How and when should we use programmable switches?

ECE/CS598HPN

Radhika Mittal

Which paper did you like the most?

- (A) BeauCoup
- (B) Elmo
- (C) NetCache
- (D) Silkroad

Which paper did you dislike the most?

- (A) BeauCoup
- (B) Elmo
- (C) NetCache
- (D) Silkroad

Did you change your opinion after reading today's papers?

- (A) Yes
- (B) No
- (C) Maybe

Other networking usecases

- Load balancing:
  - HULA: Scalable Load Balancing Using Programmable Data Planes, SOSR'16
- Congestion control:
  - Evaluating the Power of Flexible Packet Processing for Network Resource Allocation, NSDI'17
    - *Support RCP and XCP on programmable switches*
  - HPCC: High Precision Congestion Control, SIGCOMM'19
    - *Obtain precise link information for congestion control*
- A new protocols for more efficient L2 switching
  - The Deforestation of L2, SIGCOMM'16
- …..

Other app-level usecases

- NetChain: in-network key-value store.
- NetLock: Switching support to manage locks.
- DAEIT: In-network data aggregation
- NetPaxos: implement Paxos on programmable switches
- NoPaxos, Eris: in-network primitives for distributed protocols.
- …..

How should we use programmable switches?

Two papers

- When should the network be the computer?
  - Dan Ports and Jacob Nelson, HotOS'19

- Thoughts on Load Distribution and the Role of Programmable Switches
  - James McCauley, Aurojit Panda, Arvind Krishnamurthy, Scott Shenker, SIGCOMM CCR Editorial.

When should the network be the computer?

Trade-offs

- Low latency and high throughput, at the cost of
  - Flexibility
  - Storage

Key Arguments (or Principles)

- Offload primitives, not applications
  - Make primitives reusable
- Keep state out of the network
  - Preserve fate-sharing
- Minimal interference with existing network functionality.

Which primitive are good offloading candidates?

- Criteria:
  - No. of operations per packet
    - Typical: O(1) or O(n) where n = length of the packet
  - Amount of state stored in switch required to process a packet.
    - O(1), O(n), O(s), where s = application's working set size.
  - For a given packet, how many packets are produced
    - O(1), O(r), O(1/r)

- Packet gain is an important benefit of "in-network" computing.

Offloading Criteria

	Ops/pkt	Amt of State	Packet Gain
BeauCoup	O(1)	O(no. active flows)	O(1)
Elmo	O(1)	O(1) (some constant)	O(r)
NetCache	O(1)	O(cache size)	O(1)
SilkRoad	O(1)	O(no. of connections)	O(1)

Table from the paper

In-network primitive	Ops/packet	State/packet	Packet gain	Class	Dominant
Network sequencing [26, 27]	$O(1)$	$O(1)$	$O(\|\text{replicas}\|)$	CC+	Gain
Replicated storage [18]	$O(1)$	$O(\|\text{dataset size}\|)$	$O(1)$	CLC	State
Caching [19, 29]	$O(1)$	$O(\ln(\|\text{dataset size}\|))$	$O(1)$	CLC	State
DNN training (allreduce) [30, 38, 39]	$O(\|\text{packet}\|)$	$O(\|\text{packet}\|)$	$O(1/\|\text{replicas}\|)$	LL-	Gain
DNN inference [12]	$O(\|\text{input size}\|^2)$	$O(\|\text{model size}\|)$	$O(1)$	GLC	Ops
Database reductions [25]	$O(\|\text{packet}\|)$	$O(\|\text{elements}\|)$	$O(1/\|\text{replicas}\|)$	LL-	Gain
Database hash joins [25]	$O(1)$	$O(\|\text{elements}\|)$	$< O(1)$	CL-	State
Virtual networking [11]	$O(1)$	$O(\|\text{flow table}\|)$	$O(1)$	CLC	State
In-band network telemetry [22]	$O(1)$	$O(1)$	$O(1)$	CCC	Ops

Other challenges

- Scale and decentralization
- Multi-tenancy and isolation
- Encryption
- Interoperability
- ....

Your opinions

- Pros
  - Talks about what "should" vs what "can" switches do.
  - Three dimensional classification.
  - In-network computing principles.
  - List of challenges.
  - Comprehensive coverage of different usecases.
  - Completely impartial (??)

Your opinions

- Cons
  - How general can primitives be in practice?
  - Justification for the three axes.
    - Is minimal state requirement too constraining?
    - Difference between asymptotic and empirical bounds.
    - What's missing?

Your opinions

- Ideas
  - Expanding the three axes (include encryption, multi-tenancy, possibility of network failures, etc).
  - Empirically evaluate the effectiveness of the classification.
  - Supporting multiple applications, handling interoperability.
  - What are alternative designs for applications that are not a good fit for programmable switches?

Thoughts on Load Distribution and the Role of Programmable Switches

Relationship with E2E arguments

- Cannot entirely appeal to E2E argument
  - E2E talks about which functionality is part of network layer.
  - The question here is what infrastructure is used for implementing the functionality (servers or switches).
  - *Although some insights could still be applicable….*

Alternatives for switch-based implementation

- Load balancing (SilkRoad)

- In-network Caching (NetCache)

Limitations of SilkRoad

- Requires large amount of state to be stored in the switches. What if we run out of space?

- Does not allow policy flexibility.

Alternative Designs for SilkRoad

- DIP information can be maintained by the client and stored in the packet header field.
  - Either update the destination address for subsequent requests.
  - Other fields: TCP timestamp, QUIC conn id, MPTCP destination port….
- Use consistent hashing in switches. Servers redirect incorrectly received packets.
  - Servers must maintain the per-connection mapping: done via a centralized controller or message exchange with other backend servers.
- In both cases, state is stored at endhosts, and switches perform routing.

Limitations of NetCache

- Limits on the size of key and value.
- Limits on switch memory.
  - Approximate datastructures to compute statistics.

Alternative for NetCache

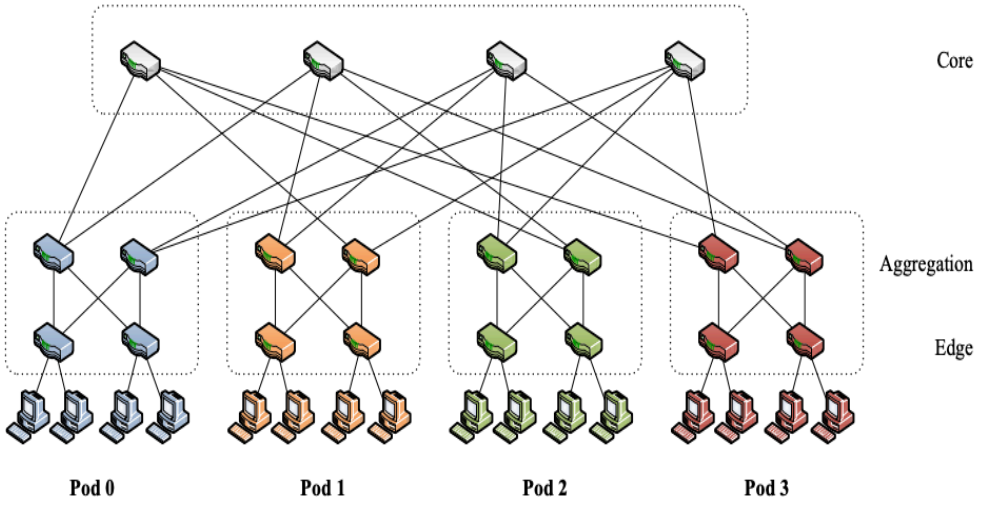
- Replicate popular keys on other *servers*.
- Maintain key access statistics in the servers.
- Switches maintain rules on which key is replicated in which servers.

In both alternative designs

- Complex processing and state management handled by the servers.
- Switches responsible for steering (appropriately forwarding) the packets.

In-network data aggregation

- Limited algorithms can be implemented in switches.
- Other alternatives to minimize incast issues.
- Co-locate switches with compute accelerators.



In-network consensus protocols

- Unclear whether performance of consensus protocol is a limiting factor.

Reasonable usecases of programmable switches

- (Congestion aware) network load balancing
- Network telemetry
- Packet scheduling
- Congestion control

Reasonable usecases of programmable switches

- (Congestion aware) network load balancing, network telemetry, packet scheduling, congestion control.

- Why?
  - Need access to packet counters. Host-based solutions may not be viable.
  - Impact multiple applications (not specific to just one).

Your opinions

- Pros
  - Both sides of the story:
    - Examples of apps that can remove logic from switches.
    - Good usecases of programmable switches.

Your opinions

- Cons
  - Lack of evaluation
  - Arguments driven by current hardware limitations.
  - Does not provide a broad argument, only looks at specific applications.

Your opinions

- Ideas
  - Evaluate alternative designs.
  - Broader analysis of how applications can benefits from splitting forwarding from computation.
  - A framework (simulator/emulator) to quickly verify the benefits/harm of offloading app functionality to switches.
  - Can switches use external memory?
  - Explore the use of other compute accelerators instead?

Which arguments are shared by both papers?

On which aspects do the two papers differ from one another?

Which paper do you agree with more?

- (A) When should network be the computer?
- (B) Thoughts on ….. programmable switches
- (C) Both
- (D) Neither