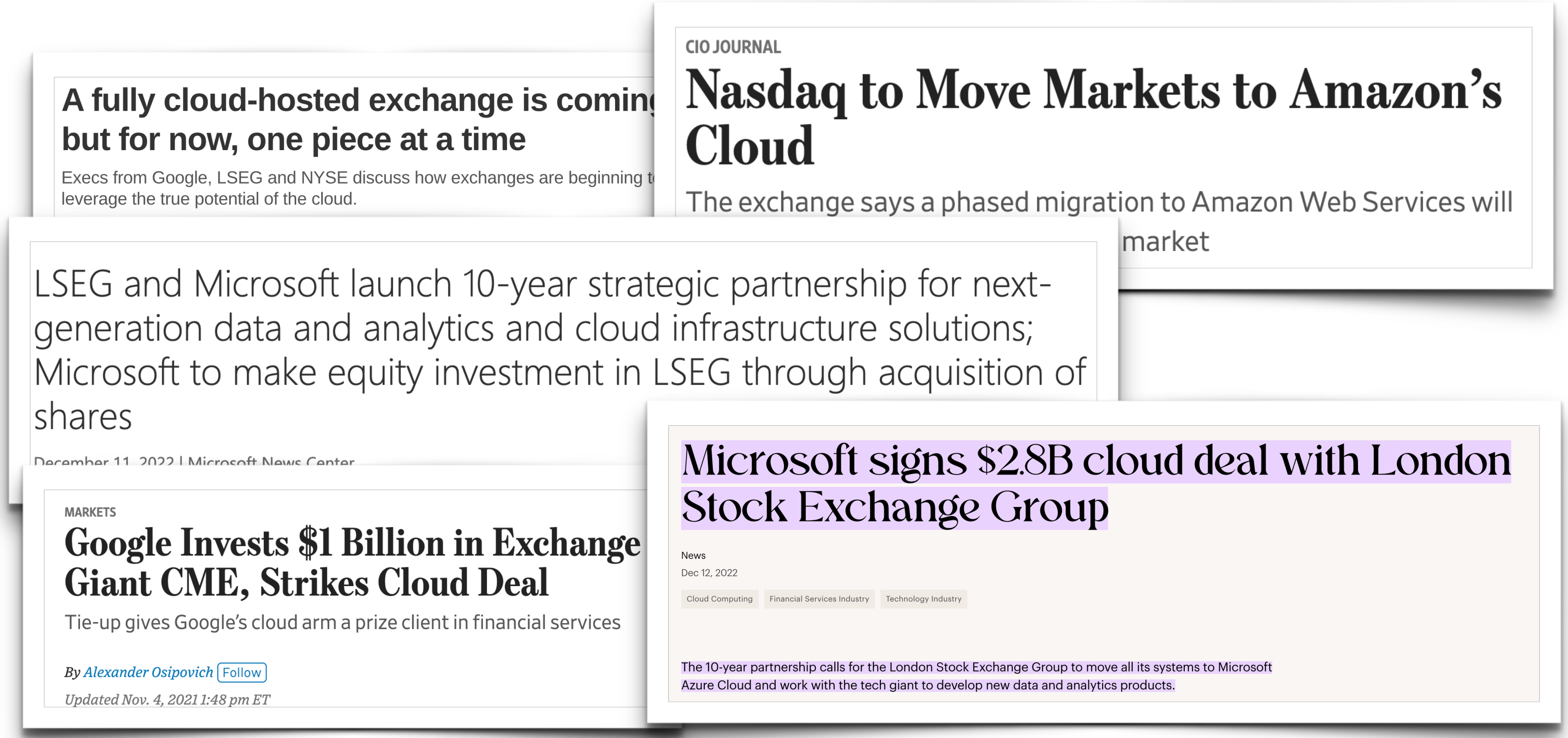
Penn
University *of* Pennsylvania

# Fairness, in on-premise infrastructure




- **Outbound**: simultaneous release of market data stream
- **Inbound**: trade processing in the order of its arrival

# Rising interest in cloud-hosted exchange services




- System scalability and resource elasticity
- Rise of remote work
- Cost reduction and ease of management
- …

# Rising interest in cloud-hosted exchange services

A fully cloud-hosted exchange is comin[g] but for now, one piece at a time

Execs from Google, LSEG and NYSE discuss how exchanges are beginning t[o] leverage the true potential of the cloud.

CIO JOURNAL

Nasdaq to Move Markets to Amazon's Cloud

The exchange says a phased migration to Amazon Web Services will … market

…ft launch 10-year strategic partnership for next-… nd an… … equity investment in LSEG through acquisition of … Center

***Cloud infrastructure can introduce unfairness!***

MARKETS

Google Invests $1 Billion in Exchange Giant CME, Strikes Cloud Deal

Tie-up gives Google's cloud arm a prize client in financial services

By Alexander Osipovich Follow

Updated Nov. 4, 2021 1:48 pm ET

Microsoft signs $2.8B cloud deal with London Stock Exchange Group

News

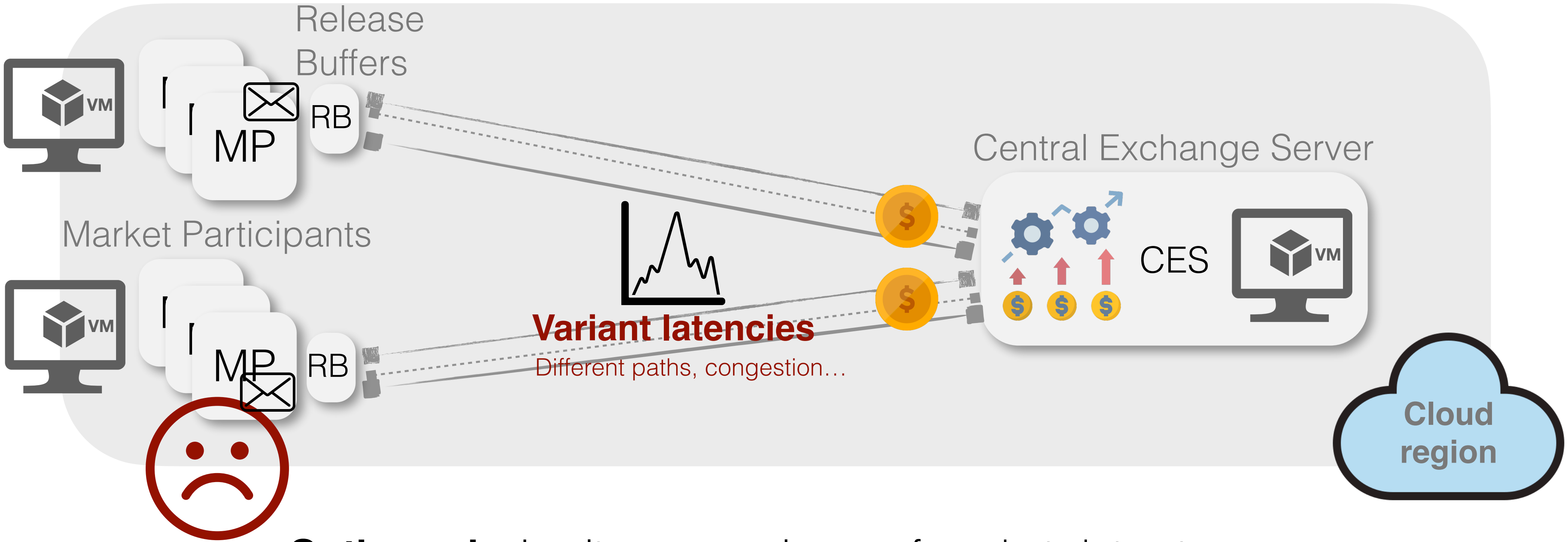
Dec 12, 2022

Cloud Computing Financial Services Industry Technology Industry

The 10-year partnership calls for the London Stock Exchange Group to move all its systems to Microsoft Azure Cloud and work with the tech giant to develop new data and analytics products.

- System scalability and resource elasticity
- Rise of remote work
- Cost reduction and ease of management
- …

# Variances in network latencies

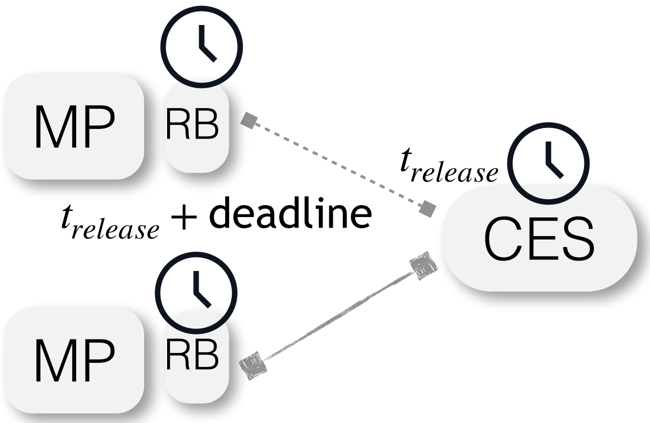



- **Outbound**: simultaneous release of market data stream
- **Inbound**: trade processing in the order of its arrival

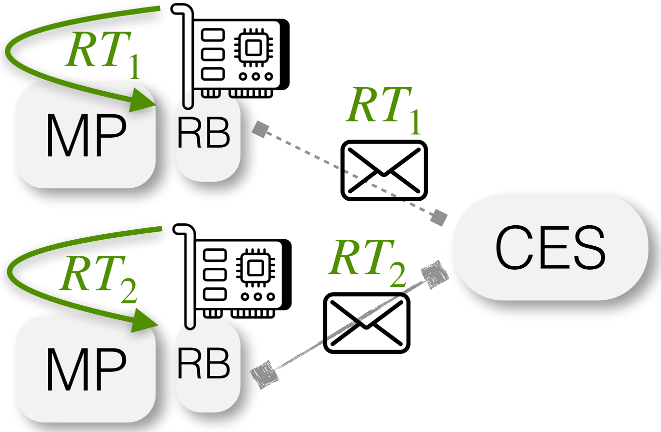
**Unfairness!**

# Efforts toward communication fairness

Clock synchronization
**(CloudEx, HotOS '21)**




Logical clock based on response time (RT)
**(DBO, SIGCOMM '23)**




Perfect clock synchronization is **hard**

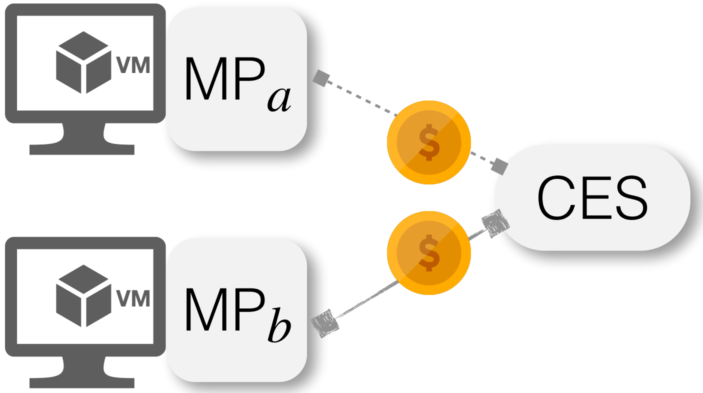
Hard to pre-determine the deadline

**Limited to trigger-point based trades**

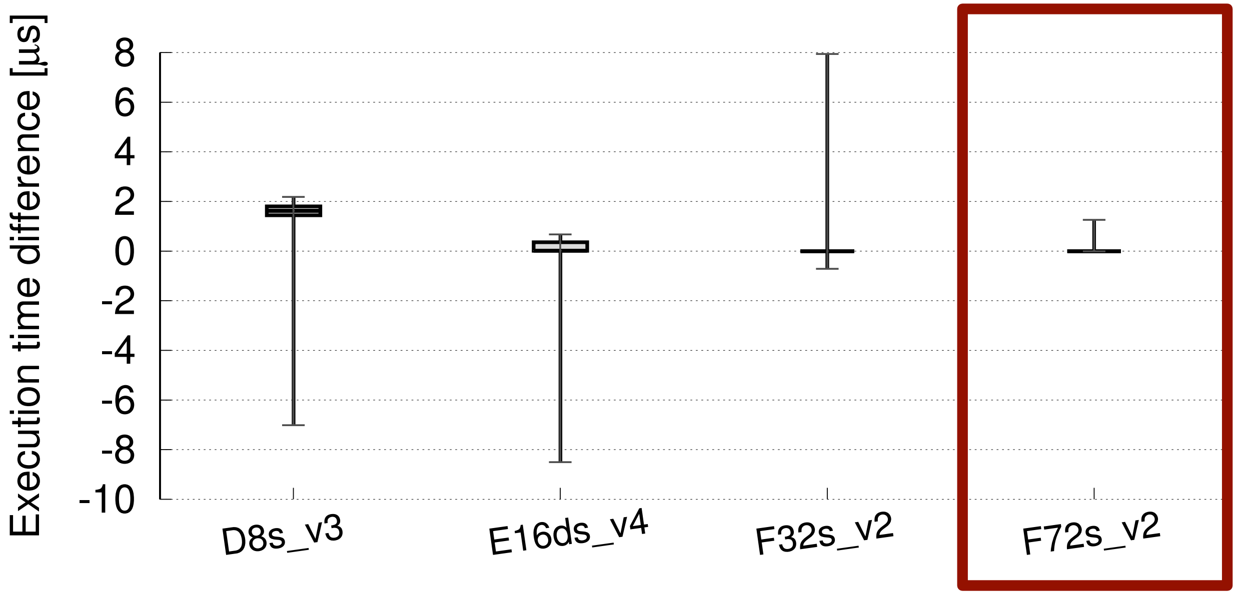
**Doesn't handle MP-RB latency variances**

# …cloud execution can also incur unfairness!

- Other sources of unfairness: noisy neighbors, thermal conditions of the processors…

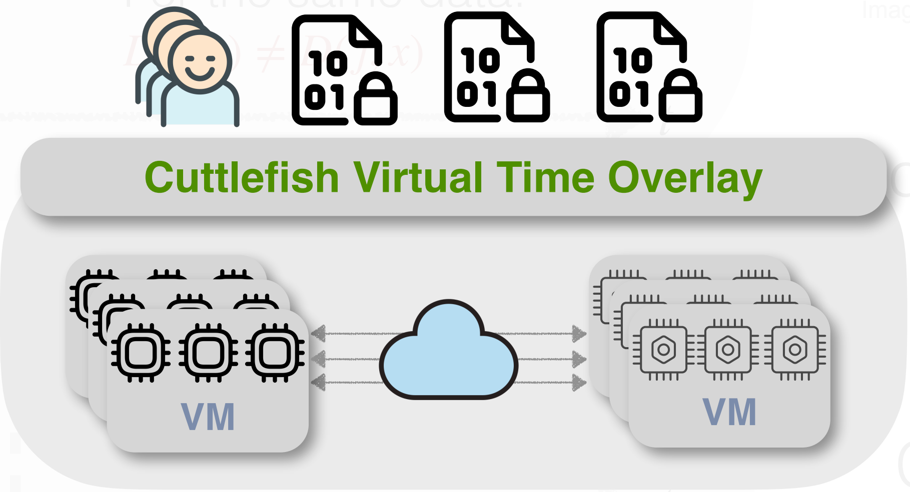


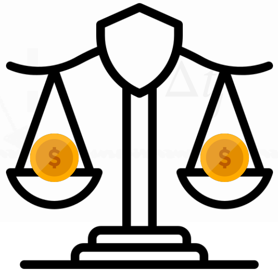

**Identical programs running on same types of VMs**



# Can we **eliminate variations** that come from the cloud infrastructure?

## Cuttlefish: A Fair, Predictable Execution Environment



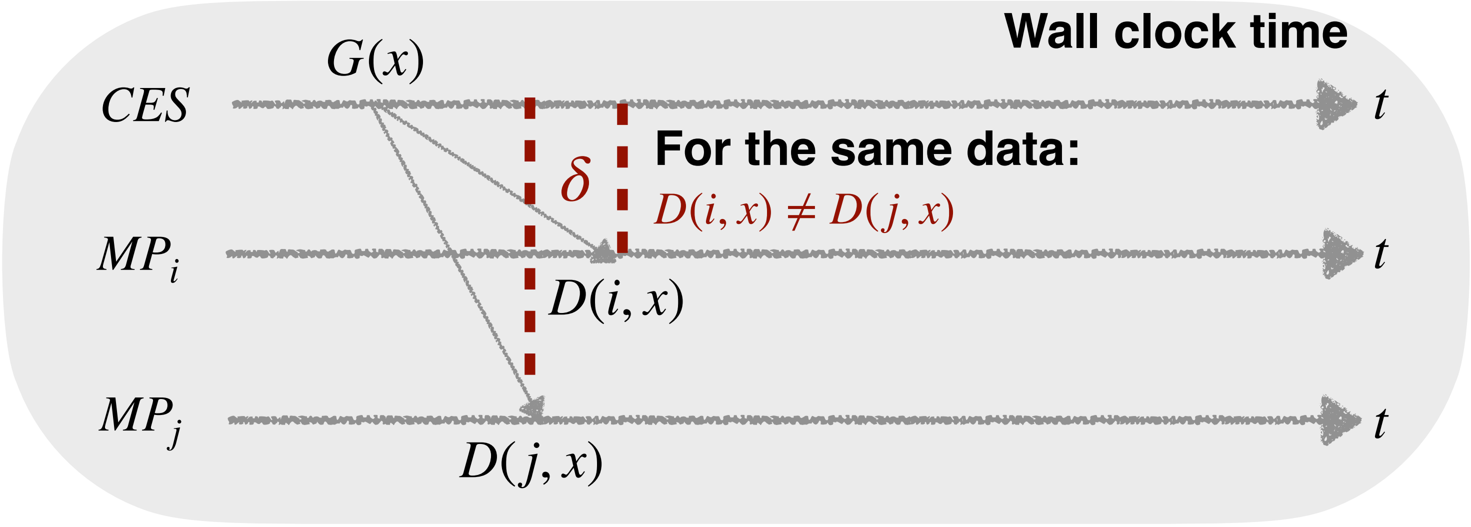



***Abstraction***
- Equal cloud networks
- Equal execution hardware
- …

# Outline

- **Conceptual foundation**
- Implementing virtual time overlay
- Evaluation

# Let's reflect on underlying model today…



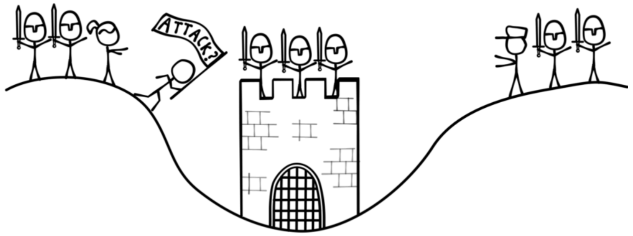
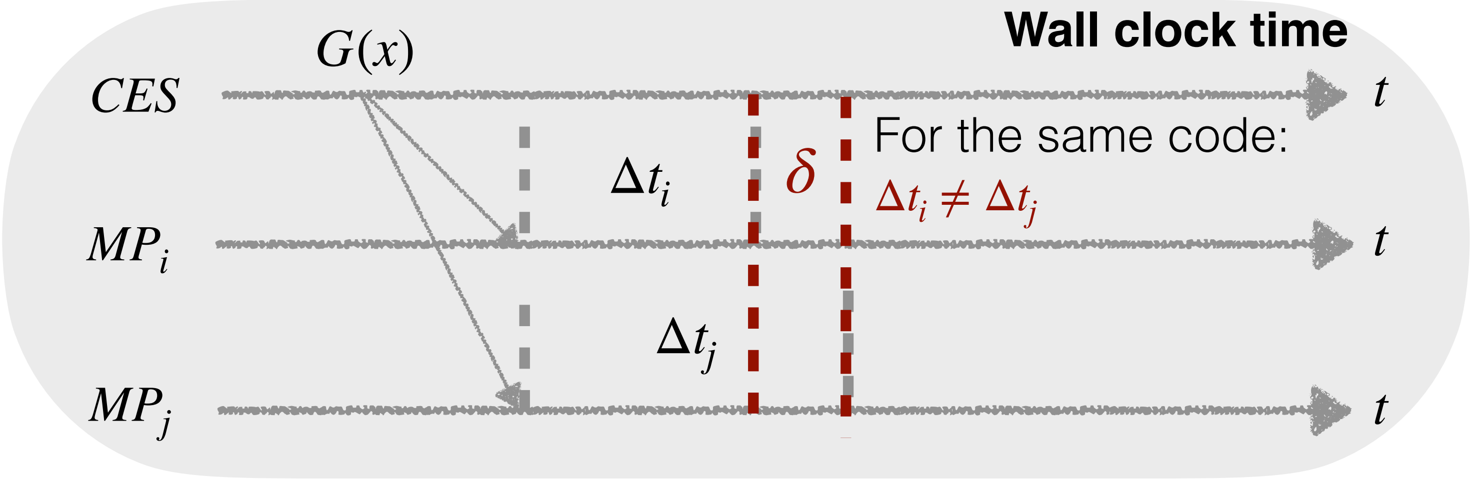



Image source: https://haydenjames.io/the-two-generals-problem/

Simultaneous delivery is ***infeasible***!




Execution time can be ***non-deterministic*** at $O(\mu s)$ (thermal condition, resource utilization…)

# Let's reflect on underlying model today…

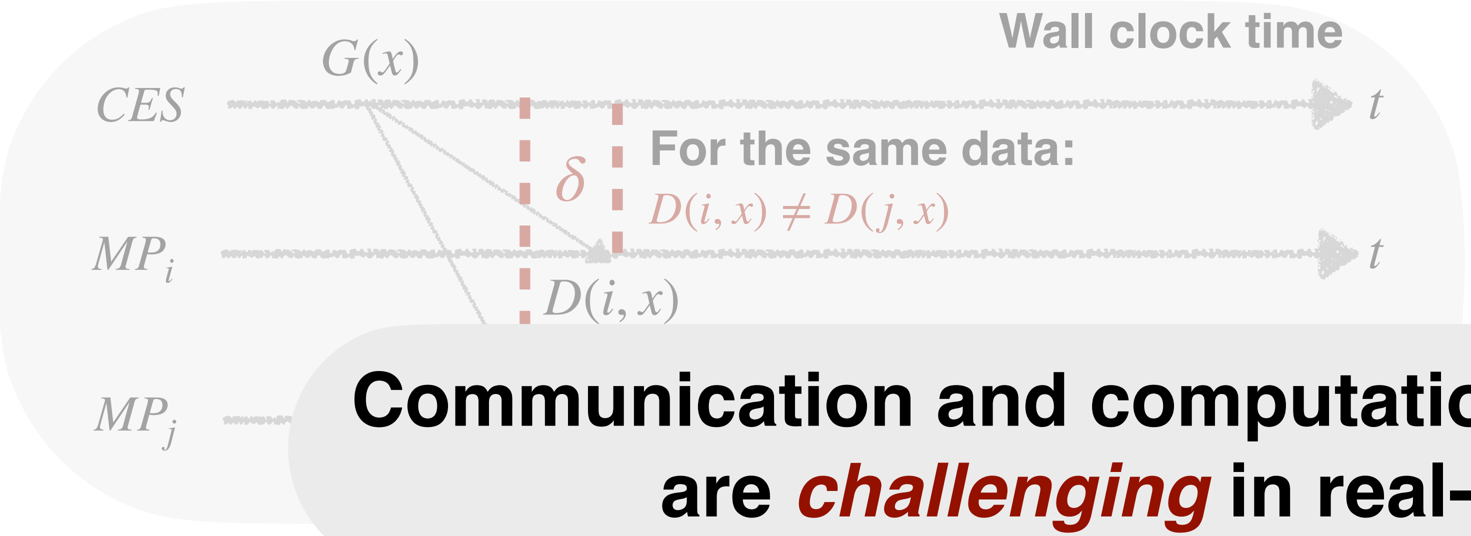
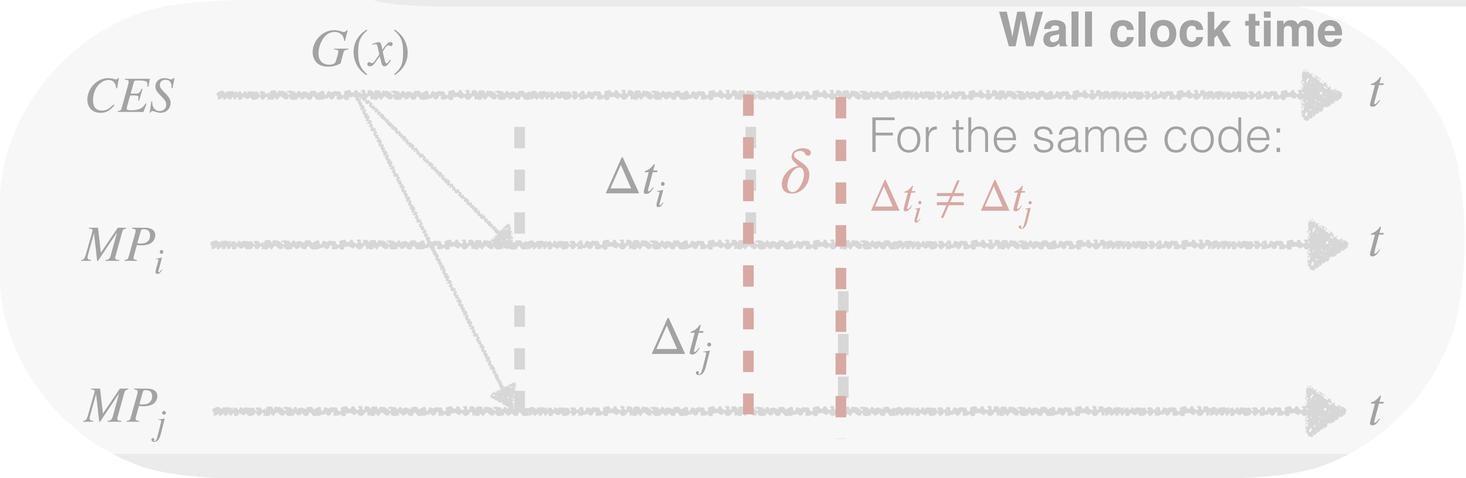




Image source: https://haydenjames.io/the-two-generals-problem/

**Communication and computation synchrony are *challenging* in real-time**

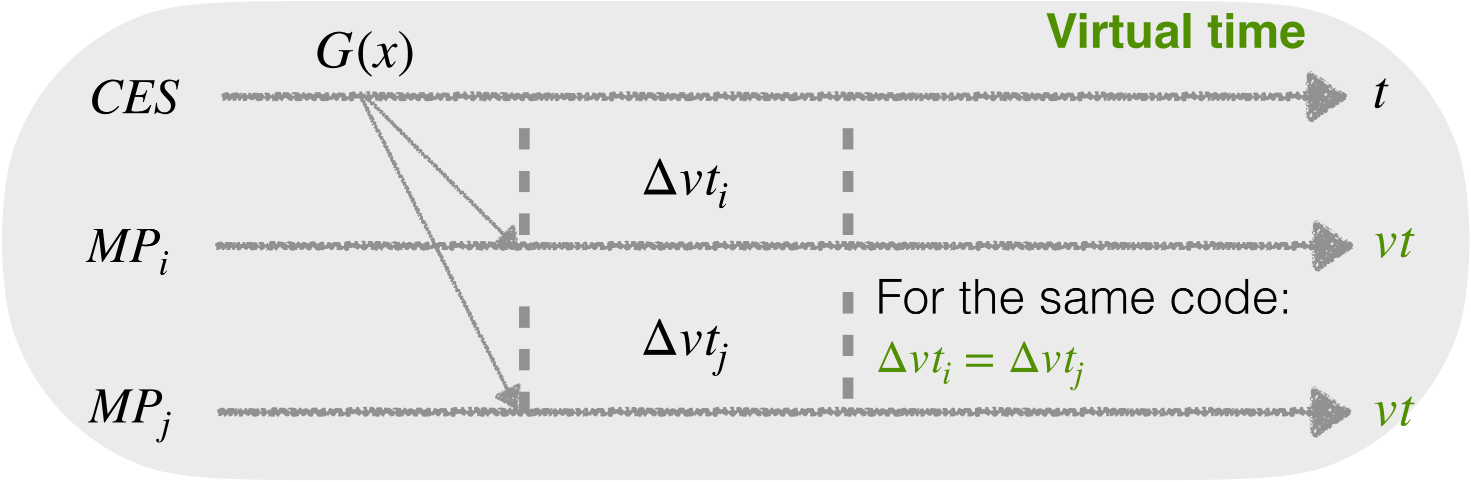
delivery
*ible*!




Execution time can be ***non-deterministic*** at $O(\mu s)$ (thermal condition, resource utilization…)

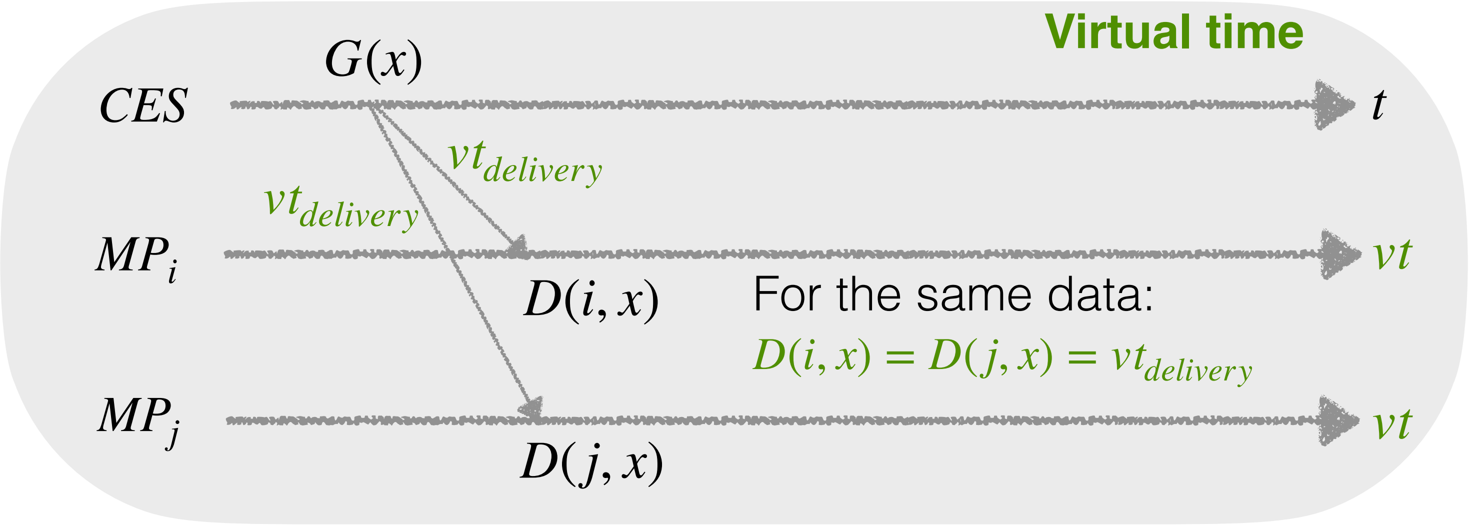
# Let's try virtual time domain ···

*Virtual time unit* ≡ *some equal amount of work*




**Execution synchrony:**

Advancing virtual time per ***'actual amount of work'***




**Communication synchrony:**

Releasing data to MPs at the ***same virtual delivery time***

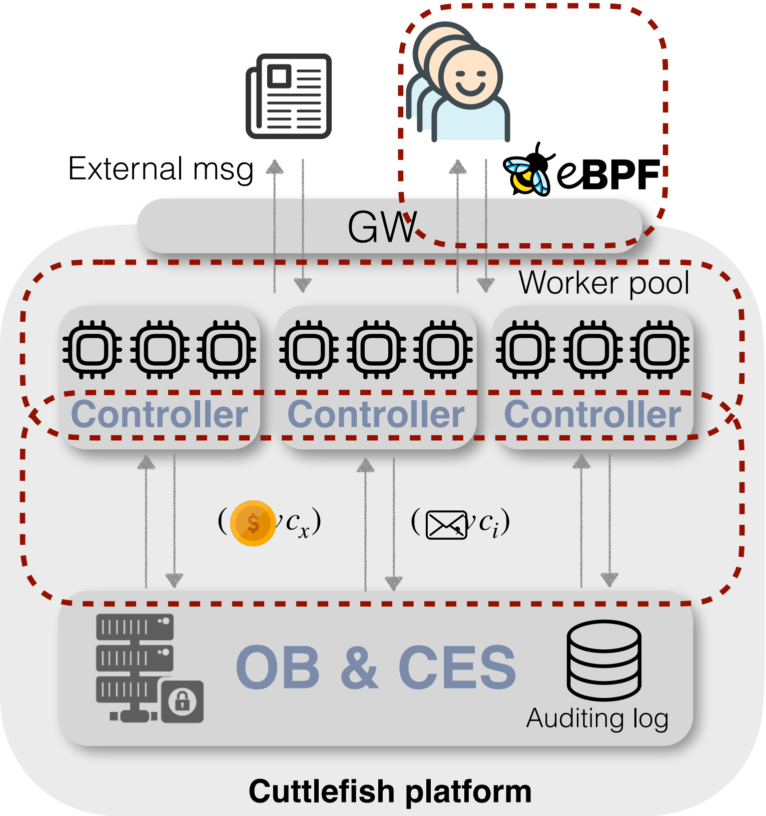
# Outline

- Conceptual foundation
- **Implementing virtual time overlay**
- Evaluation

# Implementing virtual time abstraction

**Instantiate** $vt$ as virtual cycles of a platform-agnostic IR/VM

**Account** and **control** the advancement of virtual cycles




1 **Programming interface**

3 **Runtime execution**

2 **Virtual cycle tracking**

# User programming abstraction

Input market data, external message…

**Online trading algorithm**

$algm^* = \text{argmax}_{algm} \text{profit}(algm)$

Trading decision(s)

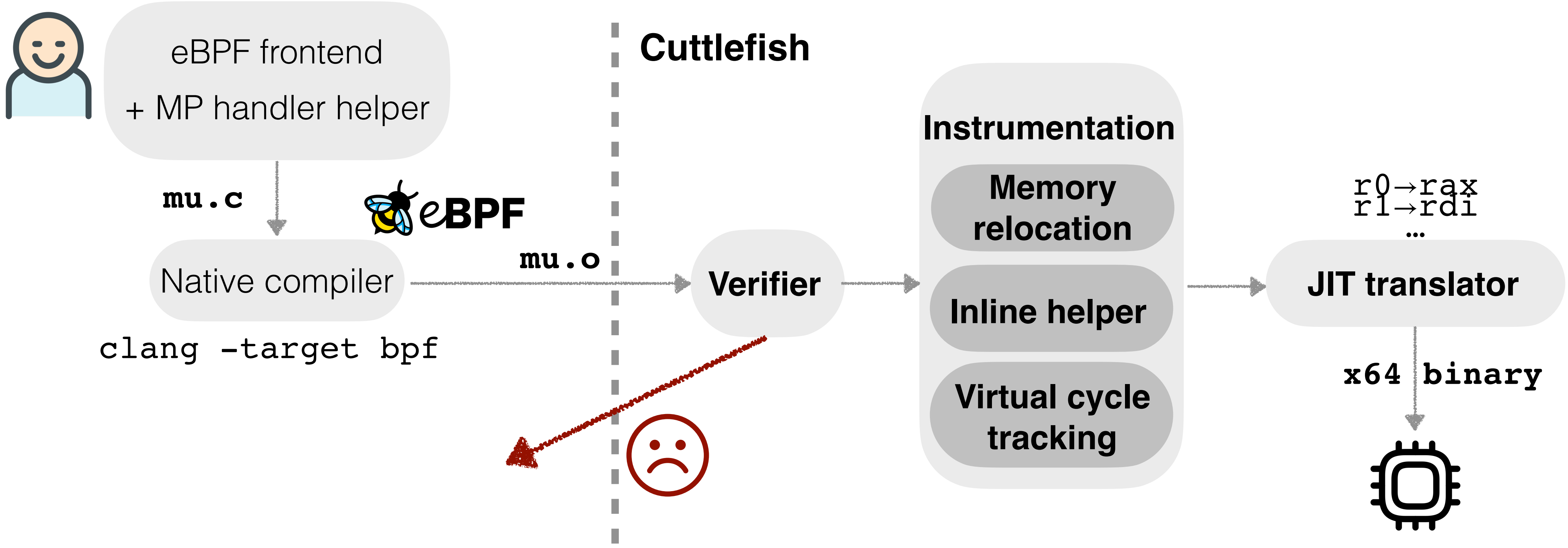
```
#include <cuttlefish_user.h>

int mp_handler(subscribed_context_t* data) {
    if ((*data) > 100) {
        // Sell
        trade_t trade = 1;
        submit_trade(&trade);
    } else if ((*data) < 10) {
        // Buy
        trade_t trade = 2;
        submit_trade(&trade);
    }
    map_update(0, &trade);
    return 0;
}
```

White-list set of extensible service APIs

Just-in-time trade submission

Narrow KV store API (e.g., lookup, update) for stateful invocations

# MP code lifetime




**2-tier compilation with the platform agnostic IR:**
***Track*** virtual cycle (fairly) in eBPF, but ***execute*** (efficiently) on native HW target

# How to track and advance virtual time cycles?

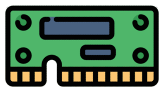
**eBPF asm**

```
0000000000000000 <u_handler>:
   0:    85 00 00 00 0b 00 00 00    call 11
   1:    7b 0a f8 ff 00 00 00 00    *(u64 *)(r10 - 8) = r0
   2:    bf a2 00 00 00 00 00 00    r2 = r10
   3:    07 02 00 00 f8 ff ff ff    r2 += -8
   4:    18 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00    r1 = 0 ll
   6:    85 00 00 00 0a 00 00 00    call 10
   7:    bf 01 00 00 00 00 00 00    r1 = r0
   8:    67 01 00 00 20 00 00 00    r1 <<= 32
   9:    77 01 00 00 20 00 00 00    r1 >>= 32
  10:    b7 00 00 00 01 00 00 00    r0 = 1
  11:    55 01 01 00 00 00 00 00    if r1 != 0 goto +1 <LBB0_2>
  12:    b7 00 00 00 00 00 00 00    r0 = 0

0000000000000068 <LBB0_2>:
  13:    95 00 00 00 00 00 00 00    exit
```

**Native HW asm**

```
; movabs r11, <vc address>
49 BB F0 DE BC 9A 78 56 34 12
; add qword ptr [r11], 2
49 81 03 02 00 00 00
                          x64
```

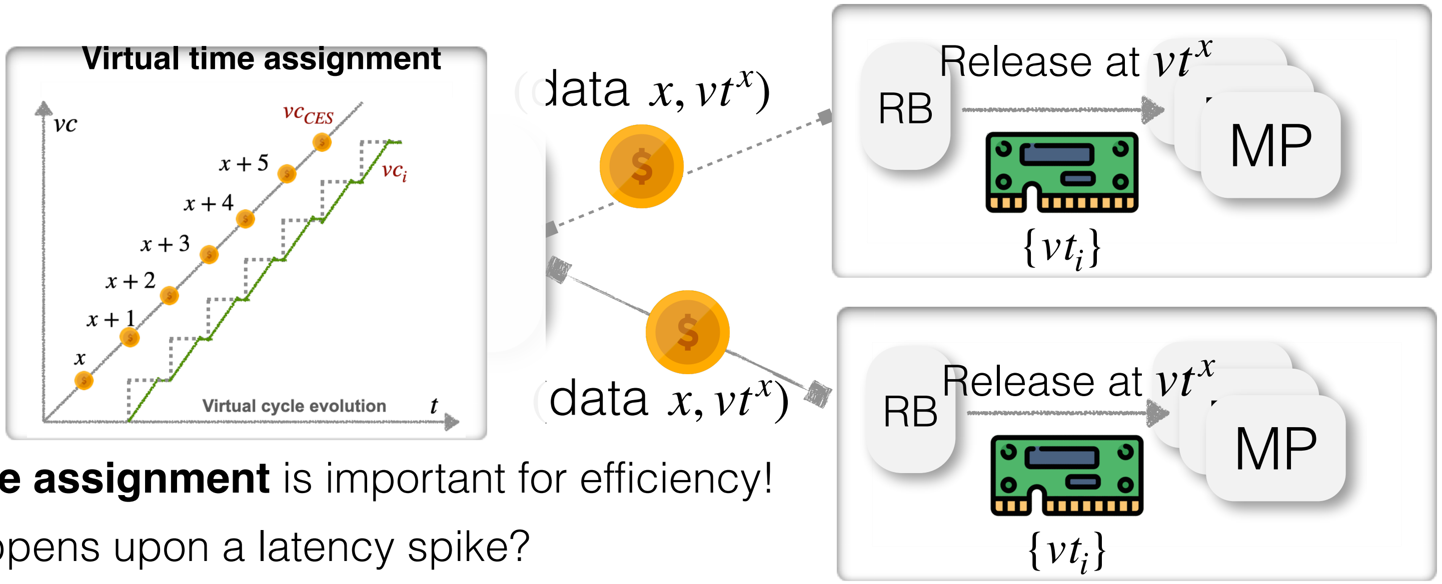
$vt_i += \Delta vt$



$\{vt_i\}$ maintained by execution runtime

- Break into basic blocks for batch updates of $vt_i$
  - JMP source, JMP destination, trade submission call
- Emit native machine code (2 x64 instr.) at the epilogue during JIT translation
  - Dummy trade/heartbeat for large blocks
  - Update the offsets for the (direct) JMP instructions

# Simultaneous data delivery in virtual time



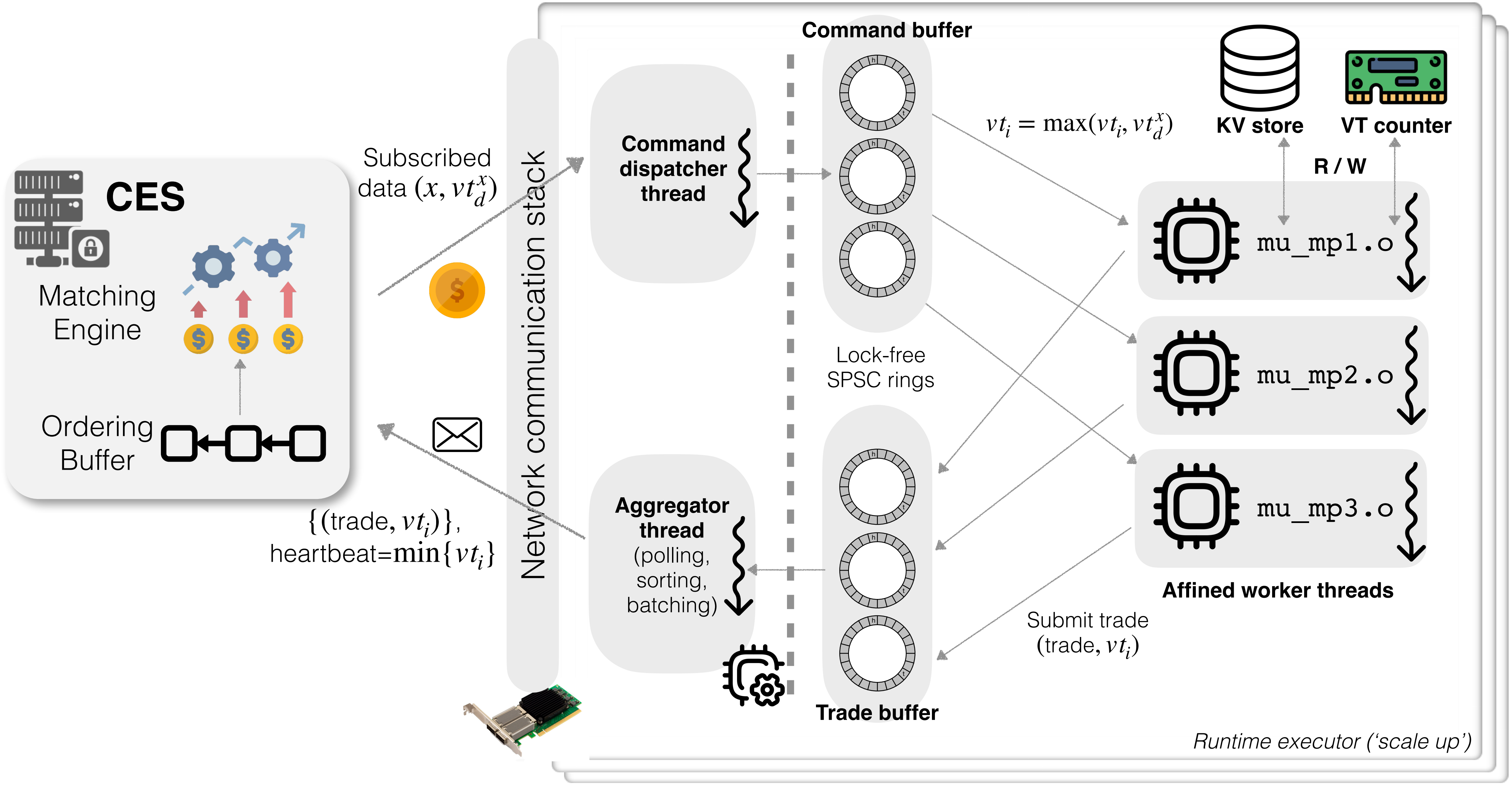

**Virtual time assignment** is important for efficiency!

- What happens upon a latency spike?
- What if some processor executions get slower?

*More details:*

- *Virtual time assignment algorithm*
- *Fault tolerance*
- *Handling external messages*
- *...*

# Runtime execution workflow



# Outline

- Conceptual foundation
- Implementing virtual time overlay
- **Evaluation**

# Comparison with existing ordering schemes

**Ordering mechanisms**

- Response Time (RT) based ordering
- FIFO ordering

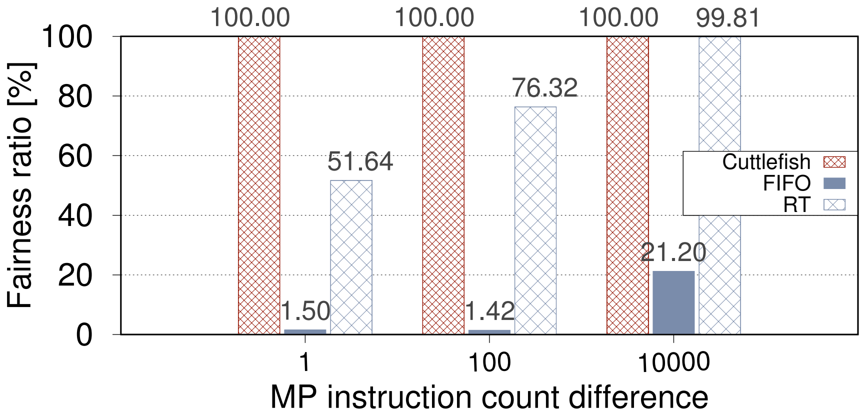
**Set up**

- Two MPs on two VMs
- $MP_a$ executes $N$ additional primitive IR instructions than $MP_b$
- Market data rate: every $\approx 100\mu s$

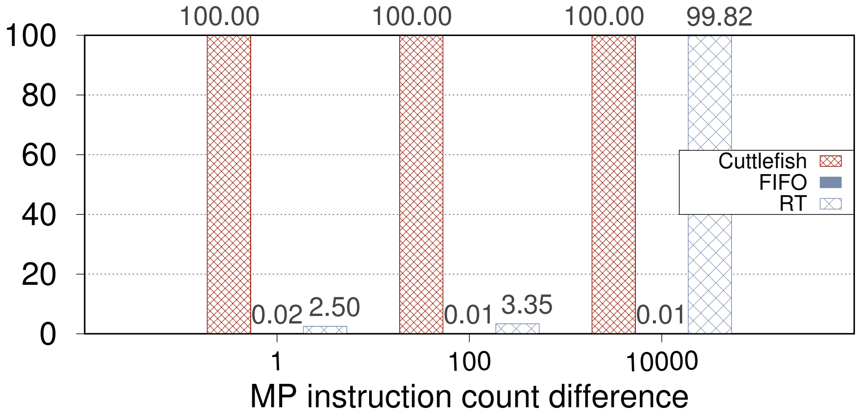
**Metric**

- Fairness ratio

**Same VM type**




**Different VM types**



# Performance cost for fairness

**Set up**

- 100 MPs on 10 VMs
- Market data rate: every $\approx 100\mu s$
- CX-4 NIC and Intel Xeon Platinum 8272CL CPU @ 2.60GHz

| | Latency (µs) | | | | |
|---|---|---|---|---|---|
| | **avg.** | **p50** | **p90** | **p99** | **p99.9** |
| MaxRTT | 52.04 | 47.74 | 49.95 | 55.85 | 144.2 |
| Cuttlefish | 54.19 | 50.82 | 53.49 | 68.46 | 166.3 |
| | +2.15 | +3.08 | +3.54 | +12.61 | +22.1 |

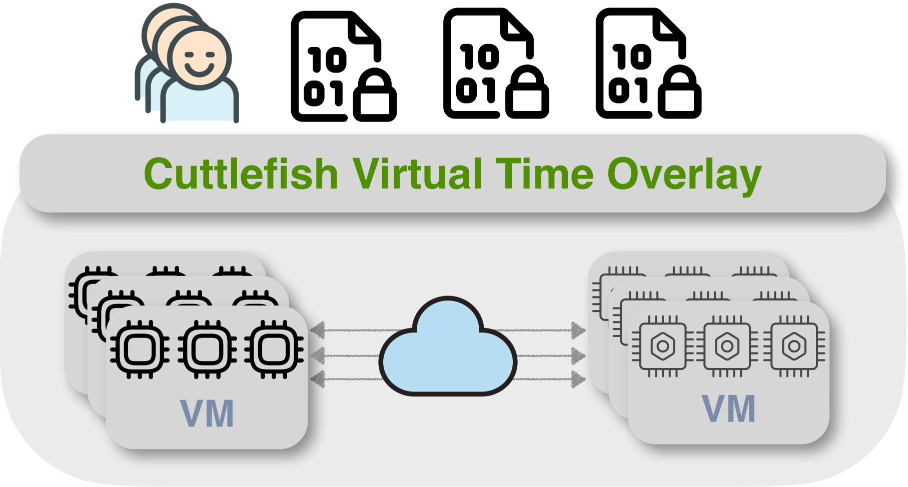
*More details:*

- *Execution throughput and latencies under processor disparities*
- *Virtual time instrumentation overhead*
- *Recovery under failures*

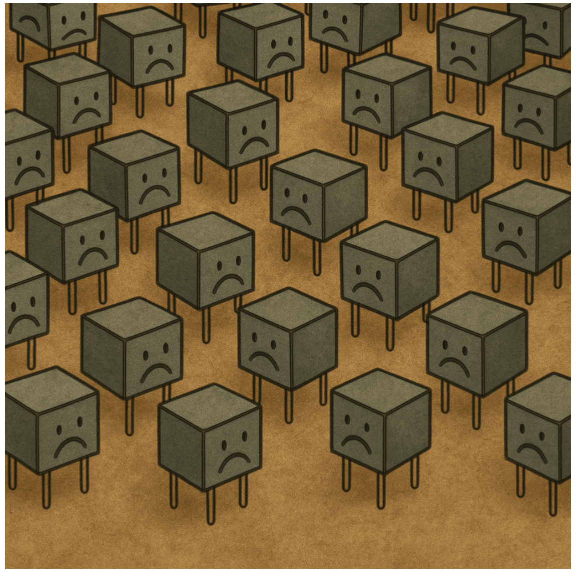
# Summary

Cuttlefish: a fair, predictable cloud-hosted exchange platform

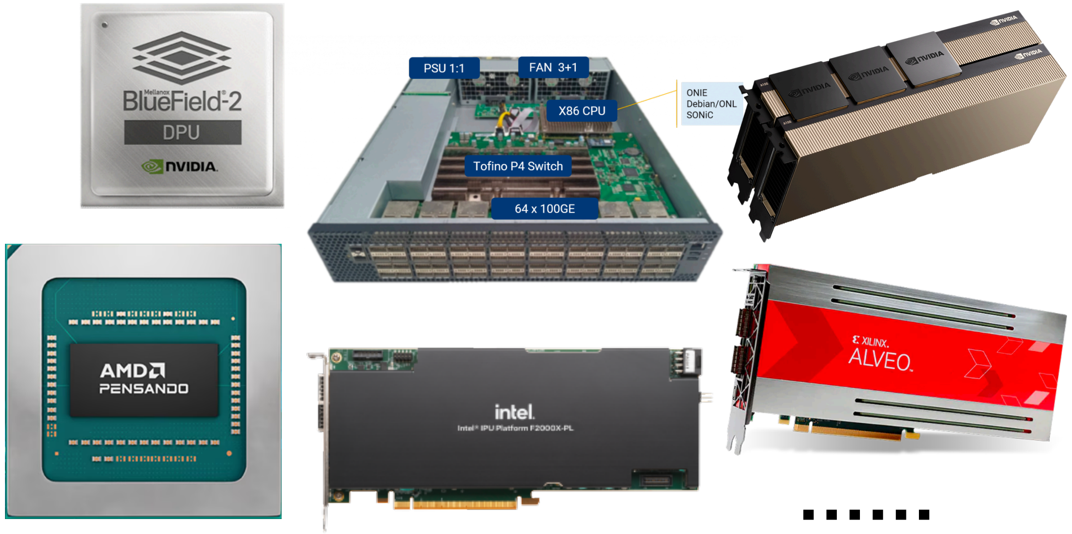
- Abstracting out variances in cloud communication and execution hardware
- An efficient implementation runnable on commercial cloud



# …something I am excited about

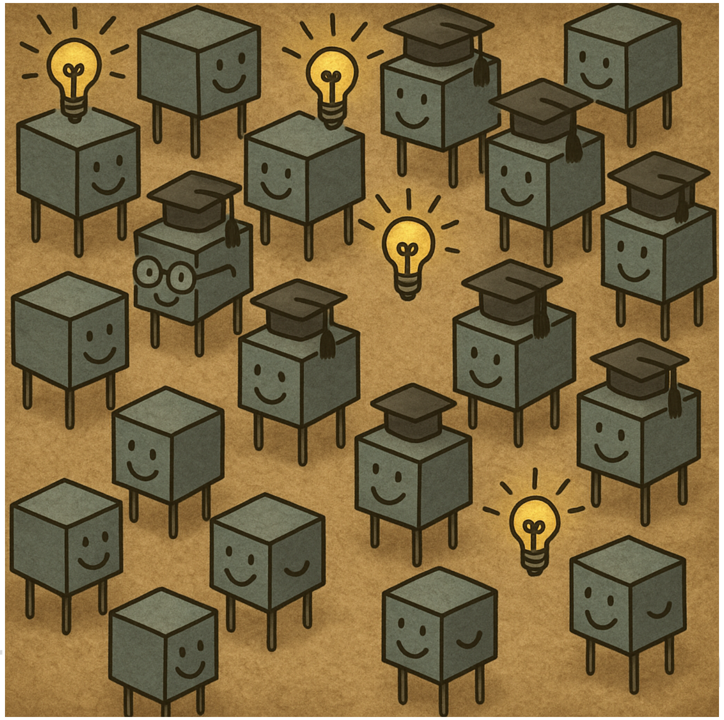


Transistor scaling is hitting walls

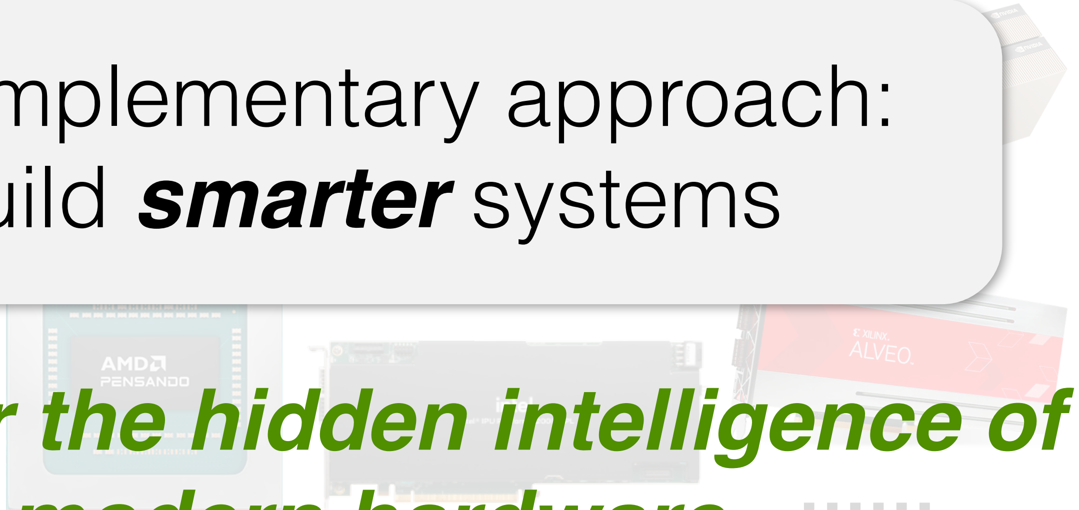



Rise of domain-specific accelerators

# ···something I am excited about



A complementary approach:
build ***smarter*** systems




***Uncover the hidden intelligence of modern hardware***

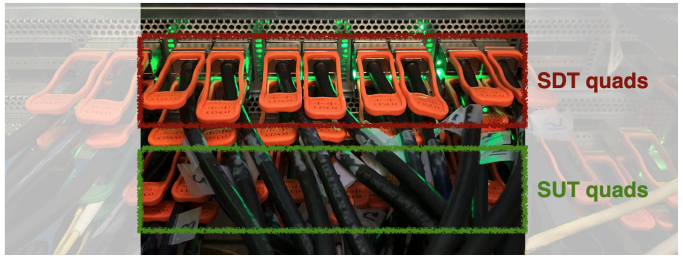
······

***…today!***

Tr… g walls

Rise of dom… ecific accelerators

# Case studies



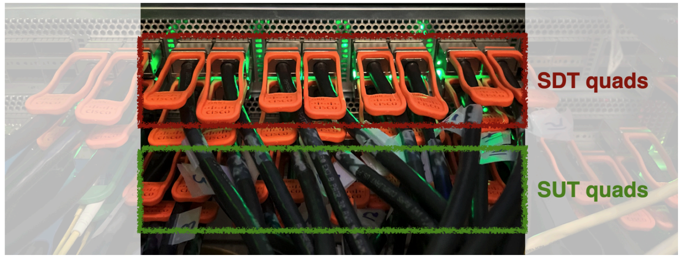

Harvesting IDLE cycles in programmable networks for in-band control functions

*OrbWeaver (NSDI ’22)*


Uncovering hidden potential of memory controllers in modern cloud servers

*Under preparation*

# Case studies




Harvesting IDLE cycles in programmable networks for in-band control functions

*OrbWeaver (NSDI '22)*

***Liangcheng (LC) Yu***, John Sonchack, and Vincent Liu



Uncovering hidden potential of memory controllers in modern cloud servers

*Under preparation*

# Networks are woven from packets

- A primary goal of computer networks: ***delivery packets***

# Networks are woven from packets

- A primary goal of computer networks: ***delivery packets***
    - ***User application***: video streaming, web browsing, file transfer…

# Networks are woven from packets

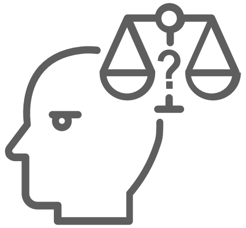
- A primary goal of computer networks: ***delivery packets***
    - ***User application***: video streaming, web browsing, file transfer…
    - ***Non-user application***: control messages, probes about network state, keep alive heartbeats…

# Networks are woven from packets

- A primary goal of computer networks: ***delivery packets***
  - ***User application***: video streaming, web browsing, file transfer…
  - ***Non-user application***: control messages, probes about network state, keep alive heartbeats…

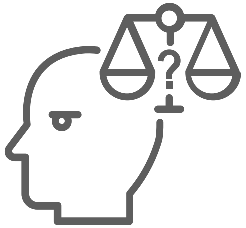
- Often, two classes of traffic ***multiplex*** the same network

# When introducing an in-band control function…



To cost ***extra bandwidth*** for ***efficacy***, or not?

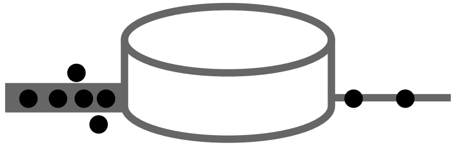
# When introducing an in-band control function···



To cost ***extra bandwidth*** for ***efficacy***, or not?

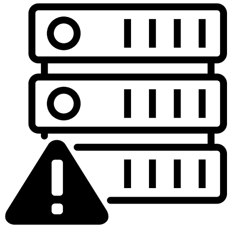
Time synchronization

**Clock-sync rate** ↔
**clock precision**



Congestion notification

**Probe data/rate** ↔
**measurement accuracy**



Failure detection

**Heartbeat frequency** ↔
**detection speed**

In-band telemetry

**INT postcard volume** ↔
**post-mortem analysis**

# When introducing an in-band control function…

To cost ***extra bandwidth*** for ***efficacy***, or not?

Can we coordinate at ***high-fidelity*** with a ***near-zero cost*** (to usable bandwidth, latency…)?

**clock precision**

**measurement accuracy**

Failure detection

**Heartbeat frequency** ↔ **detection speed**

In-band telemetry

**INT postcard volume** ↔ **post-mortem analysis**

# When introducing an in-band control function…

To cost ***extra bandwidth*** for ***efficacy***, or not?

clock precision

measurement accuracy

Can we coordinate at ***high-fidelity*** with a ***near-zero cost*** (to usable bandwidth, latency…)?

***Idea: Weaved Stream***

- Exploit ***every gap*** ($O(100$ns$)$) between user packets opportunistically
- Inject customizable ***IDLE packets*** carrying information across devices

Failure detection

In-band telemetry

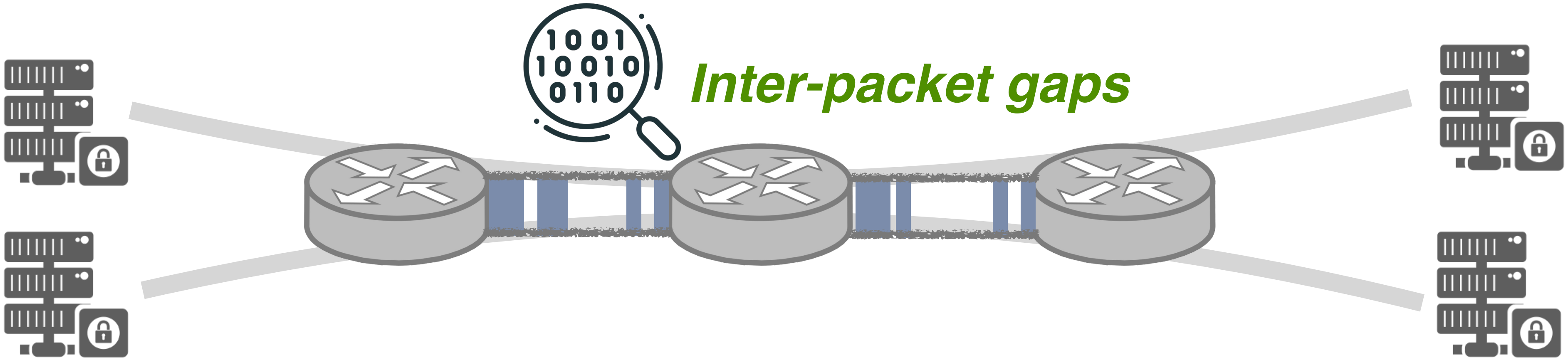
Heartbeat frequency

INT postcard volume

detection speed

post-mortem analysis

# Opportunity: $< \mu s$ gaps are prevalent

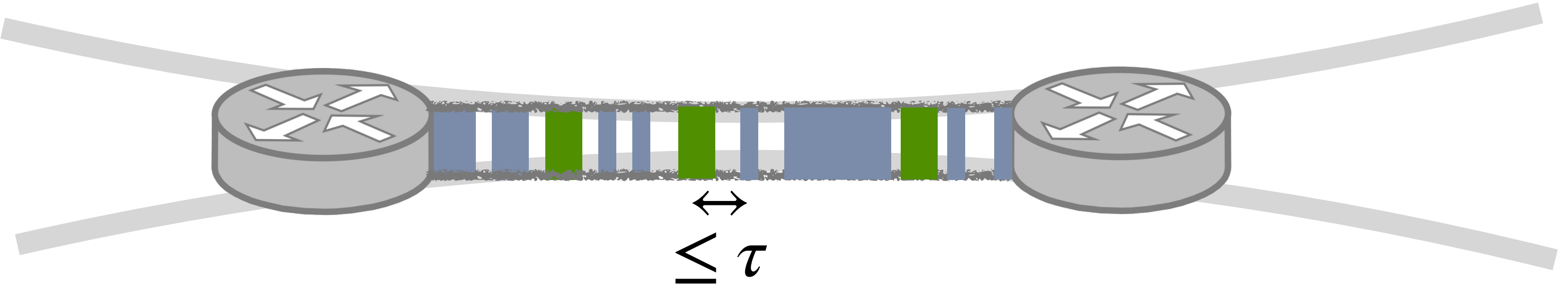



*Root causes?*

- Uncertainties in application load patterns (e.g., burstiness)
- Conservative resource provisioning for peak usages
- Bottlenecks at CPU processing vs network BW
- TCP effects
- Structural asymmetry
- …

# Abstraction: weaved stream

*Union of **user** AND **IDLE** (injected) packets*




***[R1 Predictability]*** Interval between ***any*** two consecutive packets $\leq \tau$

$$\tau = B_{100Gbps}/MTU_{1500B} = 120ns$$

***[R2 Little-to-zero overhead]*** Near-zero impact to user packets or power draw

# Abstraction: weaved stream

*Union of **user** and **IDLE** (injected) packets*

Implement many ***in-network functions***

*(failure detection, clock sync, congestion notification…)*

***for free!***

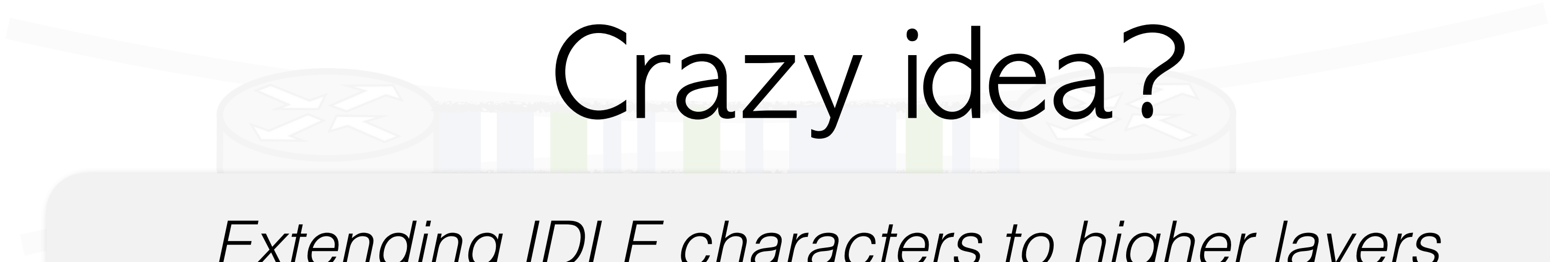
***[R1***

$$\tau = B_{100Gbps}/MTU_{1500B} = 120ns$$

***[R2 Little-to-zero overhead]*** Not impact user packets or power draw

# Abstraction: weaved stream

*Union of **user** and **IDLE** (injected) packets*



# Crazy idea?

*Extending IDLE characters to higher layers*

- Data plane packet generator
- Replication engine
- Data plane programmability
- Flexible switch configuration (priorities, buffers…)

***[R1 Pre***

***[R2 Little-to-zero overhead]*** Not impact user packets or power draw

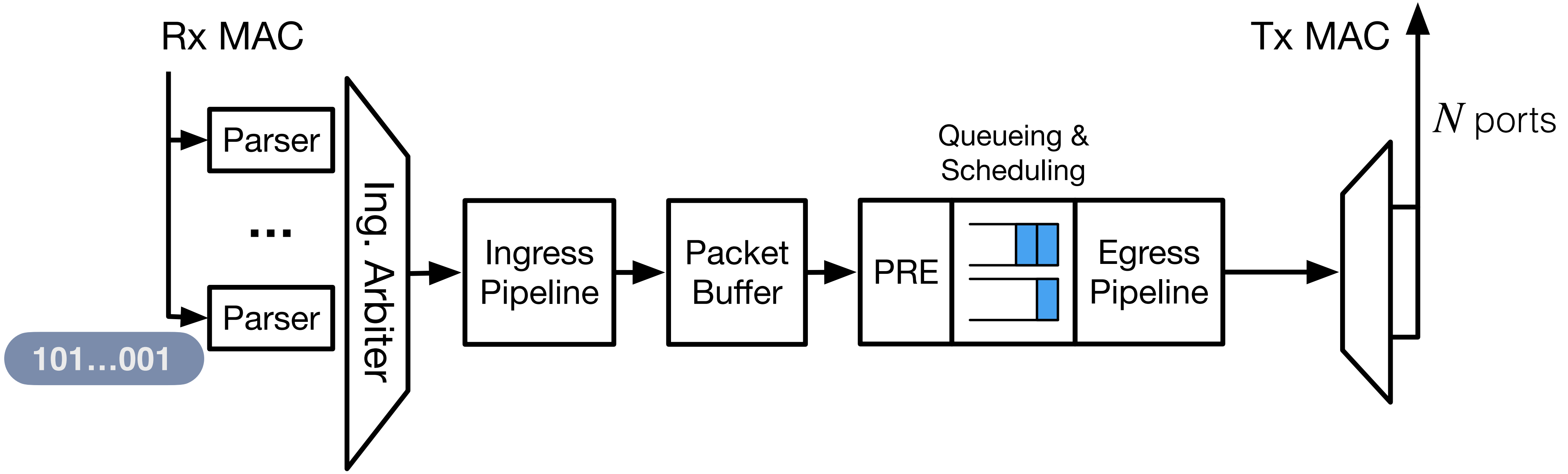
# OrbWeaver: outline

1. RMT switch data plane architecture

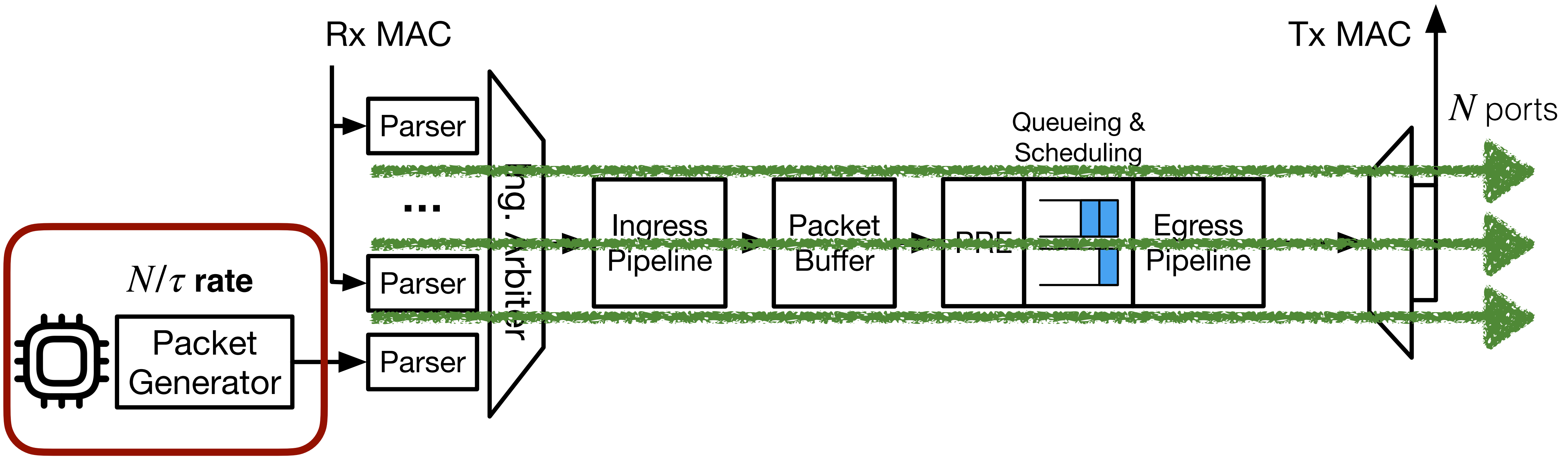
2. Implementing weaved stream abstraction

3. OrbWeaver applications

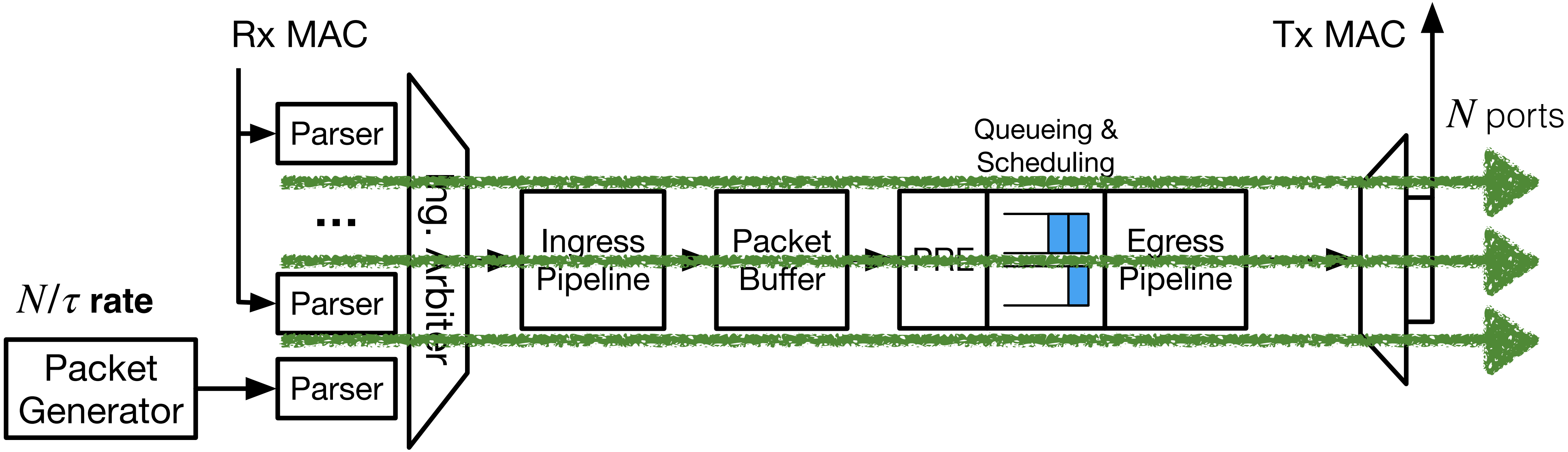
# RMT switch architecture



# Strawman: blind packet generation

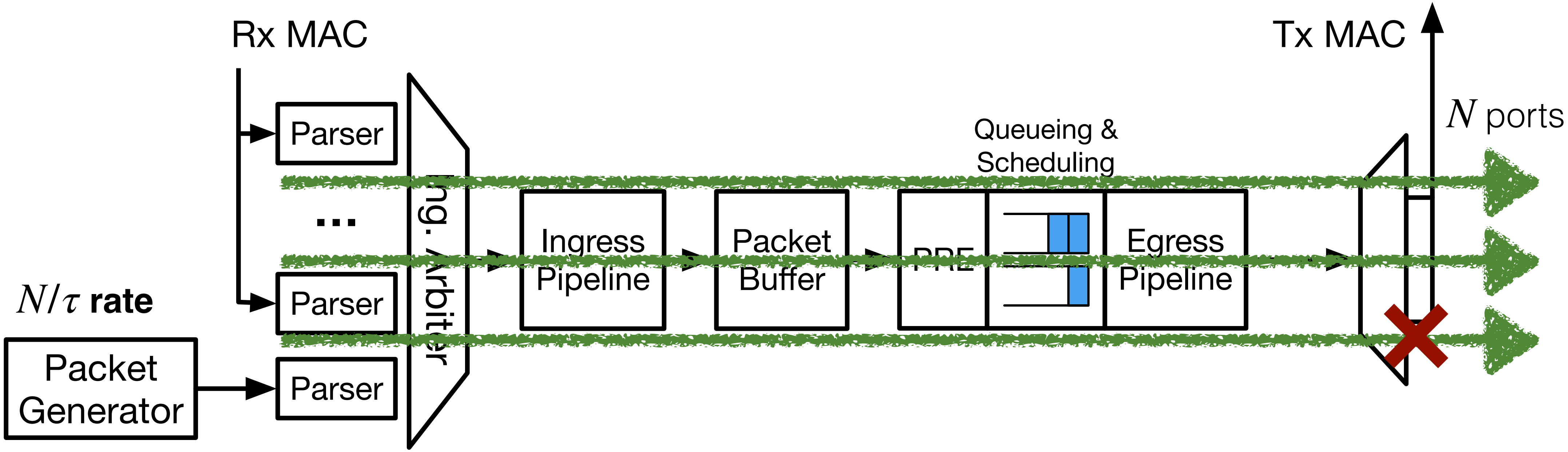


# Strawman: blind packet generation



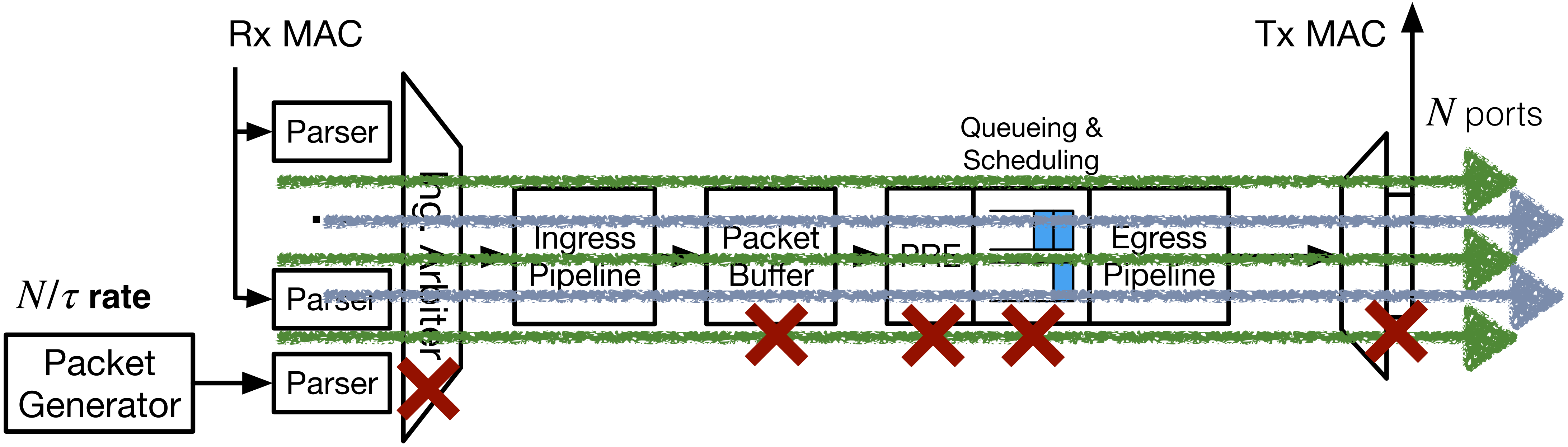

Predictability even there is no user traffic ✅

# Problems with blind packet generation




**#1 Scalability**: overwhelm generator capacity to satisfy target rate for all ports

# Problems with blind packet generation

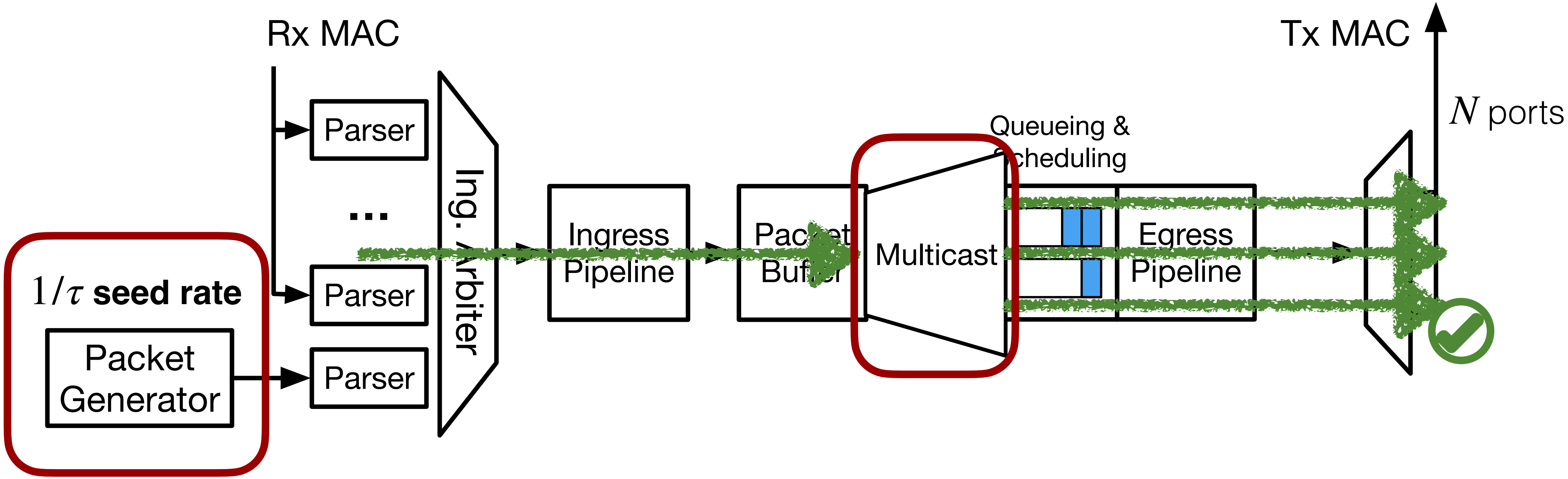



**#1 Scalability**: overwhelm generator capacity to satisfy target rate for all ports

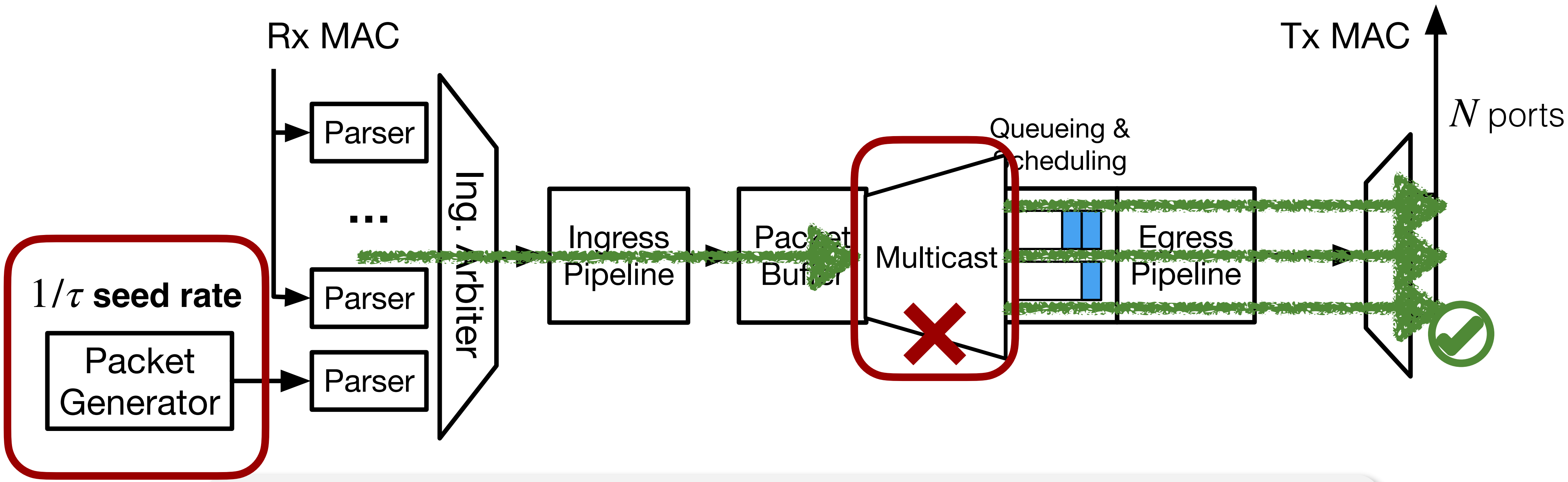
**#2 Cross-traffic contention**: affect throughput, latency, or loss of **user traffic**!

# Problem #1 : scalability

***Solution***: *seed stream amplification*



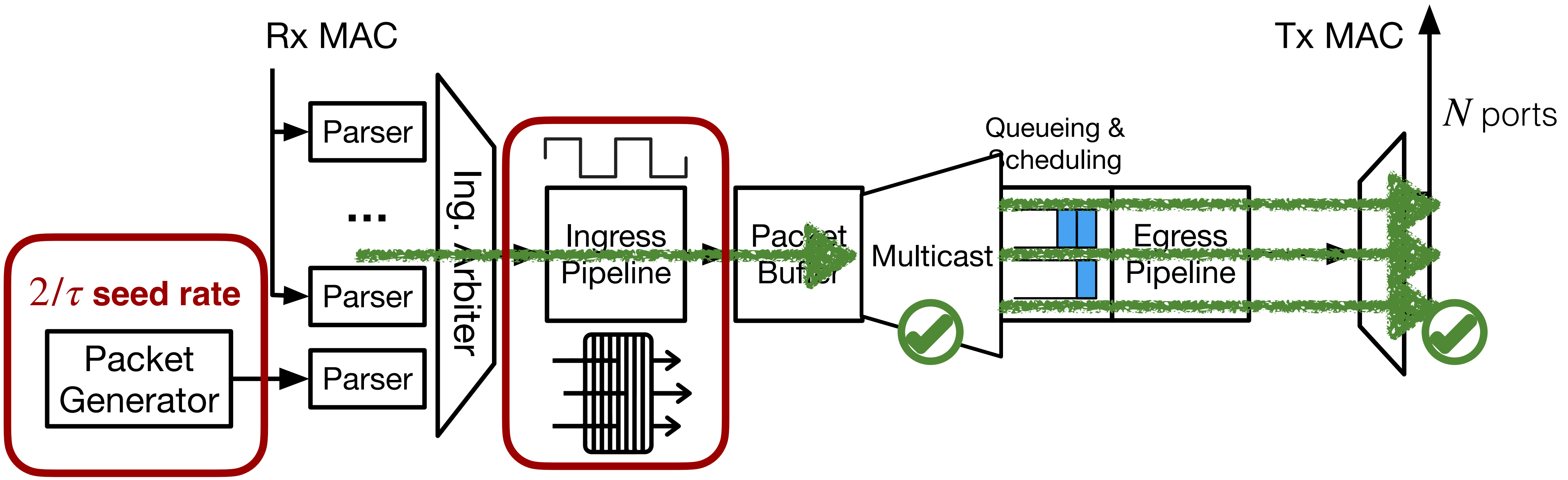
# Problem #2: cross-traffic contention at PRE




**Monopolize usage and waste PRE packet-level BW!**

# Problem #2: cross-traffic contention at PRE

***Solution**: amplify seed stream **on-demand***

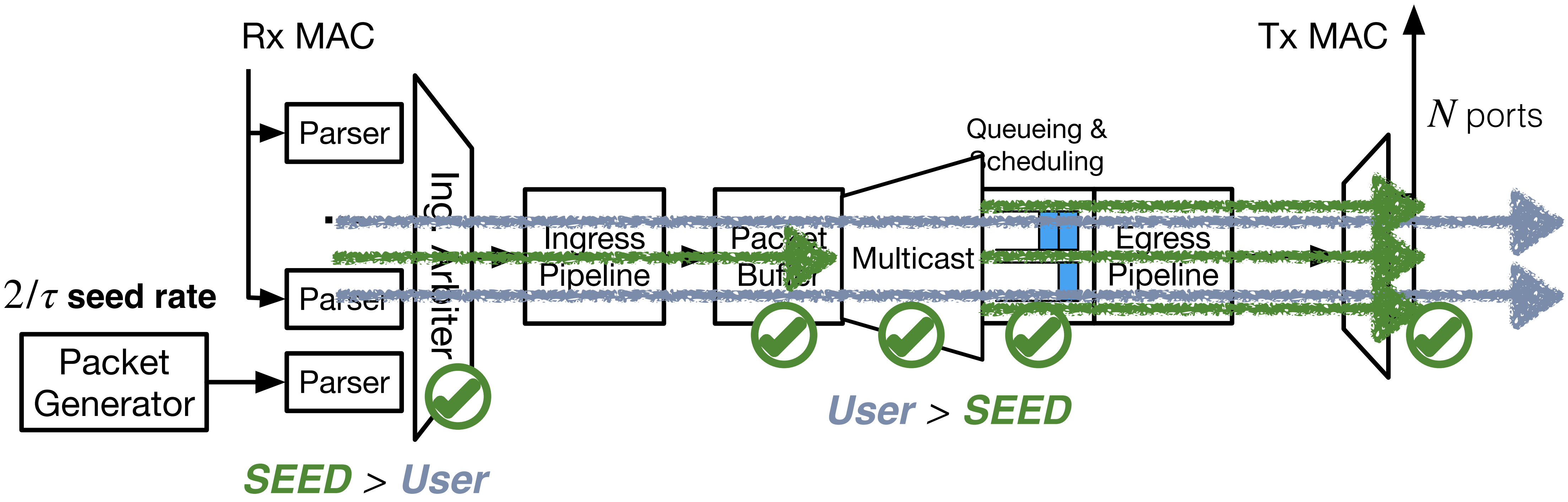



***Selective filtering***

- Per-egress port bitmap indicating packet presence in the last $\tau/2$ cycle
- If not, replicate an IDLE to the port

# Problem: other contention points

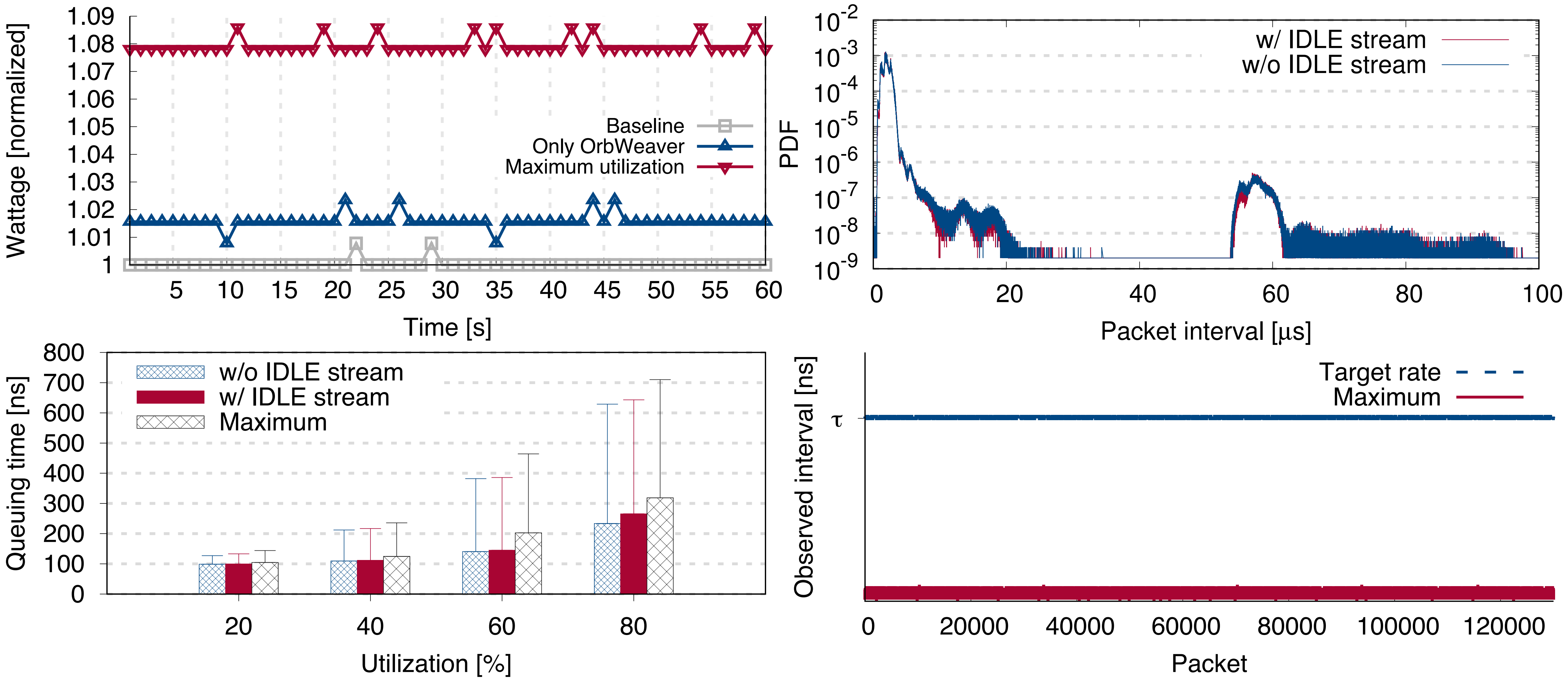
***Solution**: leverage rich configuration options for priorities and buffer management*



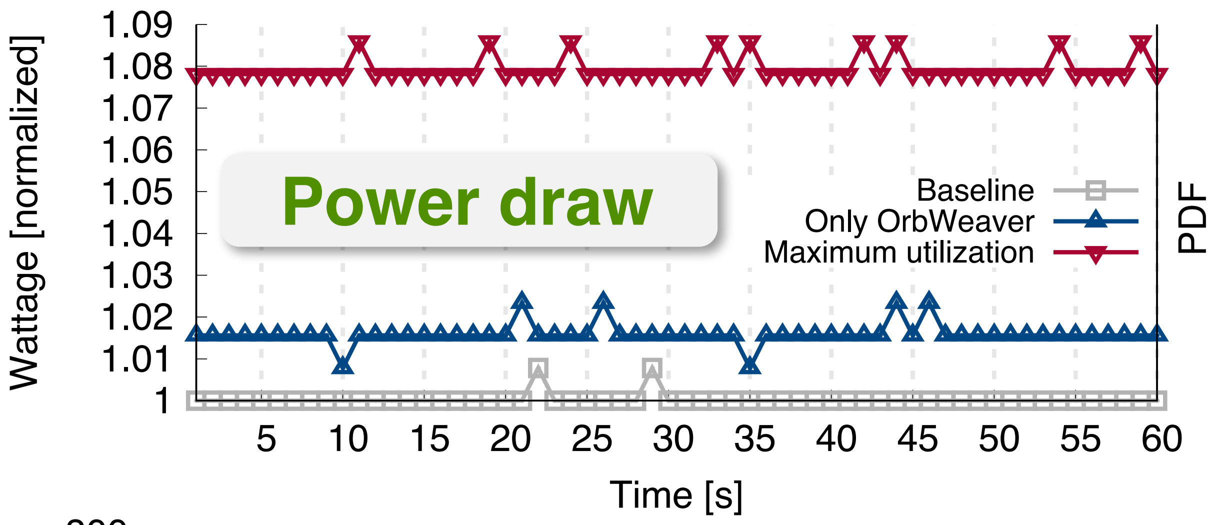

- *Zero impact of weaved stream predictability*
- *Zero impact of **user traffic** throughput or buffer usage*
- *Negligible impact of latency of **user packets***

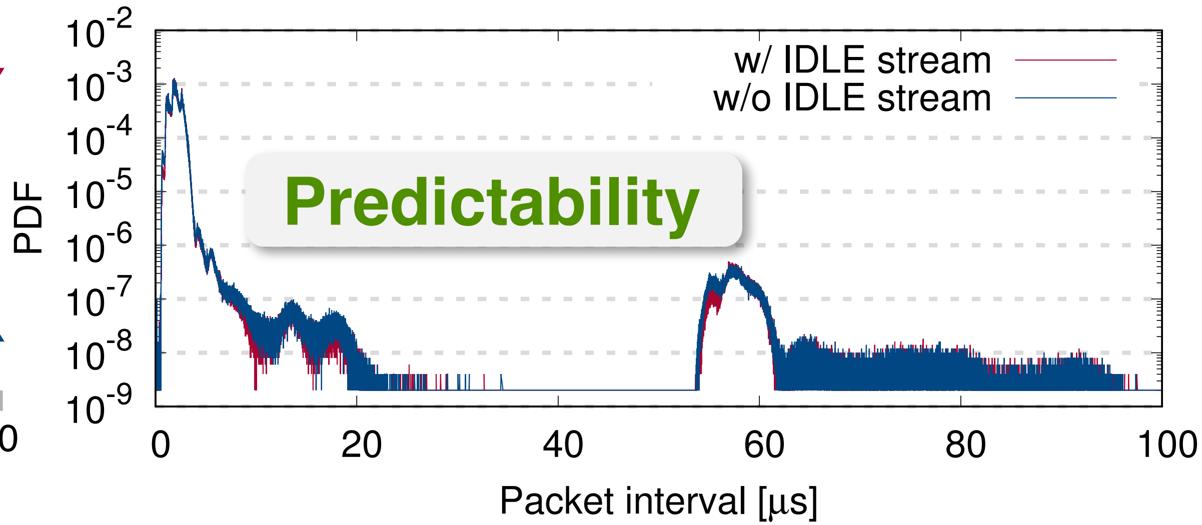
# Implementation and evaluation

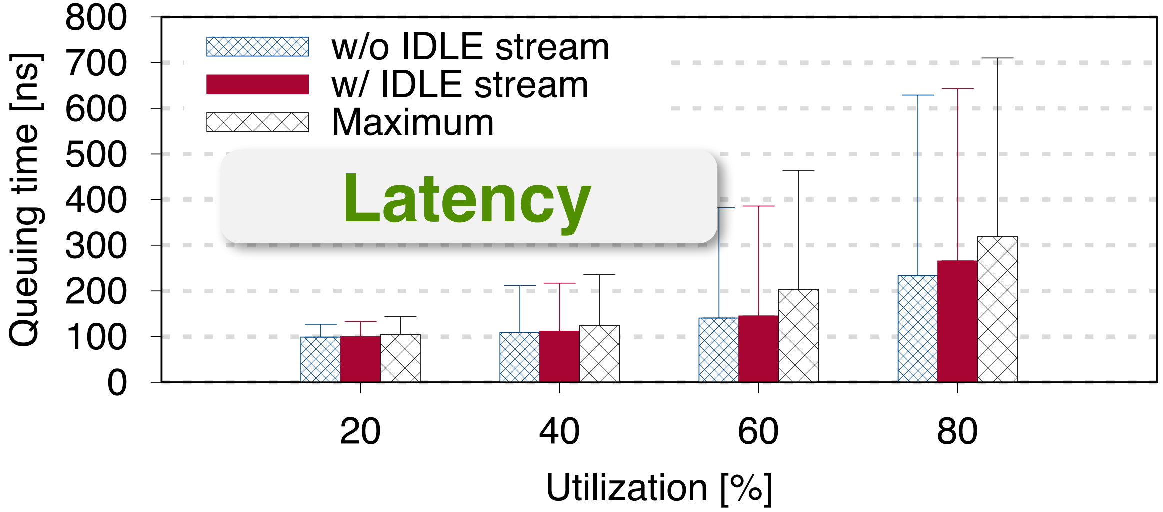
*Hardware prototype on a pair of Wedge100BF-32X Tofino switches*

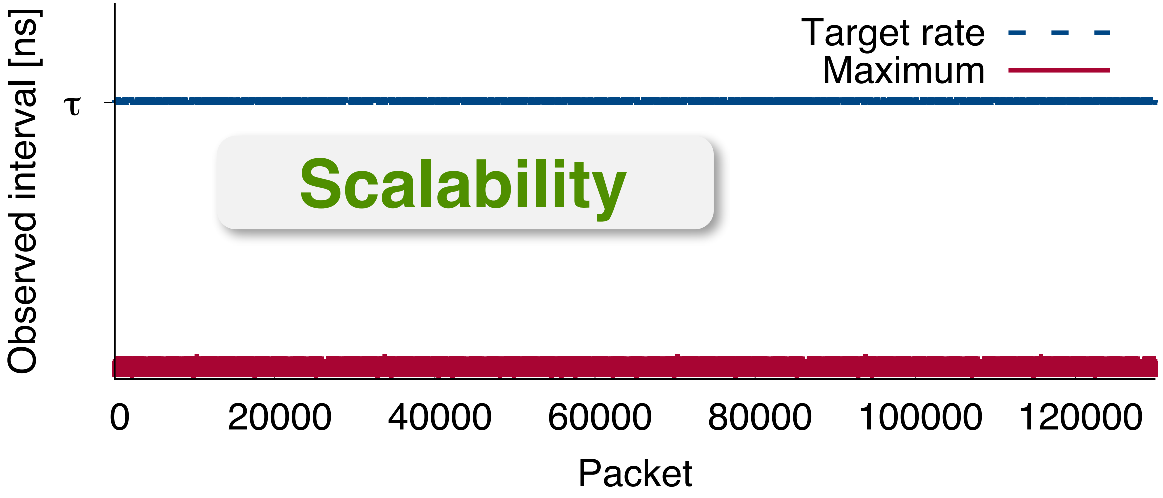


***Takeaway***: **Little-to-no impact** of power draw, latency, or throughput while guaranteeing **predictability** of the weaved stream!

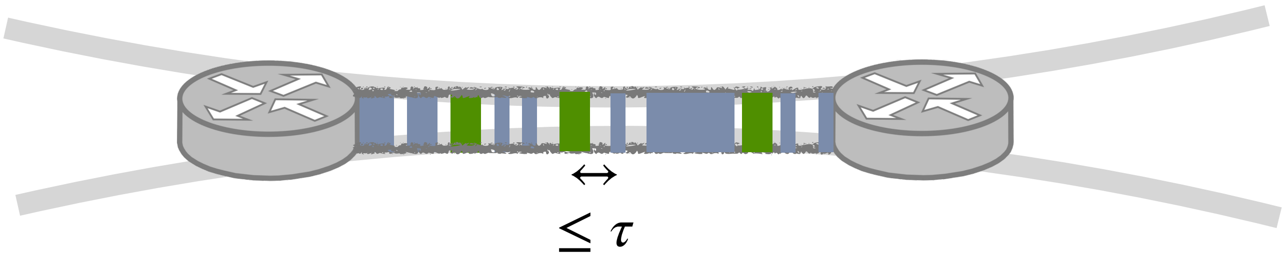


# OrbWeaver use cases



***[R1 Predictability]*** → Infer network state at fine-granularity!

***[R2 Little-to-zero overhead]*** → Inject information using IDLE cycles!

# OrbWeaver use cases

Self-healing failure detection

Flowlet load imbalance

Performance aware routing

Latency localization

In-band telemetry

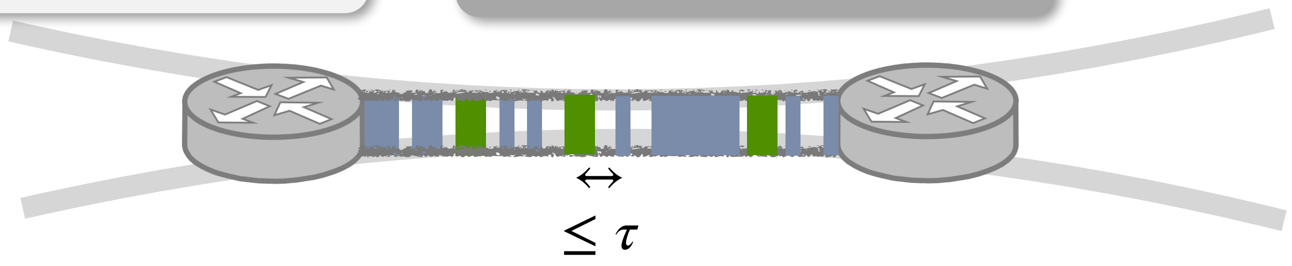
Header compression

Microburst detection

Consistent replicas

Packet forensics

Clock synchronization

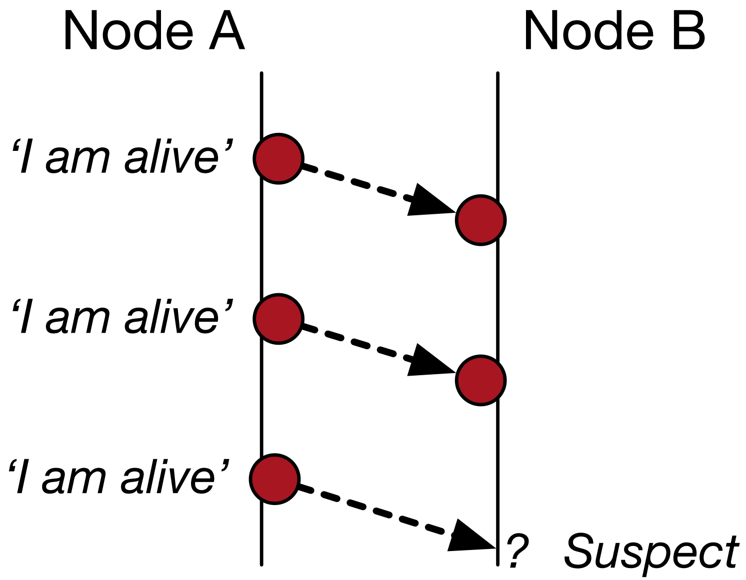



***[R1 Predictability]*** → Infer network state at fine-granularity!

***[R2 Little-to-zero overhead]*** → Inject information using IDLE cycles!

# Example: failure detection

Traditional uni-directional heartbeats




**Common approach:**

Periodic, high priority heartbeats

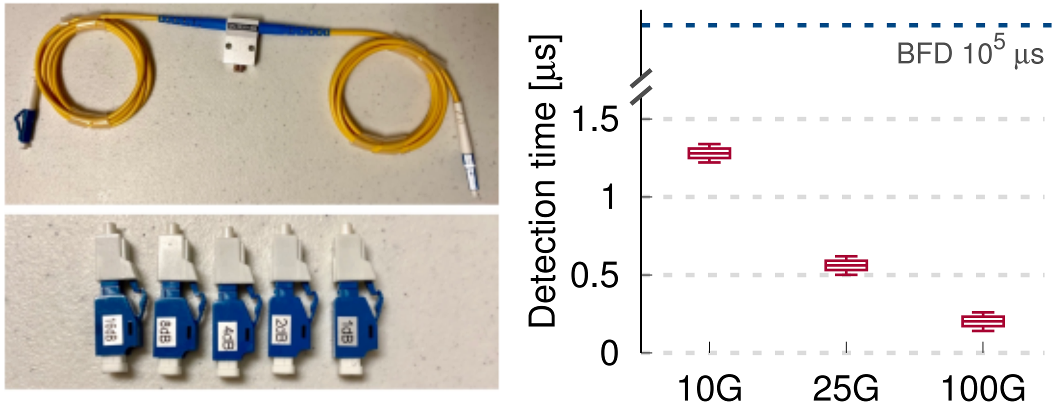

***Fundamentally indistinguishable****:*
*message drop or actual failure?*

Empirically, use conservative detection thresholds

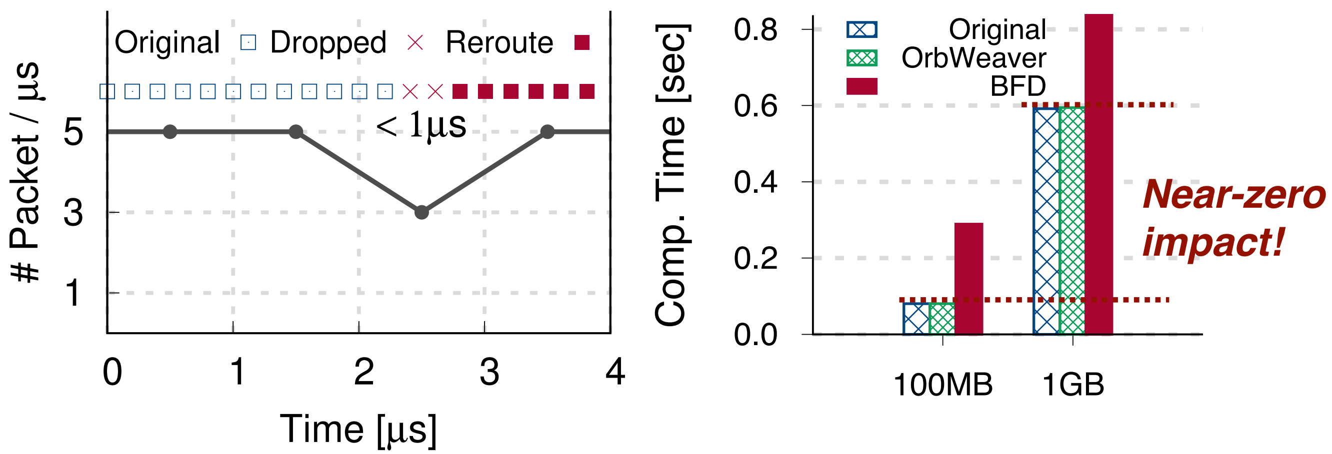
# Failure detection with OrbWeaver

*<u>Before:</u> Weak guarantee of the messaging channel*

*<u>After</u>: Guaranteed maximum inter-packet gap (120ns for 100 GbE)*



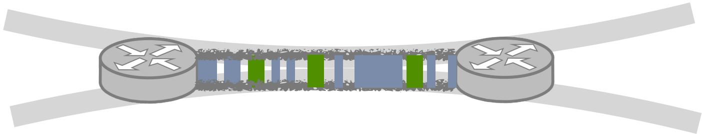

*Detection time of emulated failures using optical attenuators under varying link speeds*




*Instantaneous self-healing failure mitigation when combined with data-plane reroute*

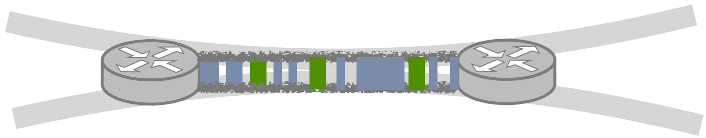
OrbWeaver pushes the detection speed to its ***limits***

# OrbWeaver: summary



- **Weaved stream abstraction** to harvest IDLE cycles
  - Sufficient for many in-band control functions
  - *Don't* need to choose between coordination fidelity and bandwidth overhead

# OrbWeaver: summary



- **Weaved stream abstraction** to harvest IDLE cycles
  - Sufficient for many in-band control functions
  - *Don't* need to choose between coordination fidelity and bandwidth overhead
- Implementable on today's RMT switches
  - Push the utilization of IDLE cycles to its limits
  - Guarantee predictability with little-to-zero overhead

# Case studies




Harvesting IDLE cycles in programmable networks for in-band control functions

*OrbWeaver (NSDI ’22)*

Uncovering hidden potential of memory controllers in modern cloud servers

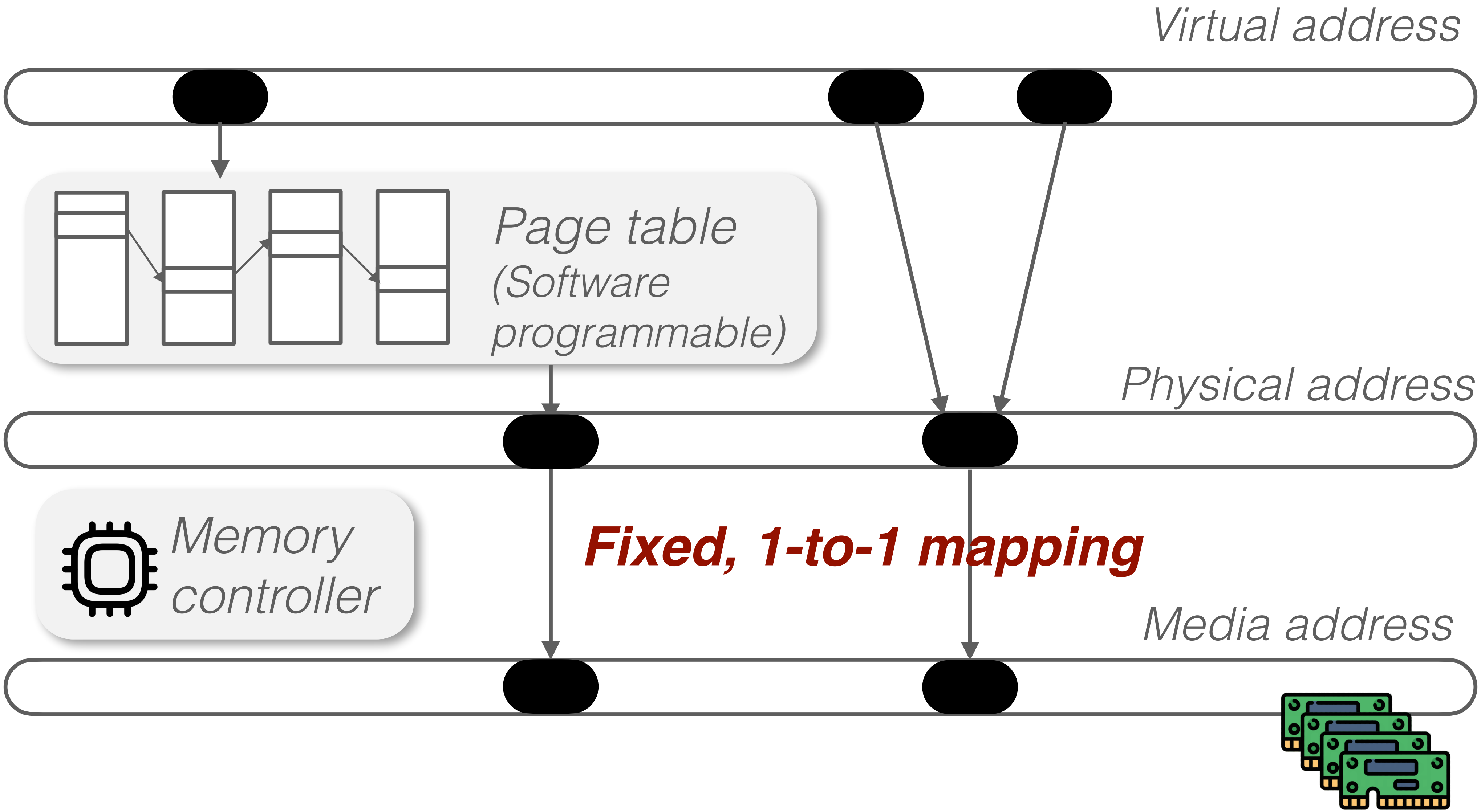
*Under preparation*

Daniël Trujillo, ***Liangcheng (LC) Yu***, Stefan Saroiu, and Alec Wolman

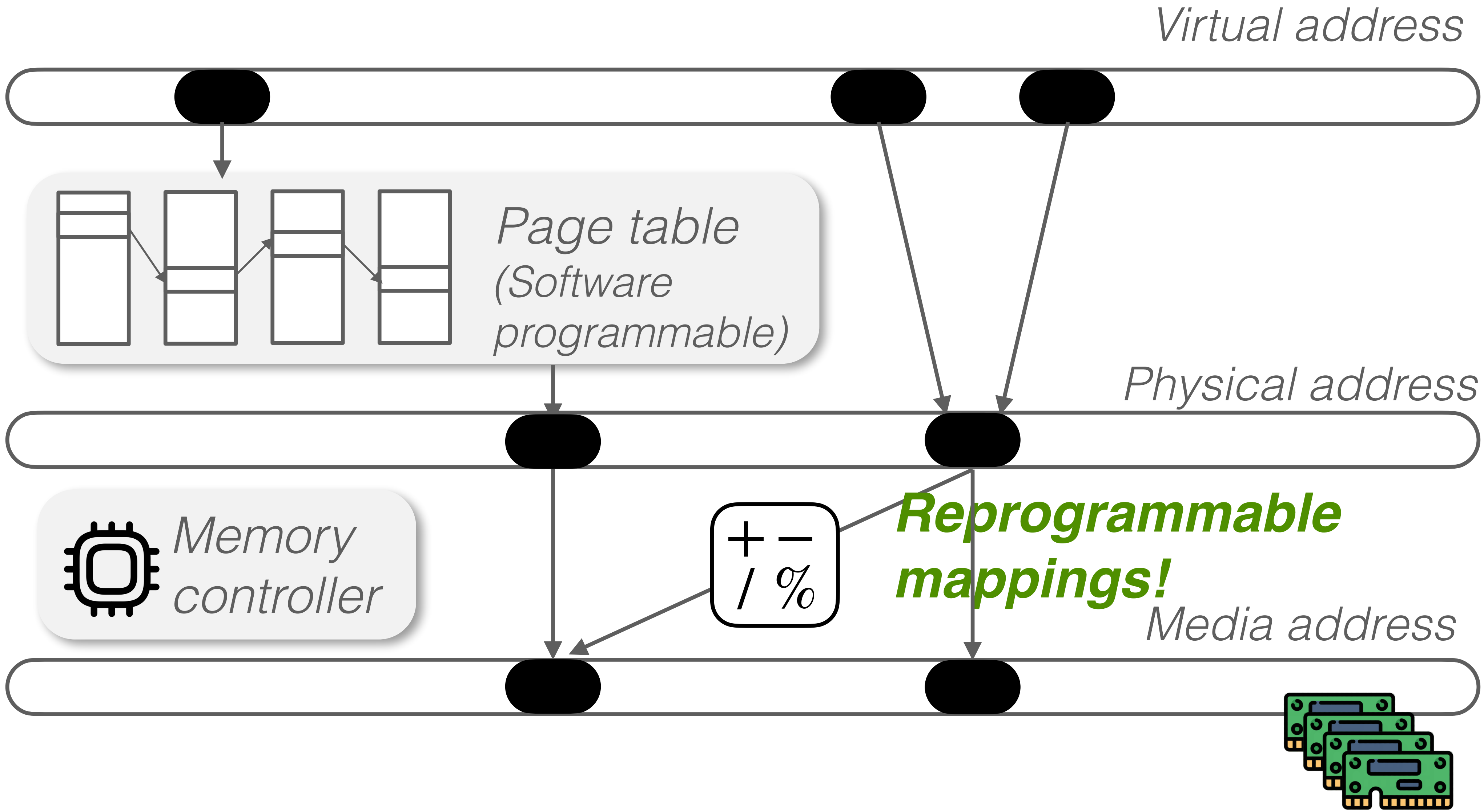
Massachusetts Institute of Technology

Microsoft Research

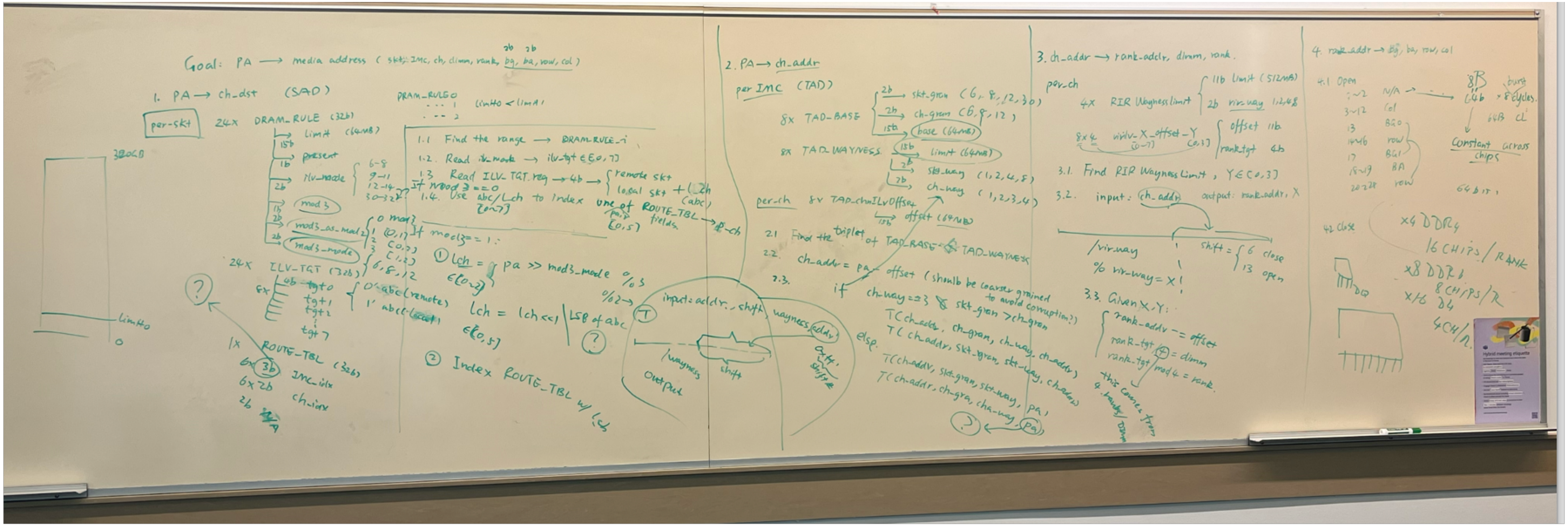
# A view of memory address translation



# Memory address translation tomorrow



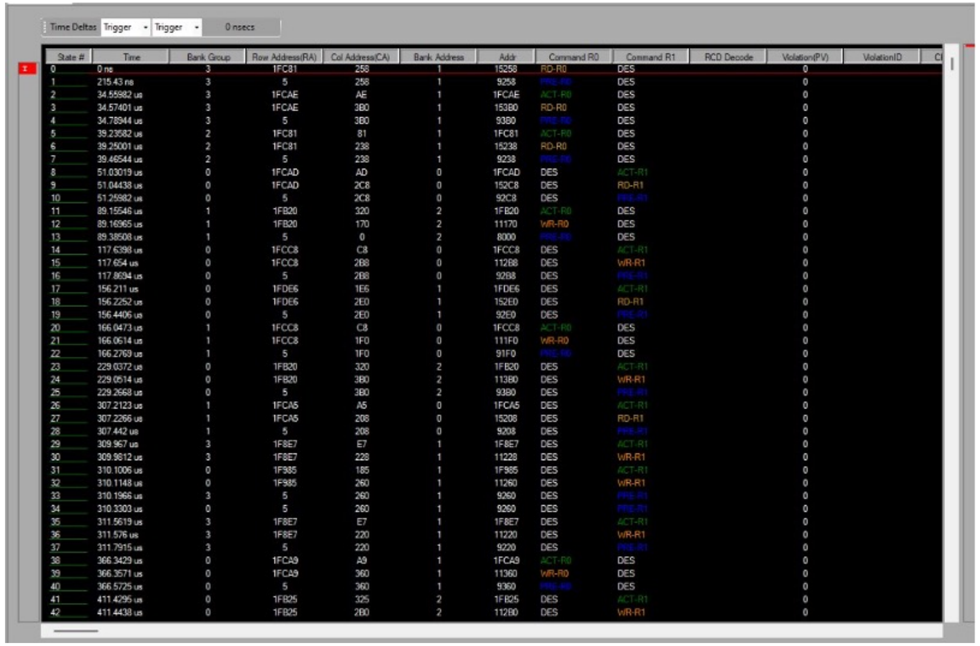
# Reprogramming DRAM maps is hard




One of the whiteboard tutorials I gave

# Our secret sauce

- Unlocked BIOS
- Reverse-engineered MC on modern cloud servers
- Bus monitor

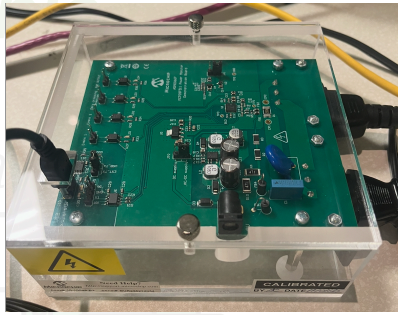
# Memory address translation tomorrow

*Implemented novel primitives improving cloud server efficiency and security!*

**Illustrative examples:**

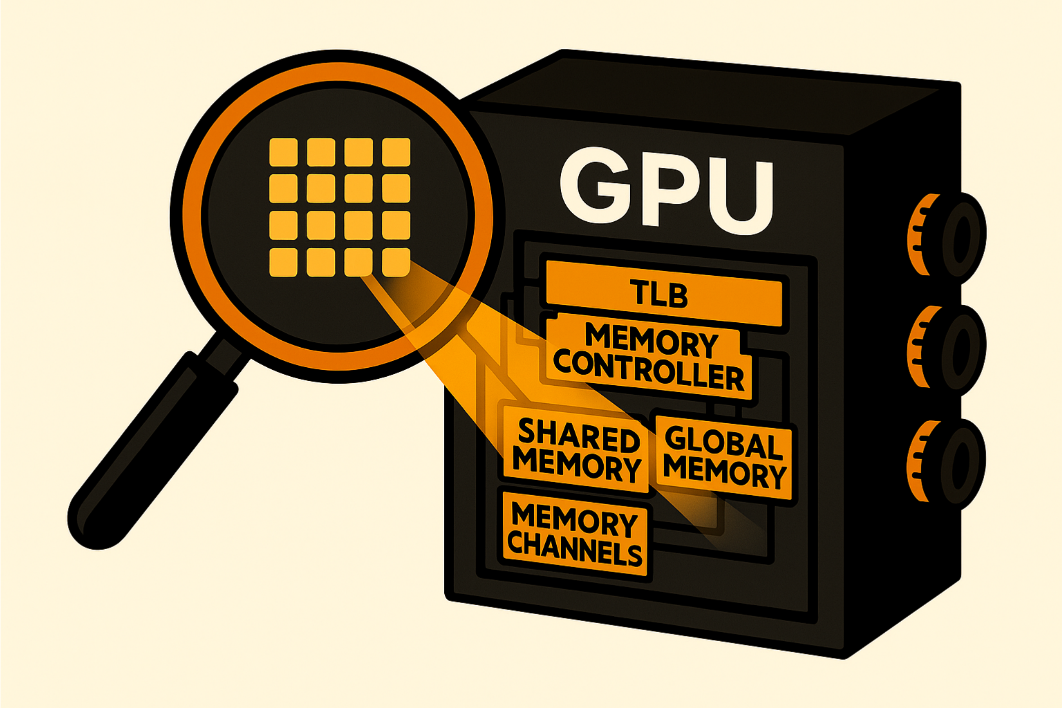
OS can choose among memory interleaving maps

Intelligent DRAM refreshes for power savings



...

# What's next?





https://jobs.careers.microsoft.com/global/en/job/1886648/Research-Intern---MSR-Software-Hardware-Co-design