OPENFABRICS ALLIANCE

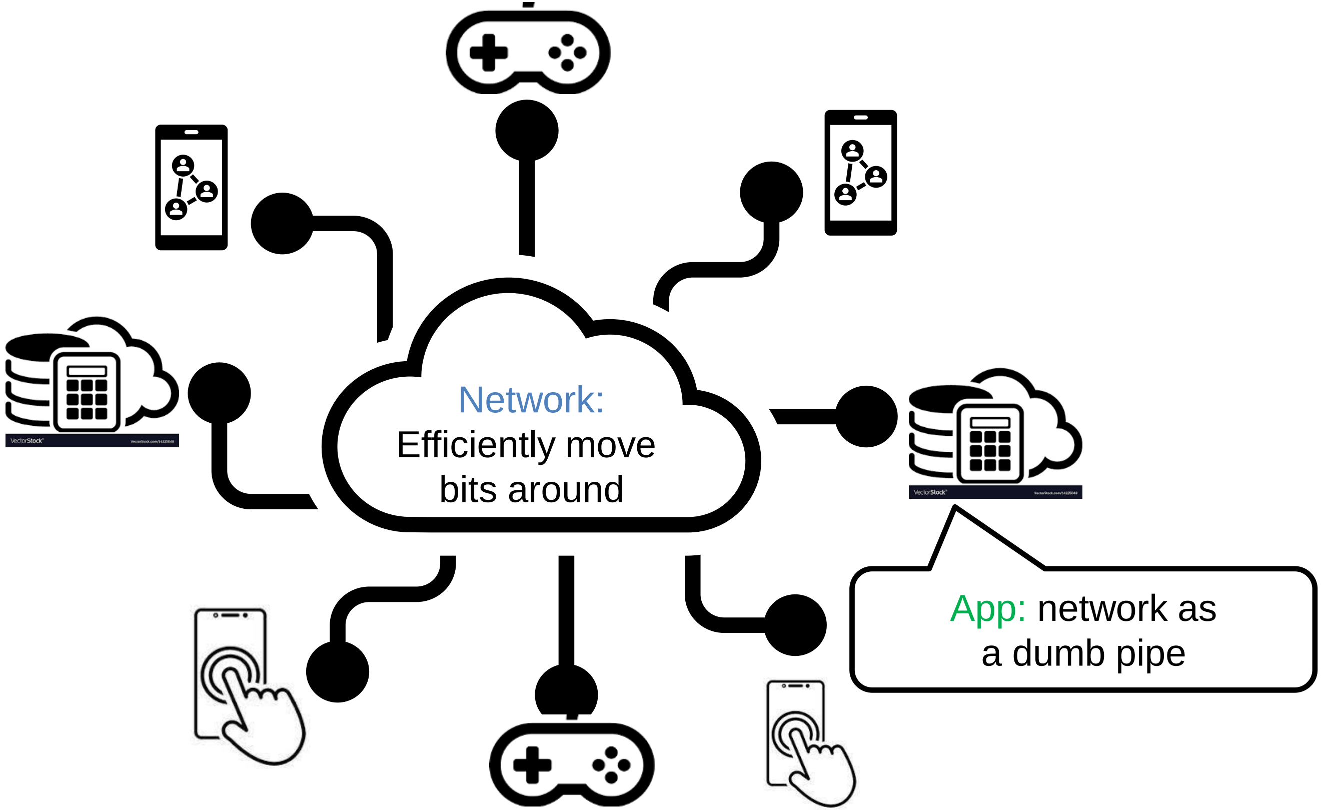
2023 OFA Virtual Workshop

The Next Decade of Networking: Challenges and Opportunities

Aditya Akella, Regents Chair in Computer Sciences

THE UNIVERSITY OF TEXAS AT AUSTIN

A “Simple” Contract Between Apps and the Network

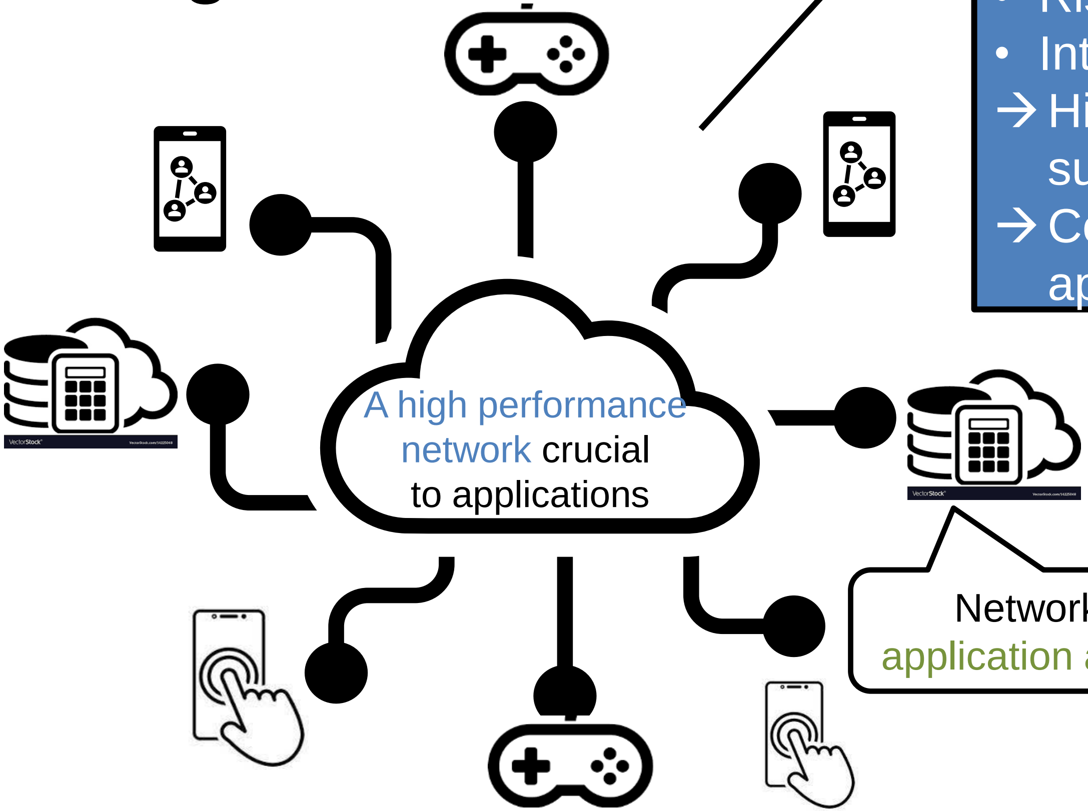


Blurring of Boundaries

- End of Dennard scaling
- Rise of resource disaggregation
- Intense demands of AI and ML

→ High throughput no longer sufficient

→ Complex requirements, application-specificity



Network as an application accelerator

Shifting and Tightening Contracts

Extremely diverse and tight SLOs

Amazon Neptune
amazon KINESIS
aws
mxnet
AWS IoT
aws elemental
Amazon Sumerian
AWS RoboMaker

High throughput, fixed vs elastic

Low latency

Gang-scheduled

High throughput, low latency

Ephemeral vs long running

Durability

Persistence

Real time streaming

Object caches

Geo-distribution

- Extreme performance demands
- Unpredictable, new application workloads
- Compute technology limitations

Perfect storm?
No! Beginning of a golden age!

What We Need

- Extracting performance, while staying as general as possible
- Allow for application and workload tailored functioning for extremely demanding high-volume applications
- A tough balancing act!

The Trajectory of Today's Networking

Claim: *Good starting points today, but they don't strike the right balance and are headed in sub-optimal directions*

- Custom designs that sacrifice generality, expensive to deploy
- Or generality at the expense of performance
- Poor abstractions

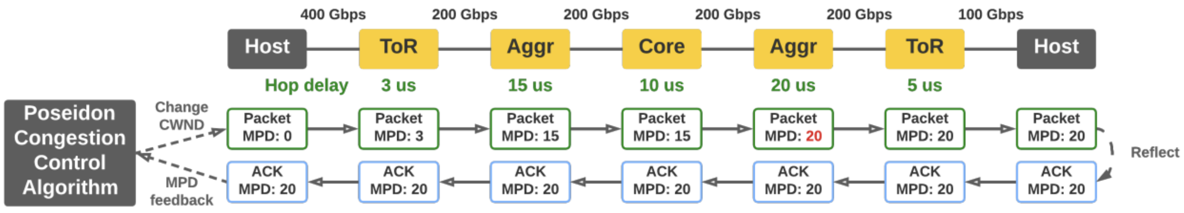
This Talk

Explore the current space and the path forward using two examples:

- Congestion control
- In-network computing

SOTA Congestion Control: The Switch Side

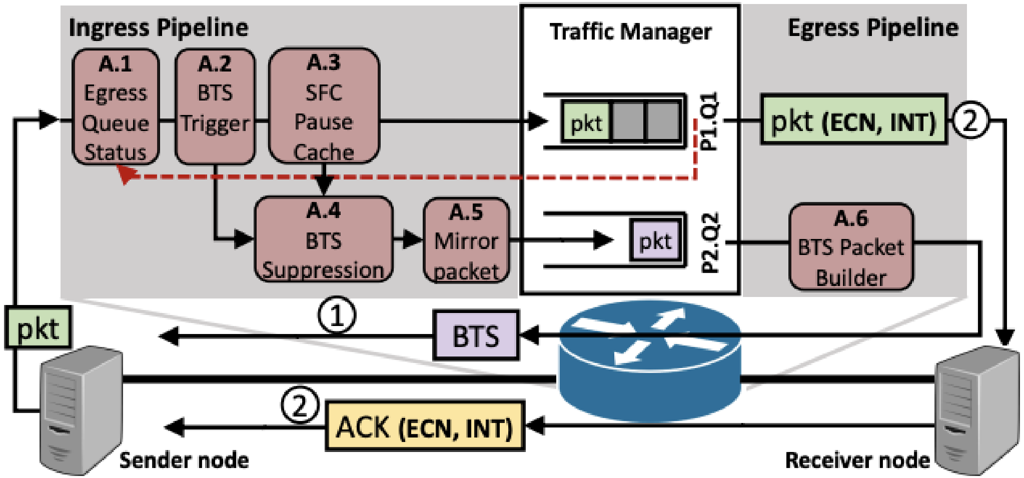
- Early switches: passive observers of congestion, provided imprecise and slow feedback
  - A single or few bits of information (CNP), delivered at RTT time scales
- INT (in-band network telemetry) has changed the game today
  - Detailed congestion signaling
  - Precise per-hop delays and headroom, egress queue measurements
  - Has improved the precision of information



Switch Side Congestion Control

- While precise, the feedback has delay and propagation issues:
  - Large RTT-timescale end-to-end signaling loop
  - Signals are carried by packets that are themselves experiencing congestion
  - Congestion queuing delay can spike up to 1ms → 1-2 orders of magnitude larger than DCN RTT!
- Insight: decouple congestion signaling loop from congestion path
  - Crucial at high link speeds
  - Most messages are well below one BDP
  - End-to-end reaction is not tenable

Back-to-Sender and Source Flow Control



- Switch ingress sends back-to-sender or “BTS” with a pause duration
- Sender instantly stops the affected traffic for the duration (SFC)
- Moves congestion queuing from the switch buffer to the sender buffer
- For near-source control, cache pause time at sender ToRs

Switch Side: The Road Ahead

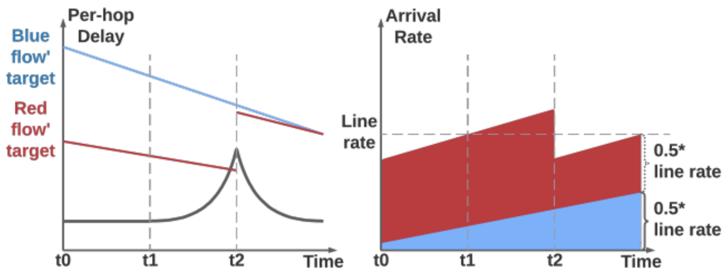
- We can do precise and timely feedback, but what to include?
- Congestion control doesn’t operate in isolation – need to also balance network-wide and app-specific needs
- Rich feedback comes at a trade-off that we need to balance

Switch Side: The Road Ahead

- Design **programmable** feedback to guide these?
  - **Path selection** – allow sender to switch to an alternate path upon experiencing persistent congestion
  - **Multi-pathing** – allow sender to determine how to spread load across multiple available paths, while determining path overlap
  - Inform **application-layer** decisions – guide placement and scheduling decisions to better overlap communication and computation

SOTA Congestion Control: The Host Side

- Early approaches: sender-side, window-driven, drop-based congestion response, all in the software stack
- Many advances:
  - Rate- and delay-based algorithms
  - Integration with INT
    - Google's Poseidon tracks max per-hop delay, adjusts sending rate until observed delay matches sender-rate specific target
  - Hardware accelerated congestion control, e.g., TCP offload engines



Host Side: The Road Ahead (I)

- Marry the speed/efficiency of hardware support with the velocity of software implementations?
- Why do we need a new approach? Aren’t existing hardware-based schemes enough?

Stateful Connection-Oriented Transport Pathologies

- Connection caches can lead to pathologies and huge performance cliffs at extreme scales
- Multiplexing operations atop a few connections can lead to head of line blocking and fate sharing
- Congestion control and loss recovery cannot evolve post-deployment

Connection-oriented-ness appears to be a bad idea

A Different Approach: 1RMA

Judicious division of labor between hardware and software leading to a simple and fixed-function 1RMA NIC aided by 1RMA software

Fixed-function NIC hardware with explicitly allocated resources

- Connection-free independent ops
- Explicitly-finite hardware resource pools
- Solicitation

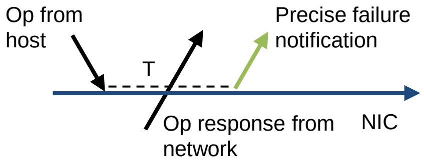
Connection-free security protocol with management ops

Software-driven, hardware-assisted congestion control

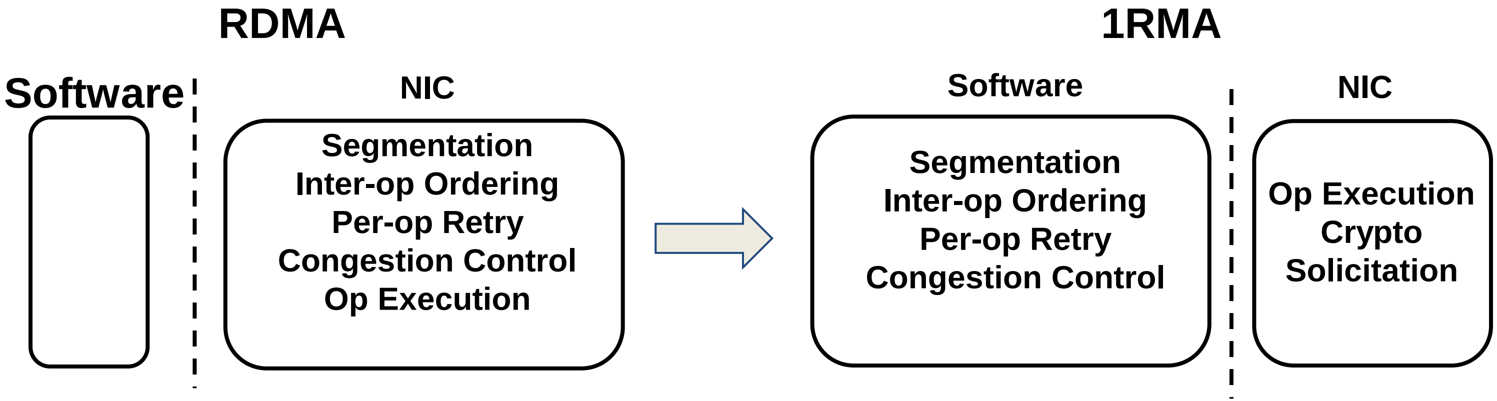
1RMA: Connection-Free Independent Ops

1RMA NIC acts on **fixed-sized ops** and treats them **independently**

- Provides **fail-fast behavior**: NIC ensures op completion within a fixed time; otherwise delivers fast and precise op failure notifications to software



1RMA NIC leaves **retry**, **ordering, congestion control and segmentation to software**



1RMA NIC state does not grow with endpoint pairs

Congestion Control: Main Takeaway

Preserve algorithmic, design, and evolution **flexibility** while enabling fine-grained and **low-level control**

A Rich Design Space: There’s More to Host-side

- **Multi-window** congestion control algorithms: custom fine-grained reactions to different bottlenecks
- **Receiver-driven** and **solicitation-based** congestion control: better modulation of load, avoid the first RTT problems, but unclear how to integrate with applications

A Rich Design Space: There's More to Host-side

- **Accelerator-to-accelerator** communication: how should the host be involved and how to enable co-existence with other transports?
- **Integrating custom ops** into transports: e.g., Scan-and-Read, collectives, dependent connections. What is the right software architecture? Safety and performance guarantees?

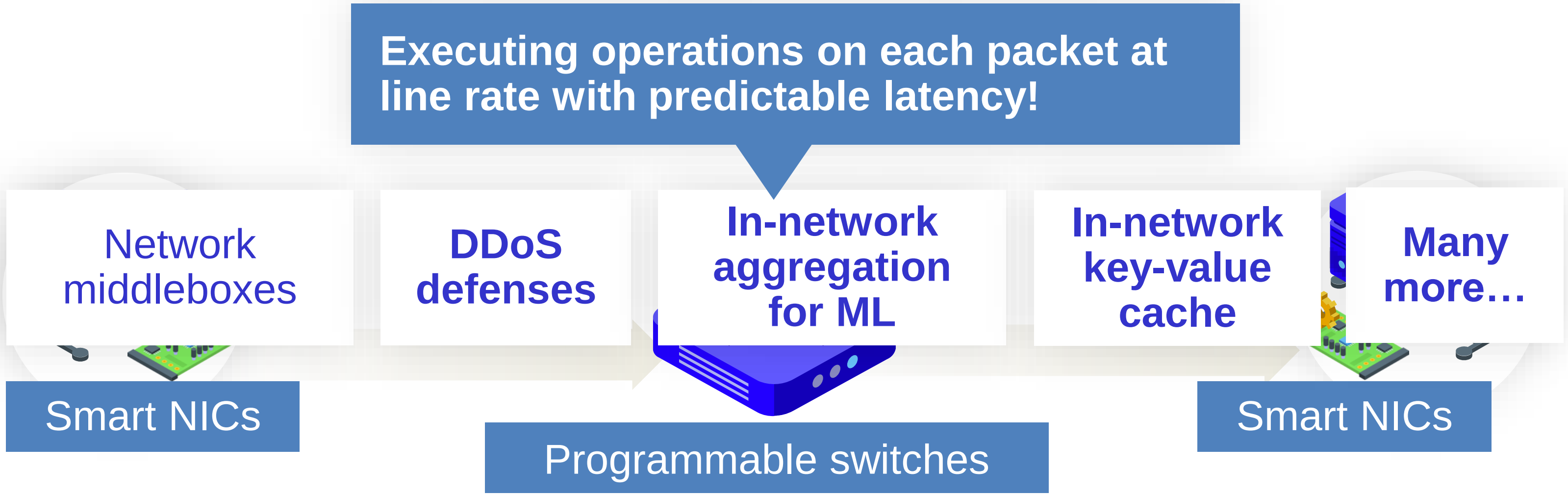
In-Network Computing

- Promises to counter limitations of hardware performance scaling
- NICs and switches, and serving a range of different applications
- Significant performance speed-up, CPU and latency savings
- There is a lot of fertile ground and room for additional work!

SOTA In-Network Computing

- Implement functions in P4 to run atop match-action tables
- Leverage simple stateful processing support on NICs or switches
- Impressive demonstrations, e.g., of ML acceleration
- Specific on-NIC accelerators, e.g., RPC stack offload, RPC load balancing and scheduling
- Exciting area, but growth appears amorphous (to me)

In-Network Computing Examples



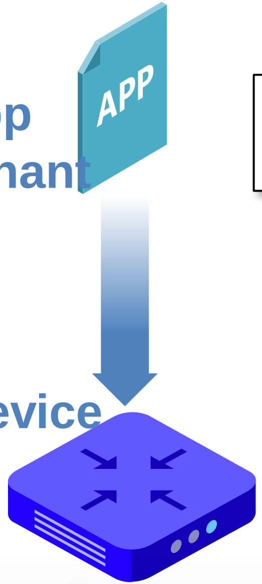
Packet and message processing functions, DMA operations

Collectives, e.g., barriers, reduction operations, and all-to-all shuffles

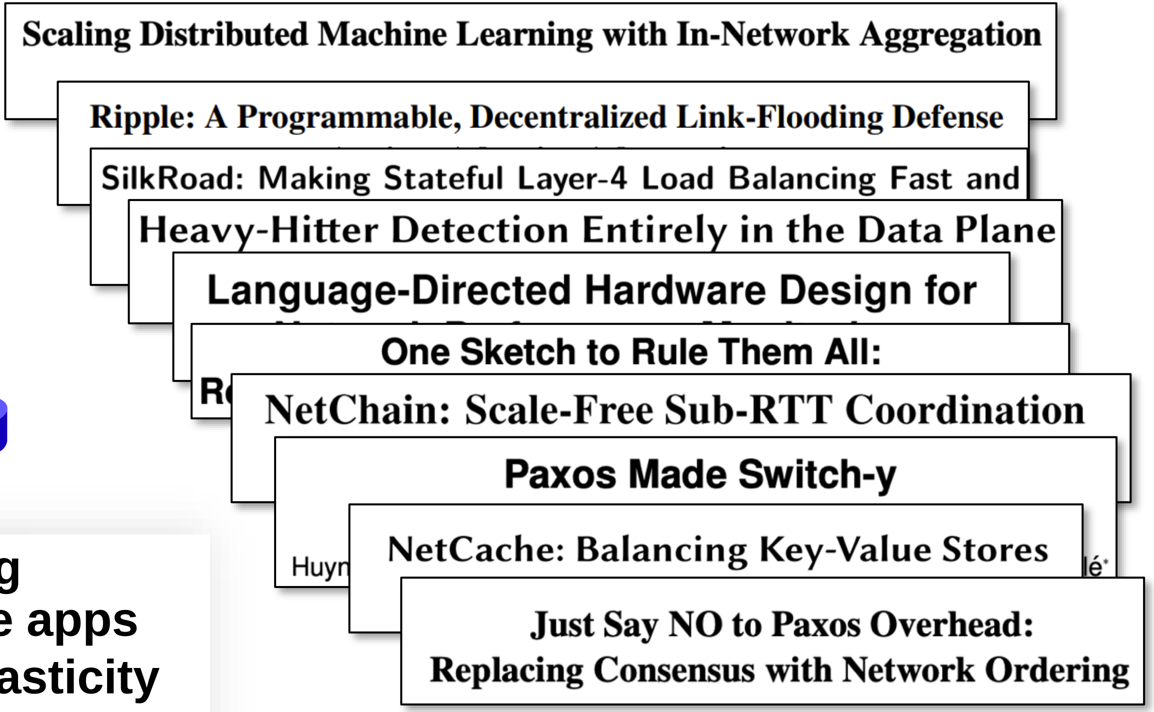
Promising early days but miles to go!

Today’s view of in-network computing

Single app
Single tenant



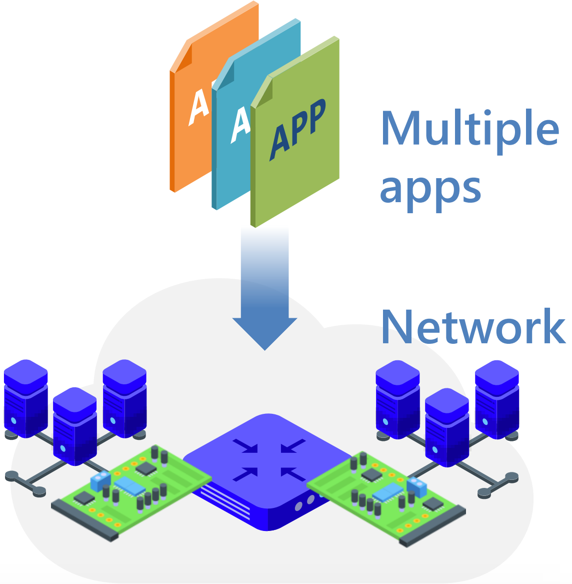
Single device



- **No multiplexing across multiple apps**
- **No resource elasticity**
- **No fault resilience**

The Road Ahead (1)

- No multiplexing across multiple apps
- No resource elasticity
- No fault resilience

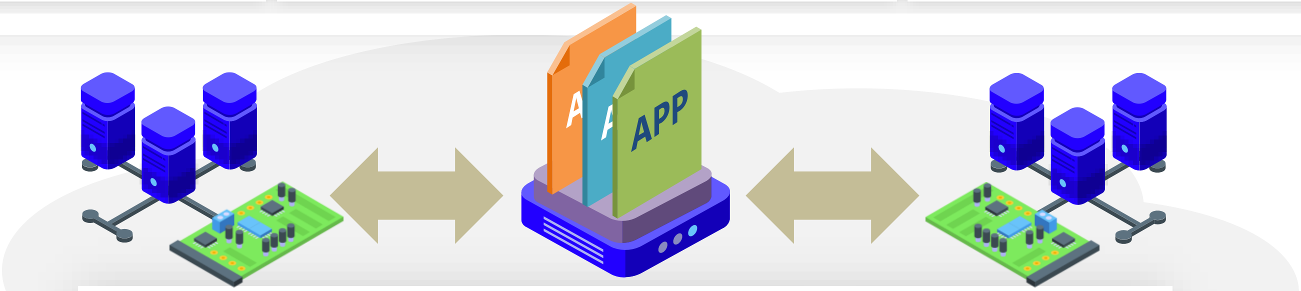


- Multiplexing across multiple apps
- Resource elasticity
- Fault resilience

Elastic and resilient in-network computing

Abstractions & Programming APIs

Virtual memory	Resource multiplexing	Fault tolerance



Runtime environments: Resource management

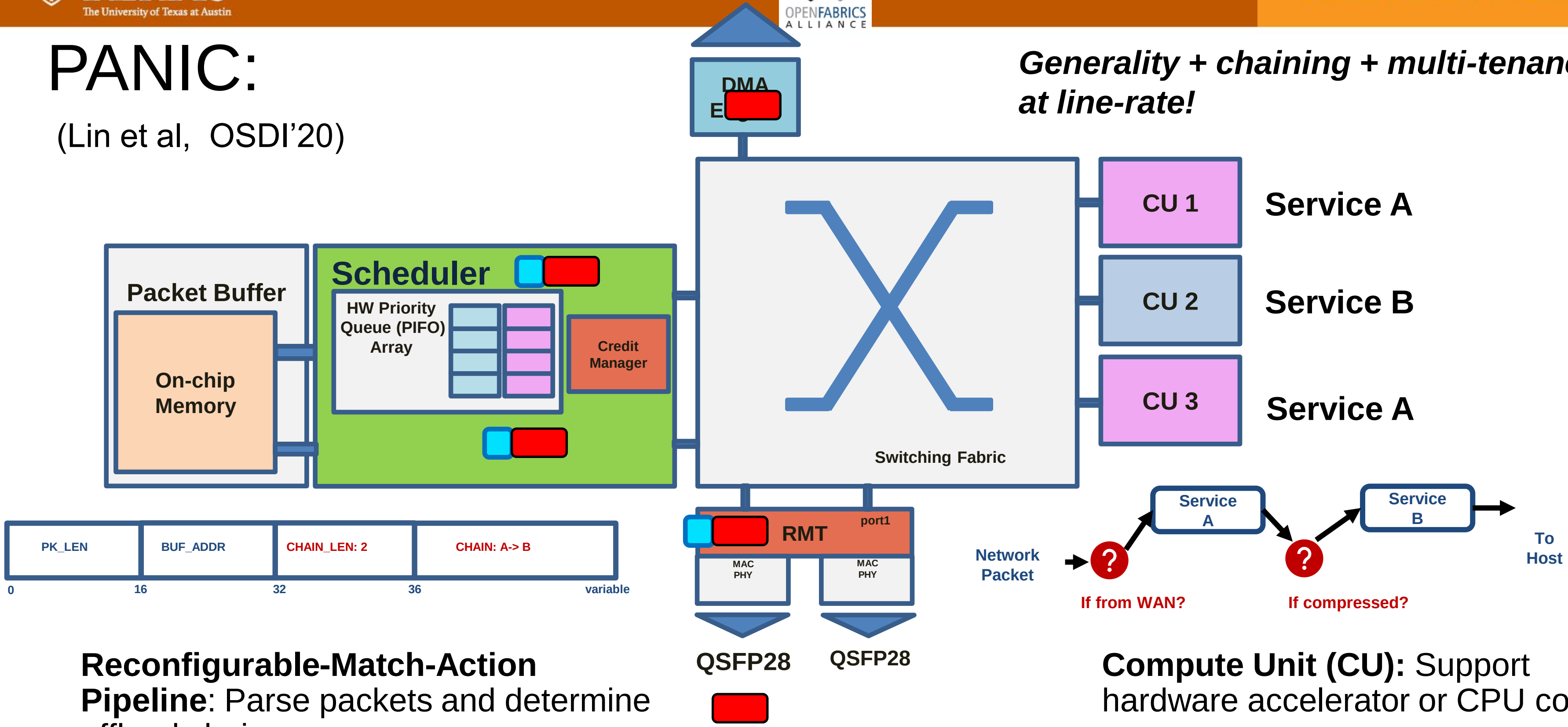
The Road Ahead (1)

- The previous was about missing **software abstractions**
- But there’s also a mismatch today between **hardware** designs and high-level requirements
- SmartNICs and programmable switches assume single tenant with fixed/rigid requirements
- **Multiple tenants**, possibly needing customizable chains of functions, is the future

PANIC:

(Lin et al, OSDI'20)

Generality + chaining + multi-tenancy – at line-rate!



Reconfigurable-Match-Action Pipeline: Parse packets and determine offload chain

Central Push-in-First-Out Scheduler: enforce isolation policies and schedule chains/packets

Compute Unit (CU): Support hardware accelerator or CPU core

High-throughput Switching Fabric: Interconnects different hardware resources.

The Road Ahead (2): New Programming Languages

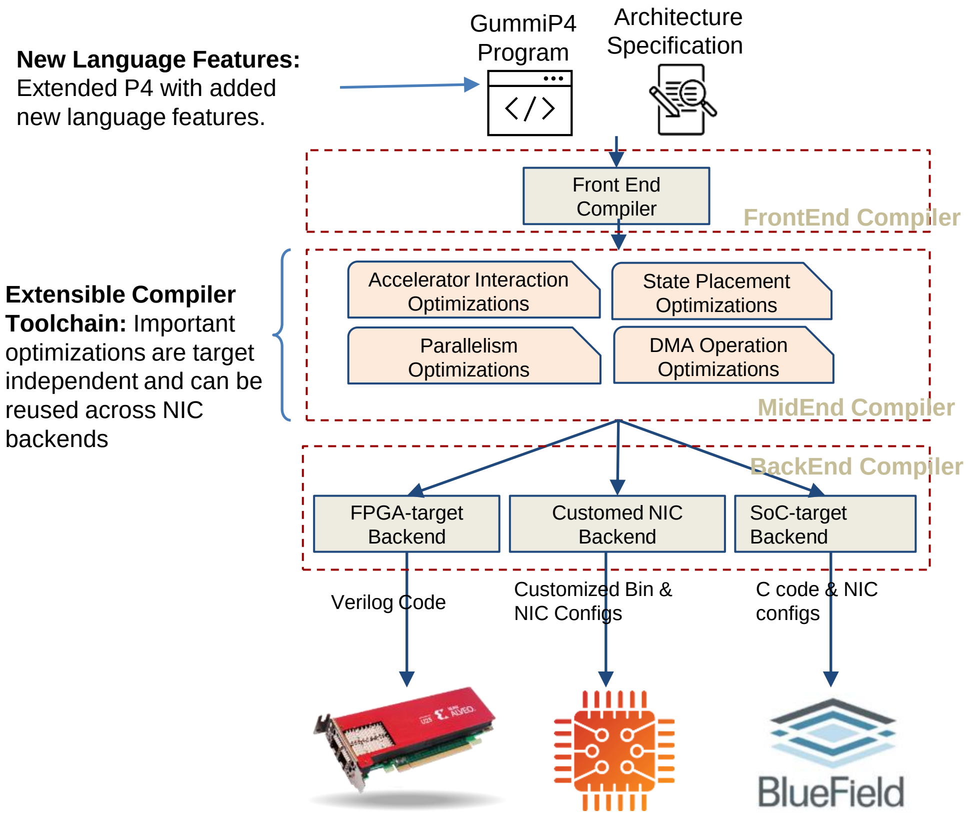
- P4 has been great, but both the language and the ecosystem have drawbacks
- P4 unsuited for the rich in-network computing applications, especially on SmartNICs
- Custom extensions to P4 limit portability, increase developer burden
- P4 abstractions are a poor fit for emerging NICs

The Need for a New Programming Language and Toolchain

- Control over parallel processing and barriers
- Richer interface to ASICs than “extern”
- Message processing support
- Simple building blocks that aid backend development

GummiP4

- GummiP4’s new language features:
  - ***Expressiveness:*** Provide new constructs to easily write interesting and useful SmartNIC programs.
  - ***Compiler Assistance***: Aids the compilation process by allowing the programmer to expose domain-specific information to the compiler.
- GummiP4’s compiler toolchain:
  - ***Optimal Performance:*** Generate highly optimized NIC programs that require minimal resources and execute quickly.
  - ***Extensible:*** Our compiler design is extensible, as all important optimizations happen in a target-independent manner. Therefore, hardware vendors can easily write new backends for upcoming SmartNICs.



In-Network Computing: Main Takeaways

- Ground-up support for **multi-tenancy** both at hardware and software levels
- Ground-up **new abstractions** to program and control emerging NICs

Parting Thoughts

- Exciting times for network technology, but providing **ground-up** and **low-level** control is key to preserving flexibility
- Congestion control continues to be a challenge, but a stable approach is to provide rich feedback to aid programmable logic
- Multi-tenancy support appears to be an obvious missing piece
- New abstractions and programming models are sorely needed, especially on the NIC front