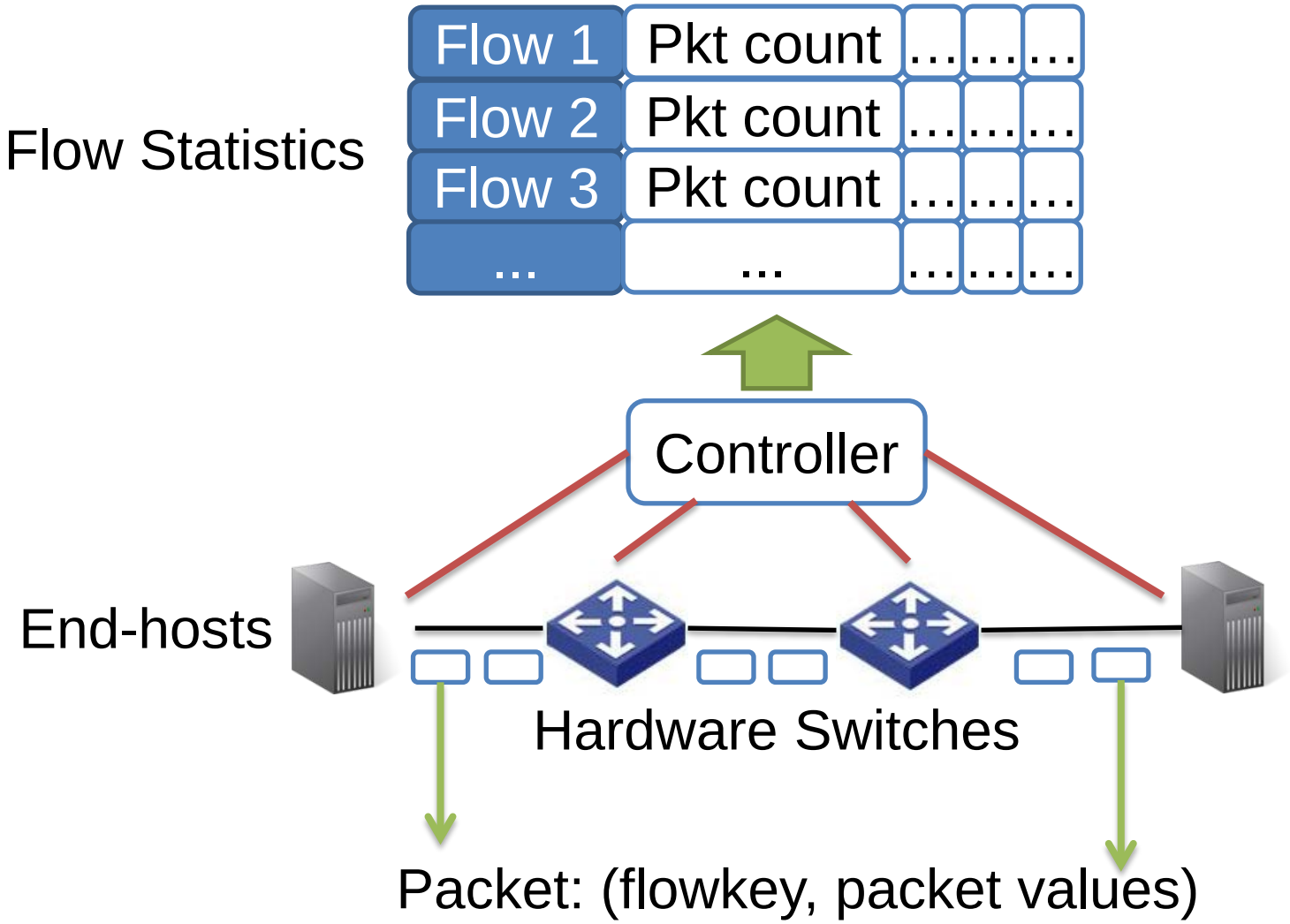
OmniMon: Re-architecting Network Telemetry with Resource Efficiency and Full Accuracy

Qun Huang, Haifeng Sun, Patrick P. C. Lee

Wei Bai, Feng Zhu, Yungang Bao

Flow-level Network Telemetry

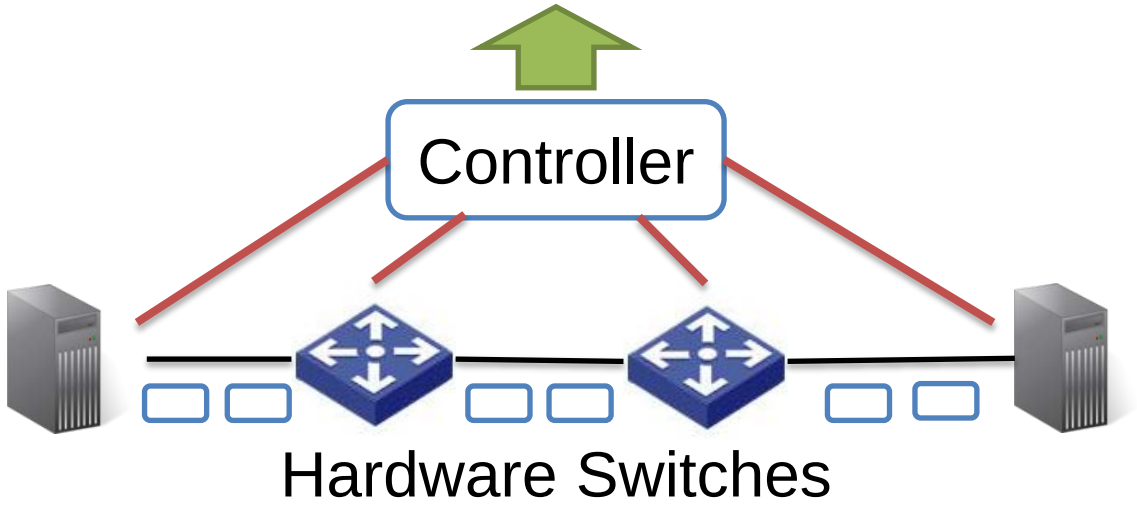


Goal

Flow Statistics

Flow 1	Pkt count	…	…	…
Flow 2	Pkt count	…	…	…
Flow 3	Pkt count	…	…	…
…	…	…	…	…

Full Accuracy



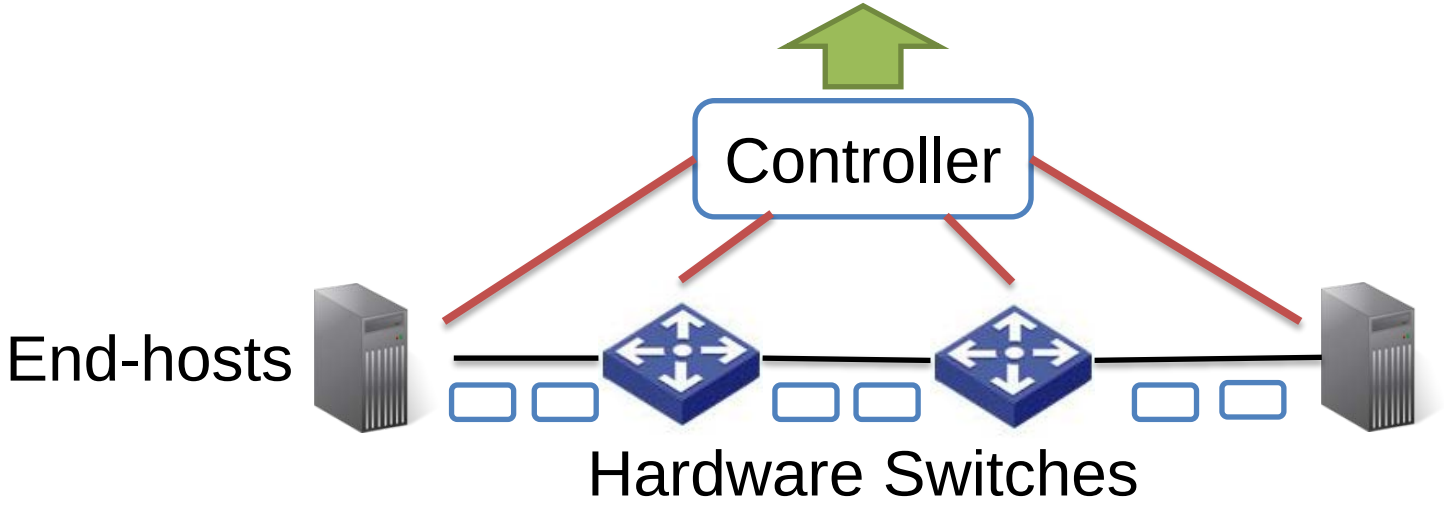
Resource Efficiency

Full Accuracy

Flow Statistics

Flow 1	Pkt count	…	…	…
Flow 2	Pkt count	…	…	…
Flow 3	Pkt count	…	…	…
…	…	…	…	…

1. Always-on: all time intervals
2. Network-wide: all devices
3. Complete: all flows
4. Correct: zero per-flow error



Resource Efficiency

Flow Statistics

Flow 1	Pkt count	…	…	…
Flow 2	Pkt count	…	…	…
Flow 3	Pkt count	…	…	…
…	…	…	…	…

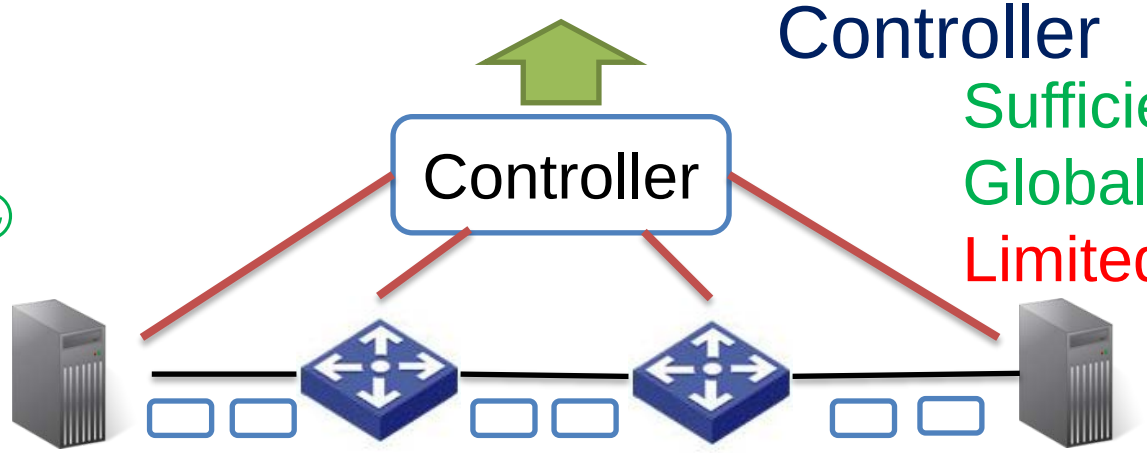
1. Always-on: all time intervals
2. Network-wide: all devices
3. Complete: all flows
4. Correct: zero per-flow error

Controller

Sufficient CPU and memory ☺
Global visibility ☺
Limited bandwidth ☹

End-Hosts

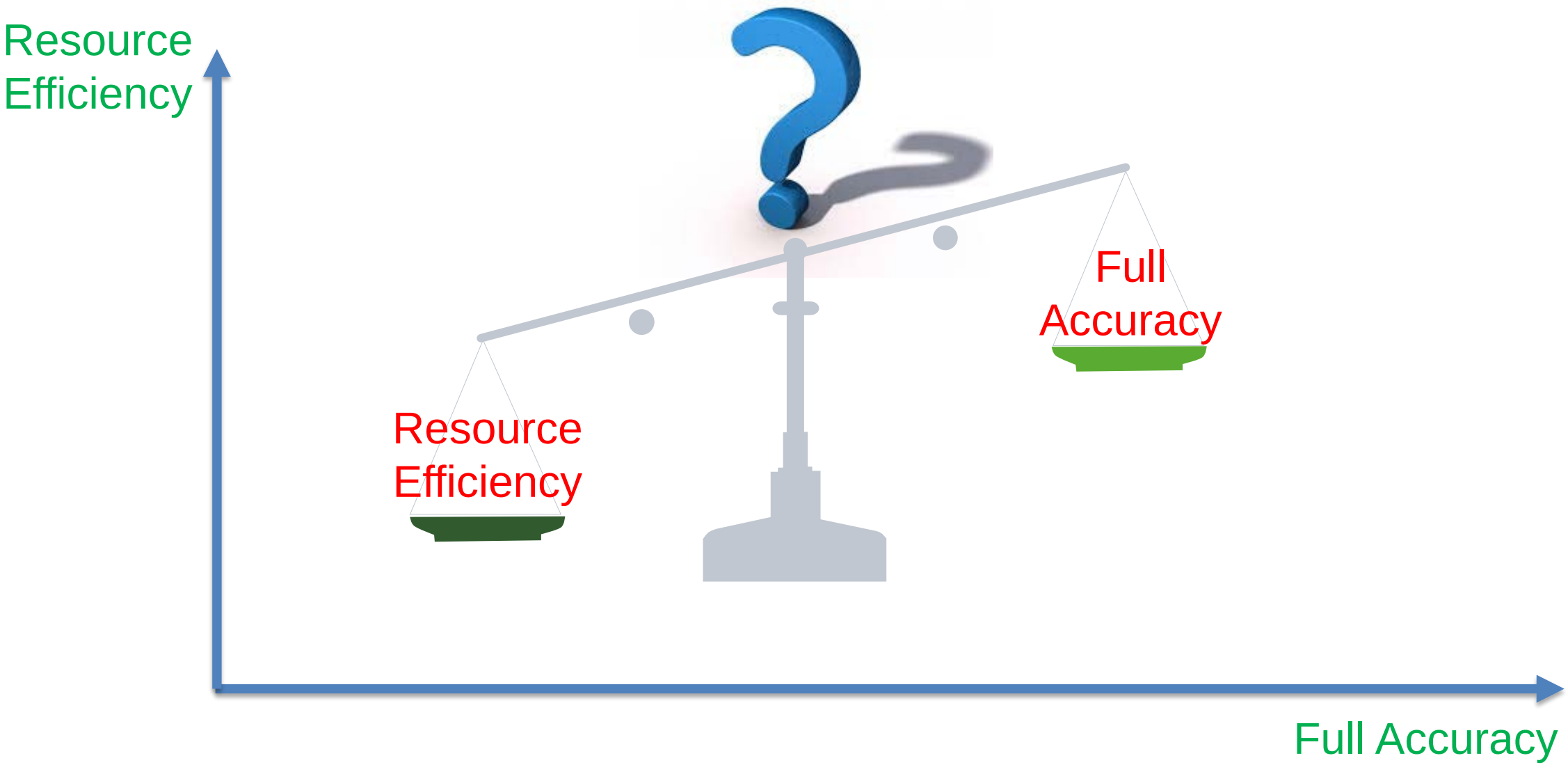
Sufficient memory ☺
Programmability ☺
Slow CPU ☹
Limited visibility ☹



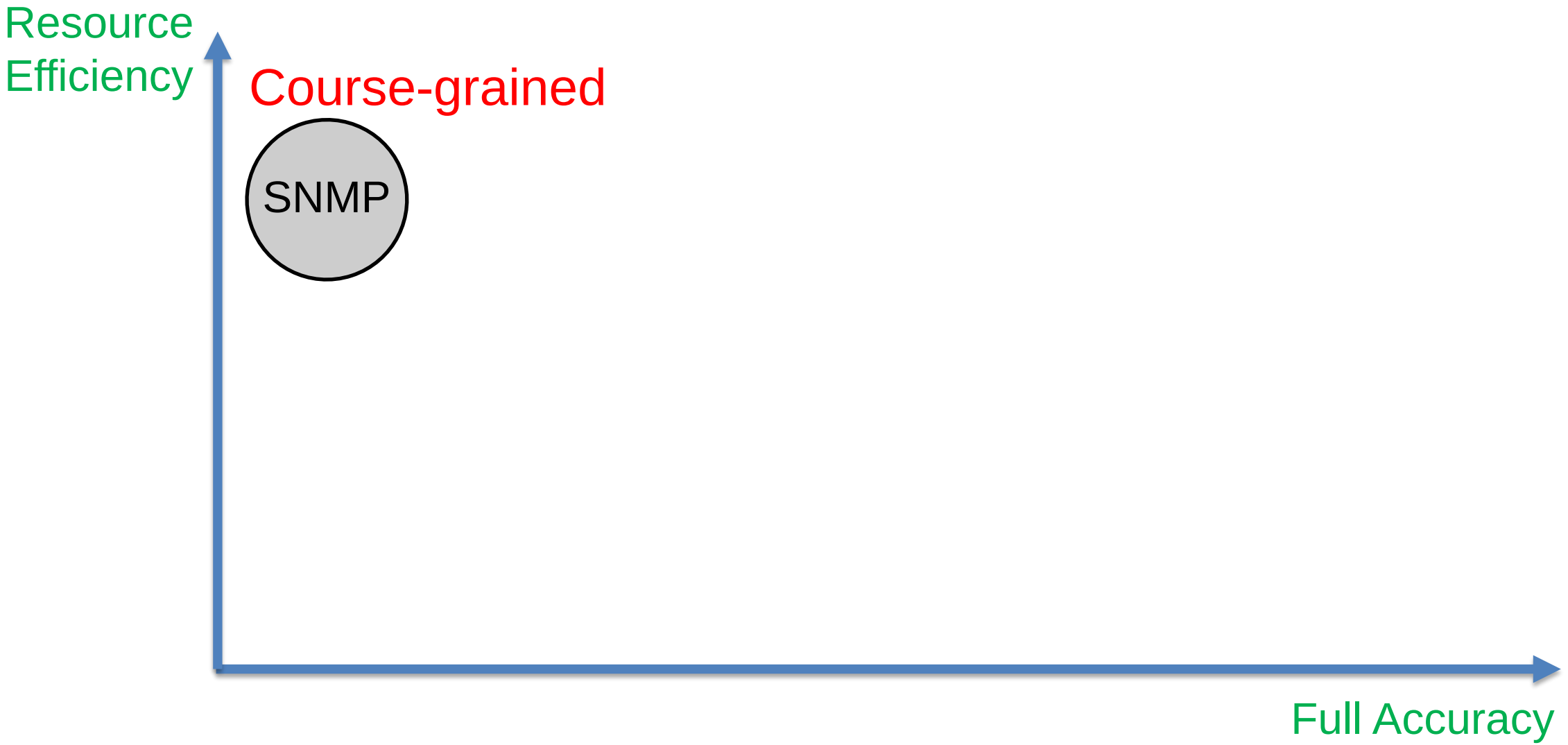
Hardware Switches

Fast ASIC ☺
Limited memory ☹
Limited programmability ☹

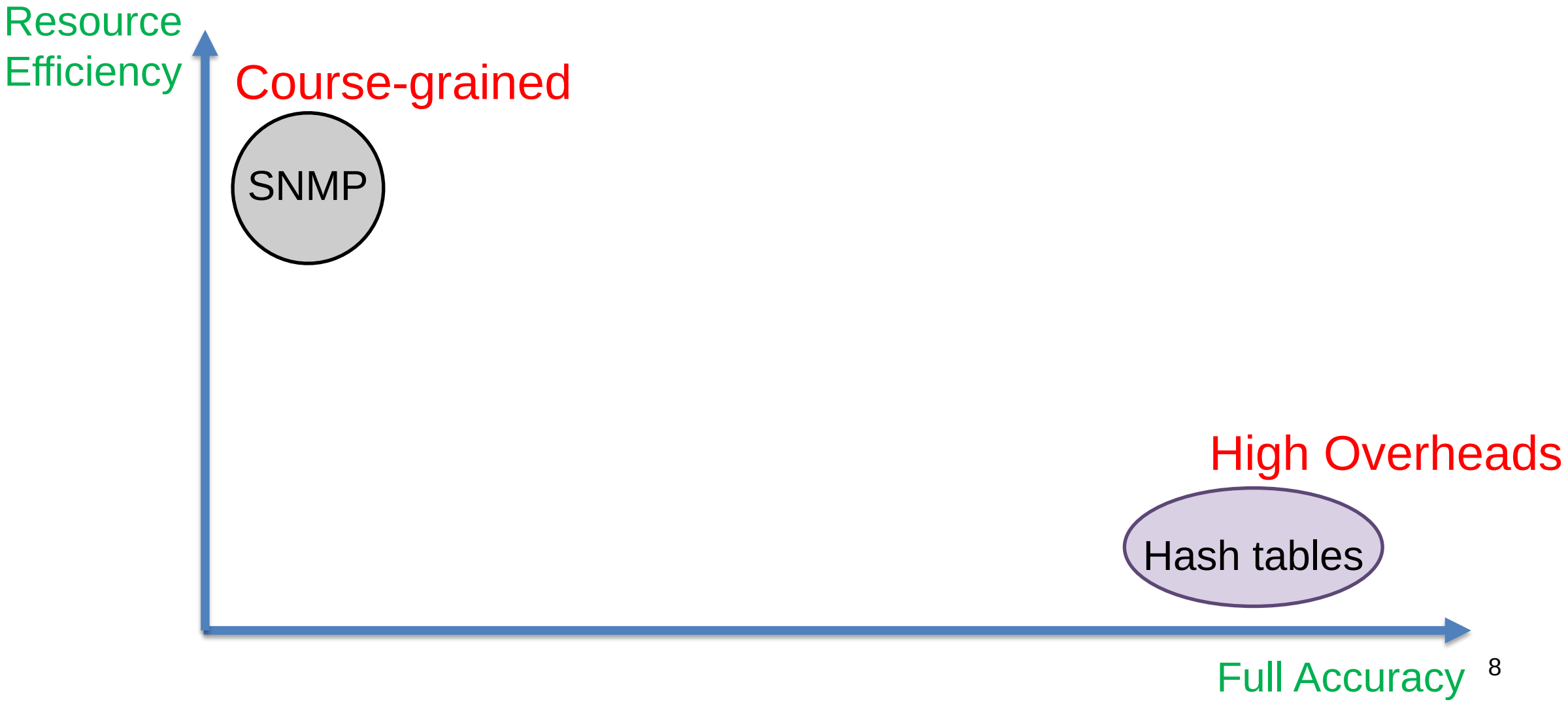
Existing Approaches: Trade-offs



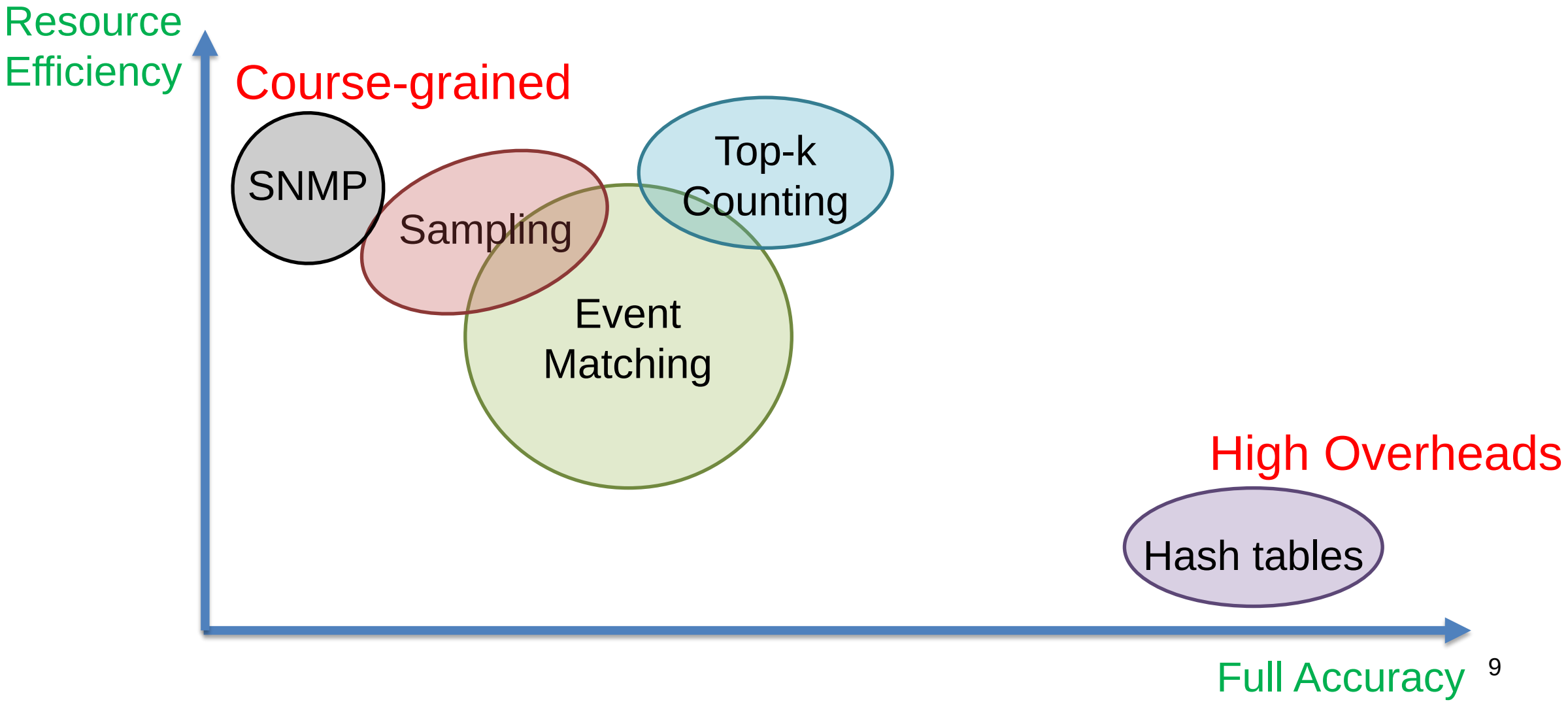
Existing Approaches: Trade-offs



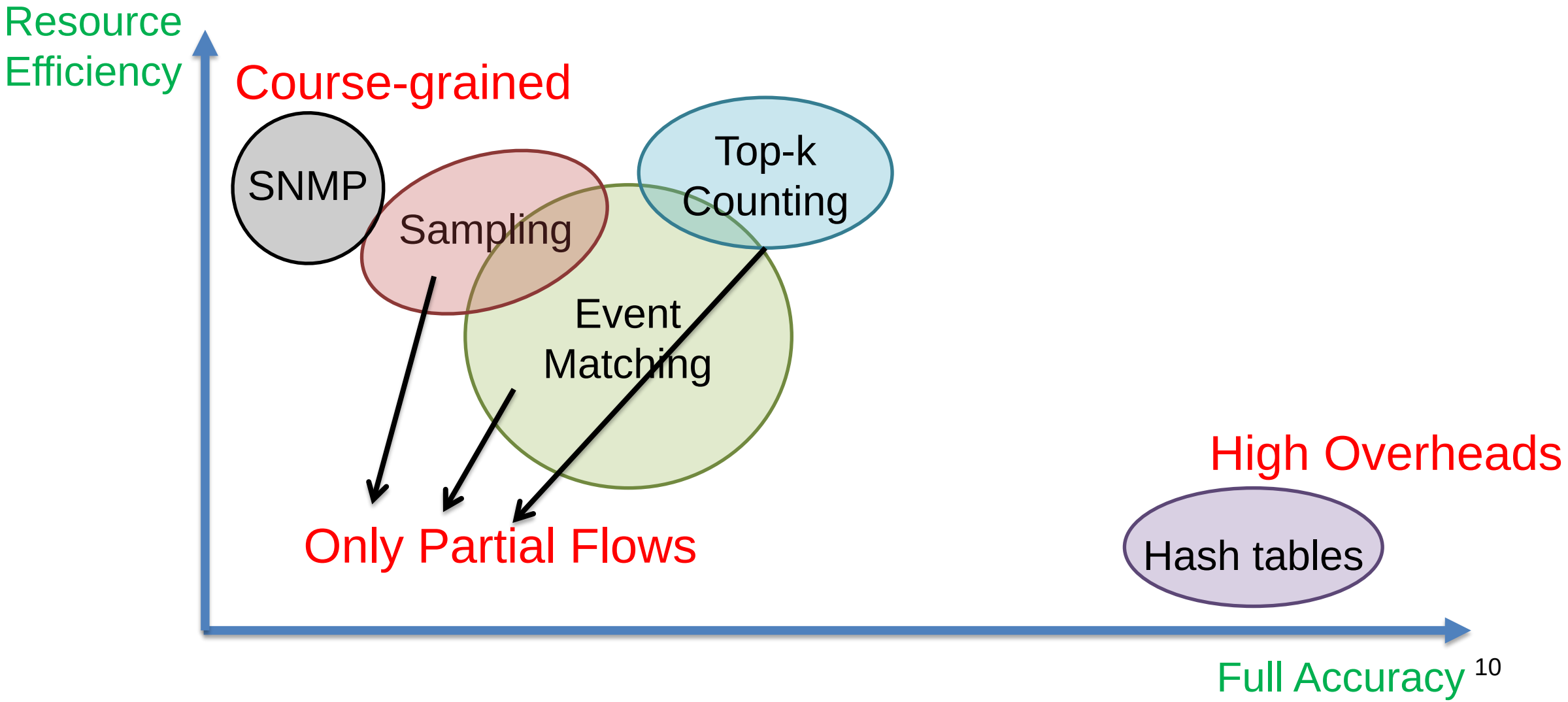
Existing Approaches: Trade-offs



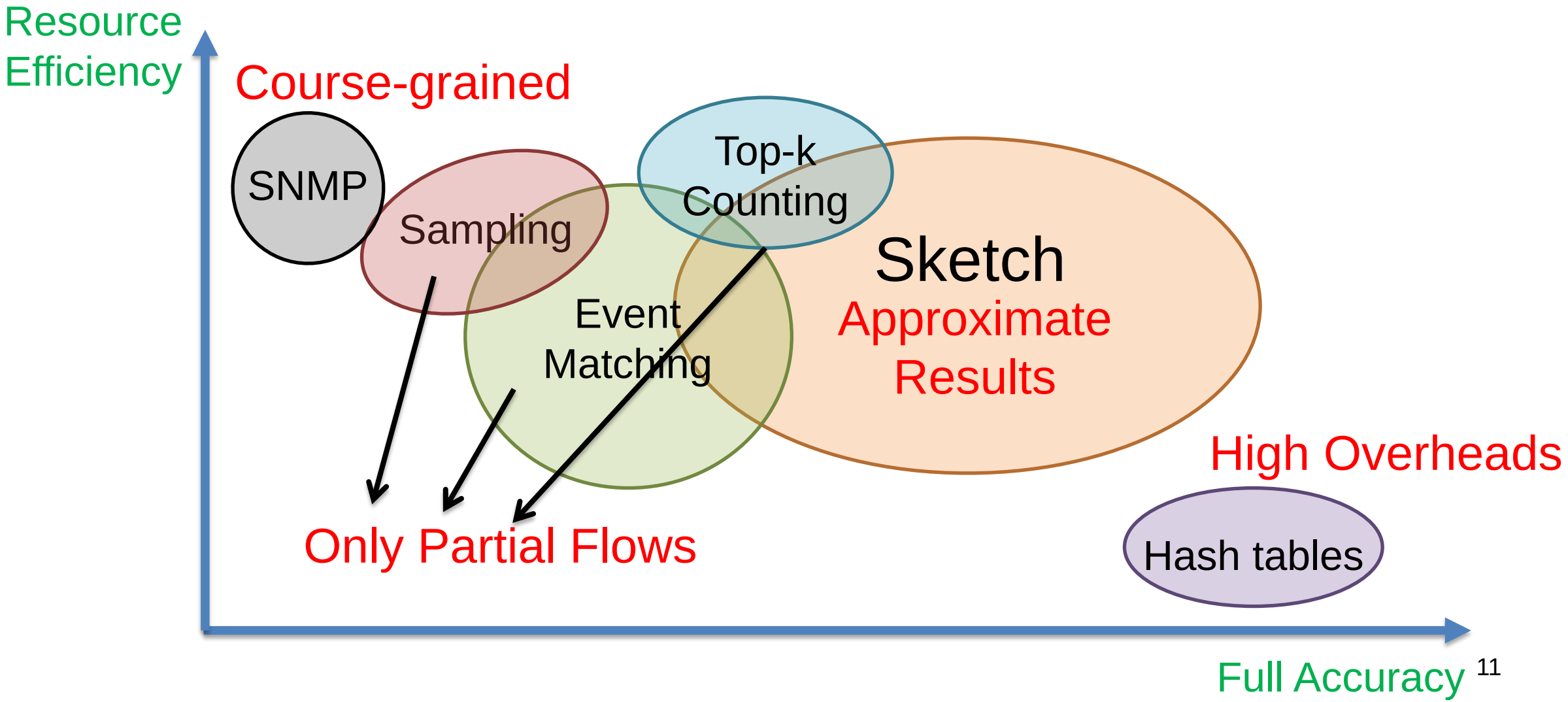
Existing Approaches: Trade-offs



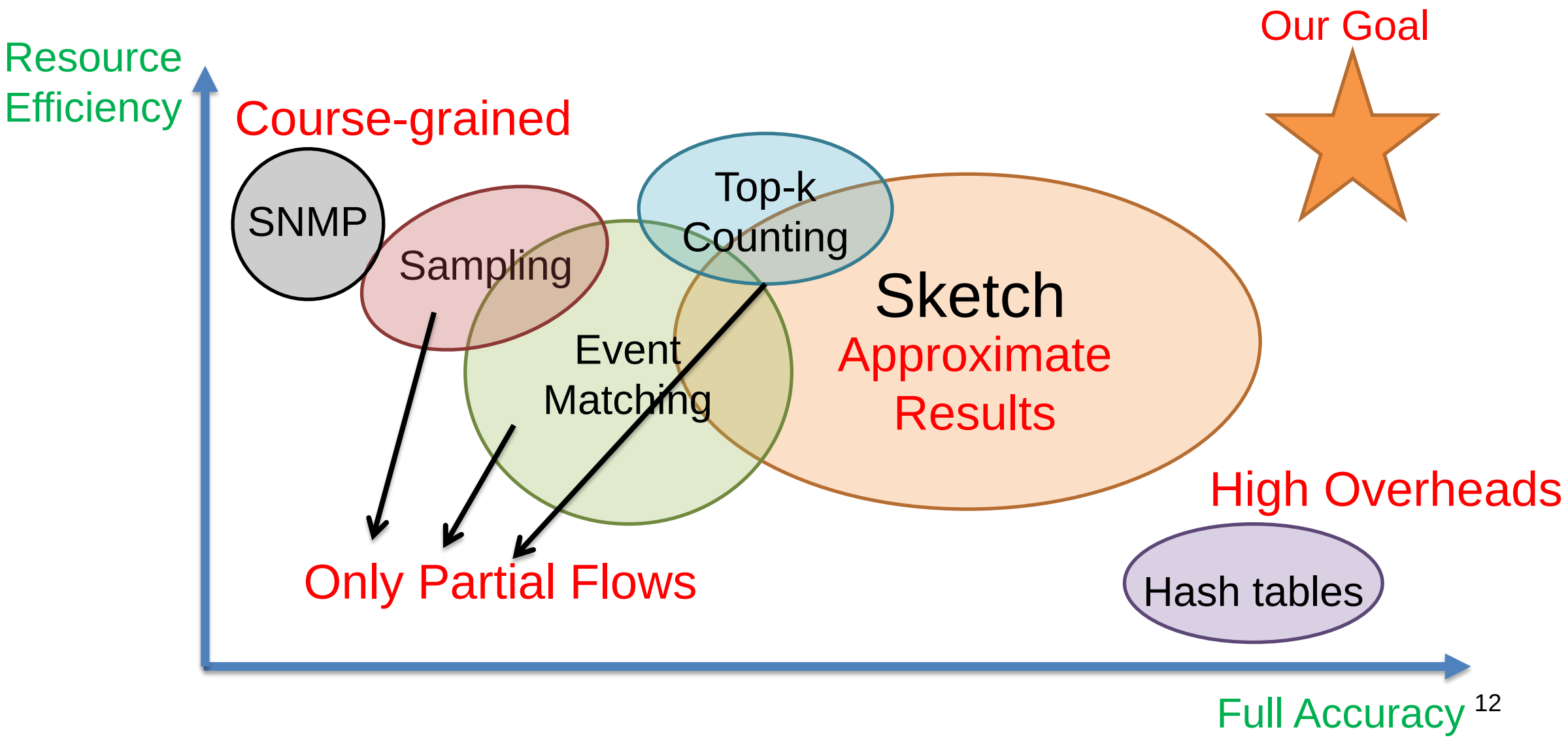
Existing Approaches: Trade-offs



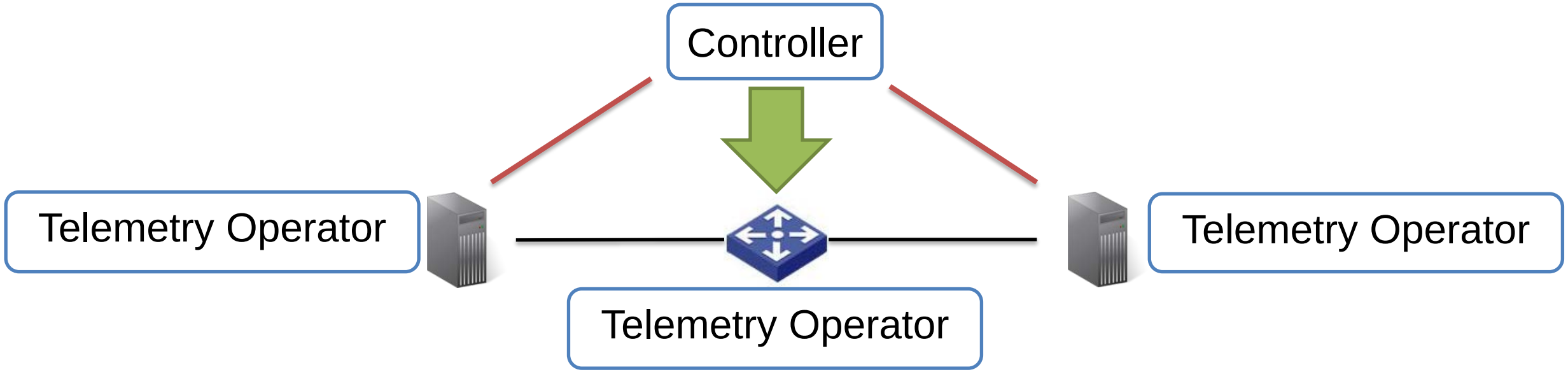
Existing Approaches: Trade-offs



Existing Approaches: Trade-offs

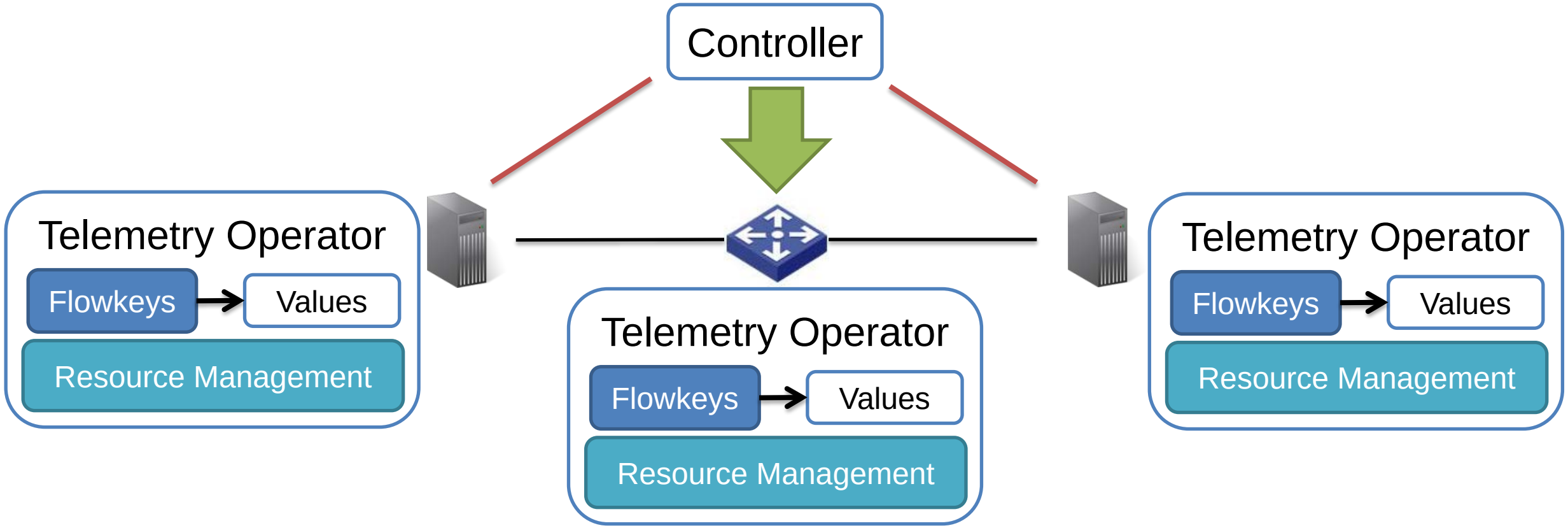


Root Cause



Operators are executed individually with limited collaboration

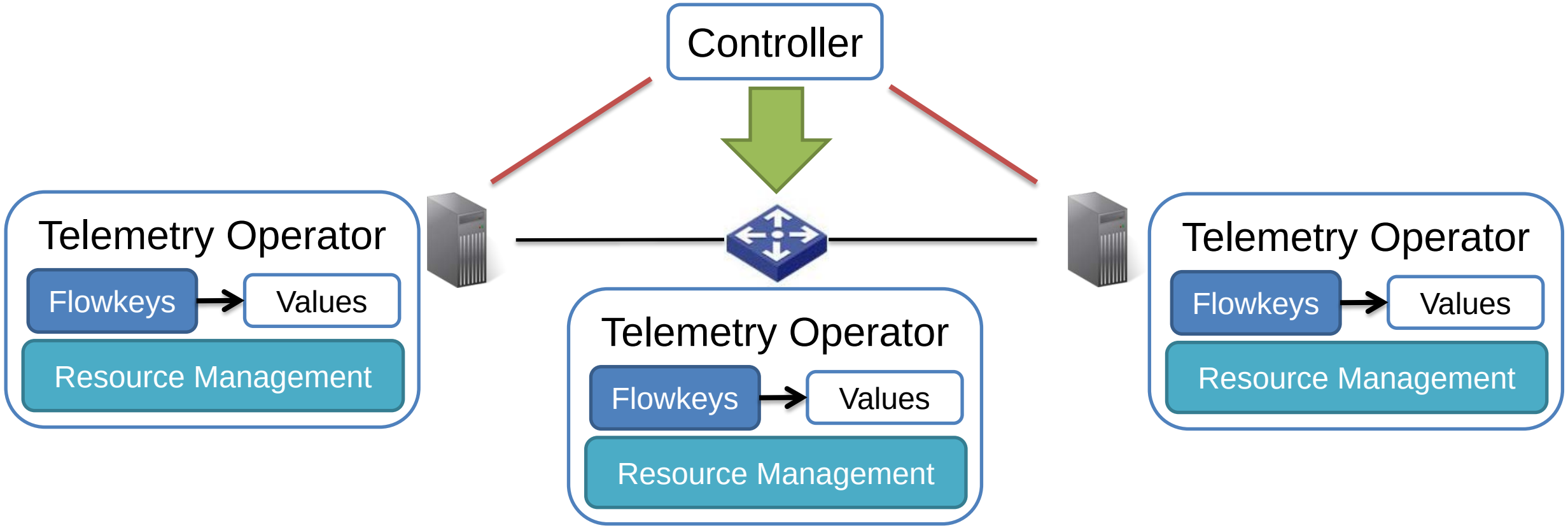
Root Cause



Operators are executed individually with limited collaboration

Operators have to be heavy and sacrifice accuracy

Root Cause



Operators are executed individually with limited collaboration

Operators have to be heavy and sacrifice accuracy

OminMon

Re-architect network telemetry by distributed design

Question 1

Coordinate different entities for network telemetry?

Question 2

Reliable guarantees for the coordination?

OminMon

Re-architect network telemetry by distributed design

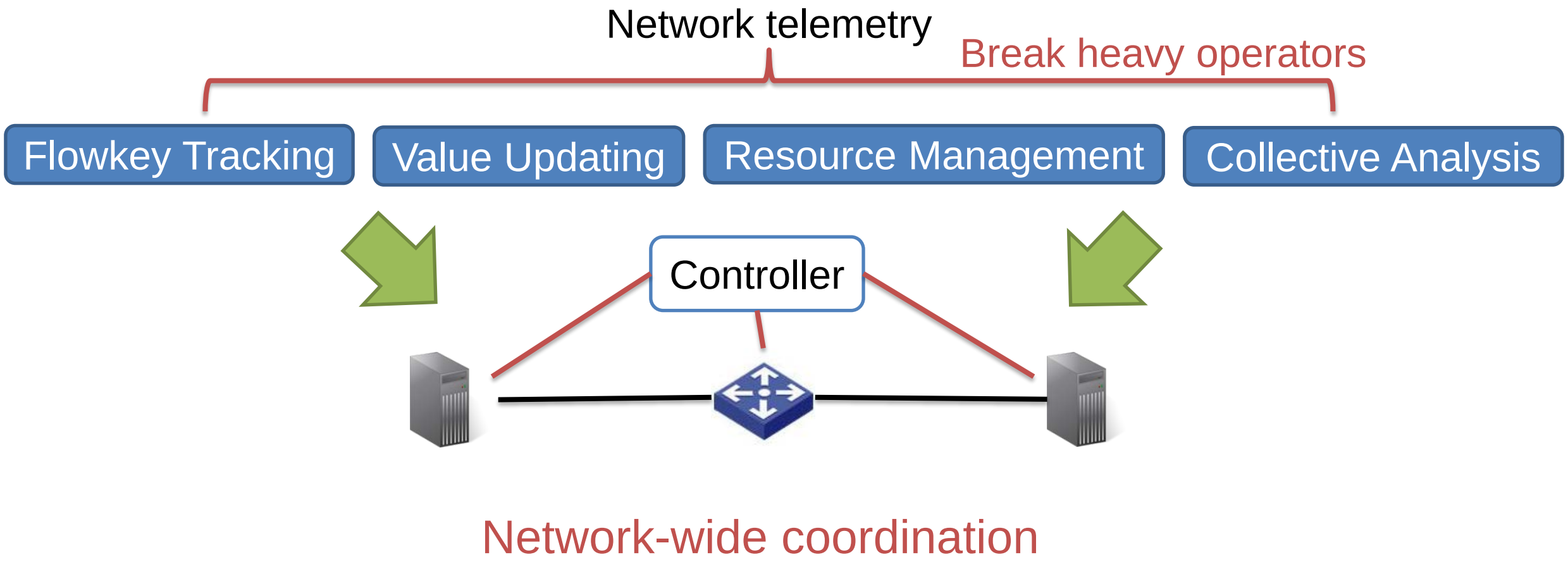
Question 1

Coordinate different entities for network telemetry?

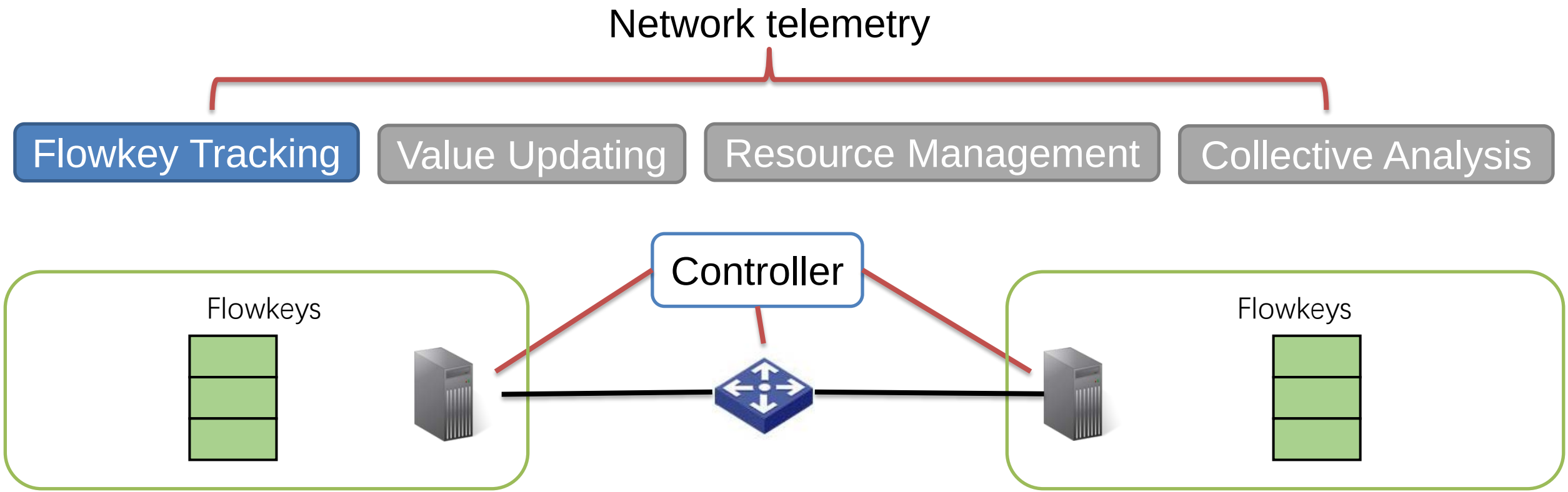
Question 2

Reliable guarantees for the coordination?

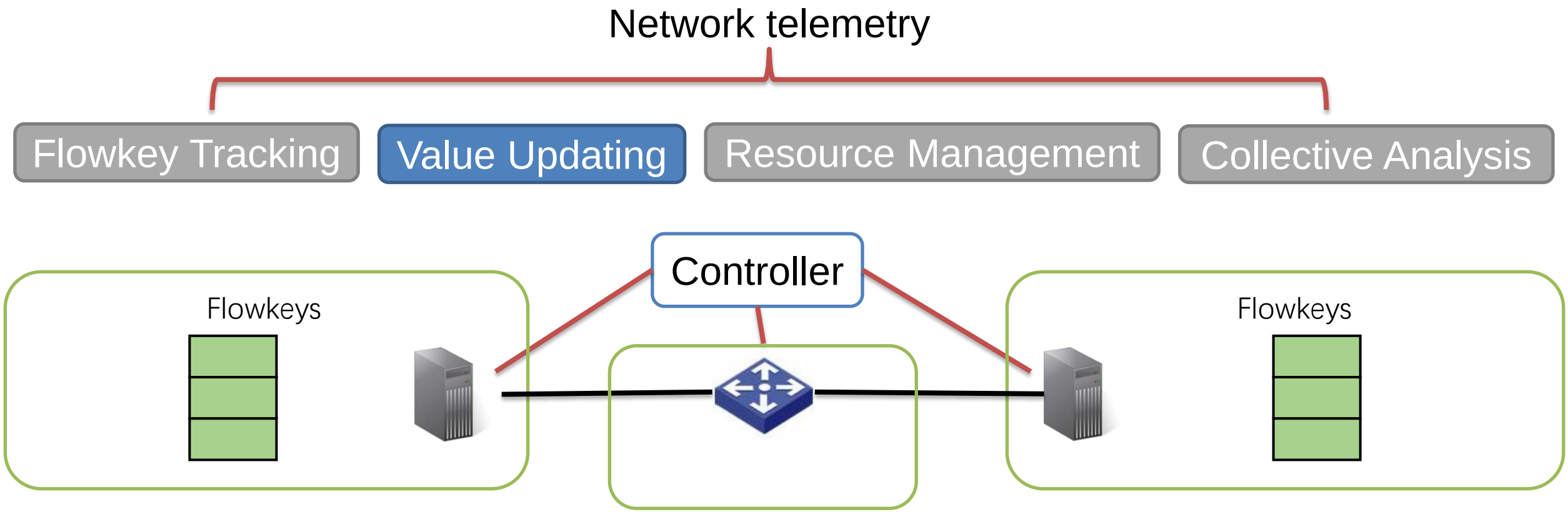
Split-and-Merge Architecture



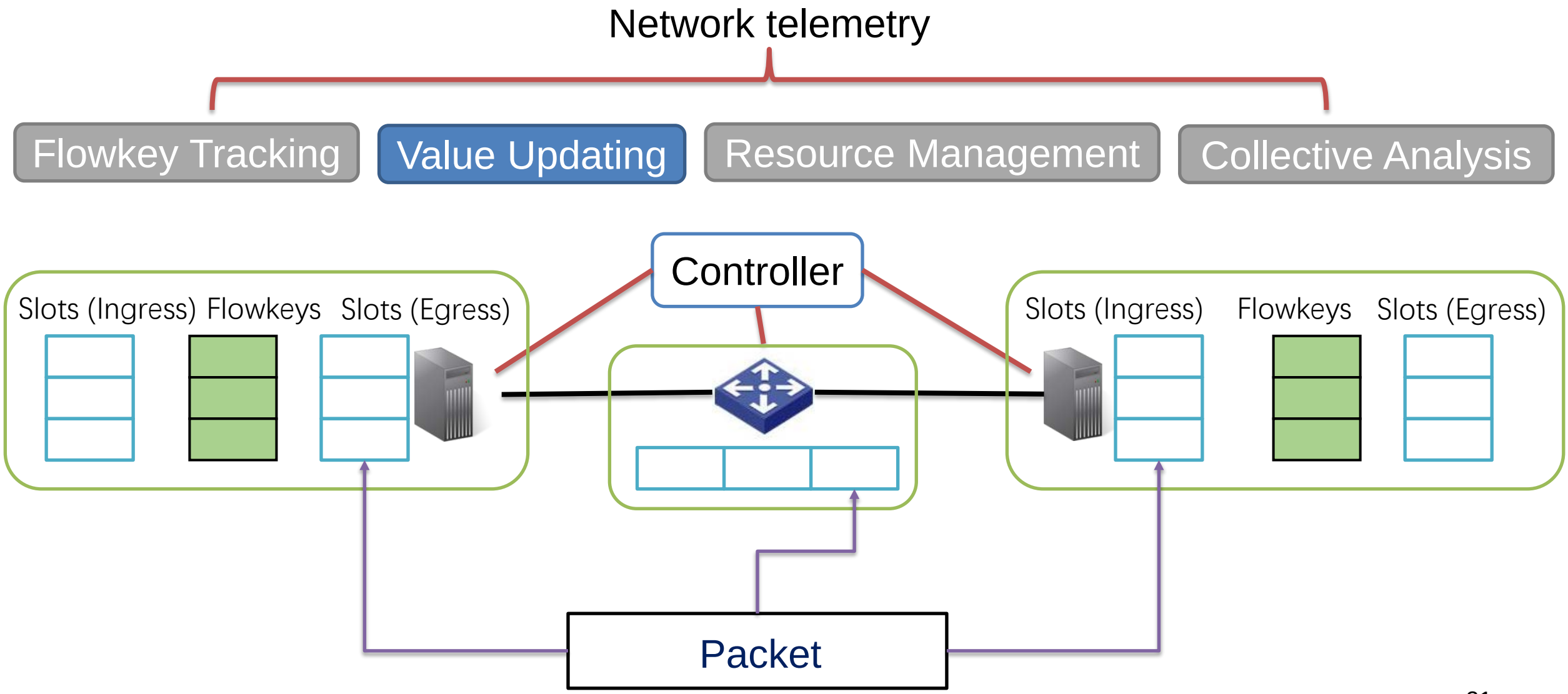
Flowkey Tracking



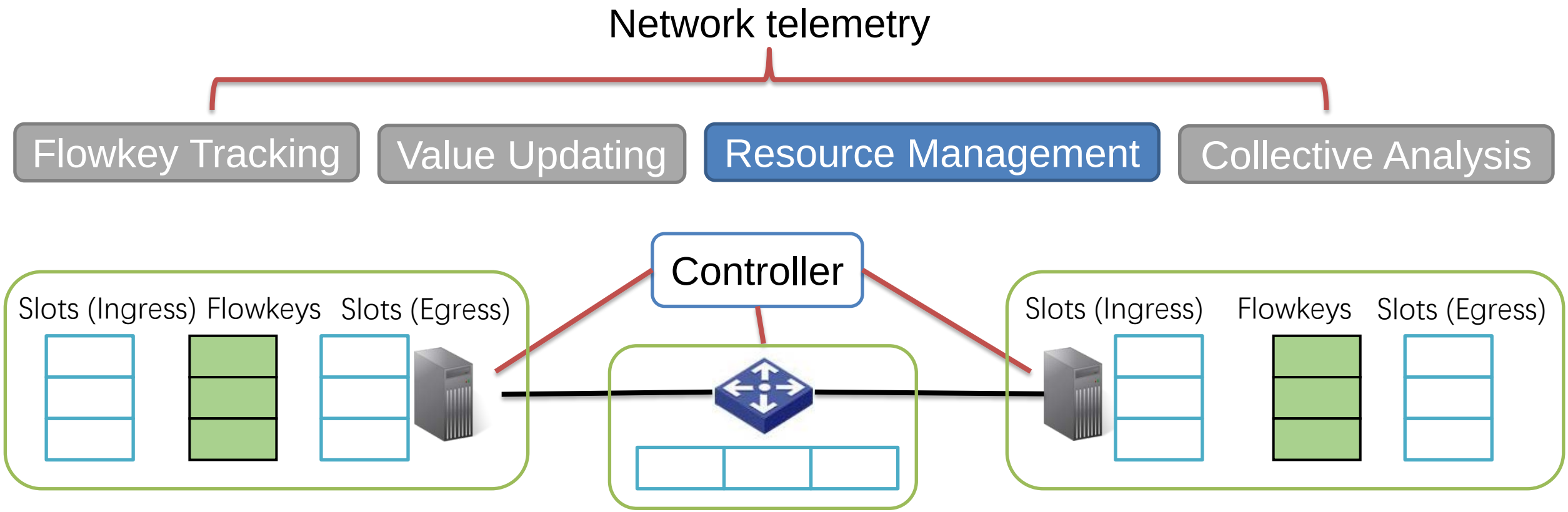
Value Update



Value Update

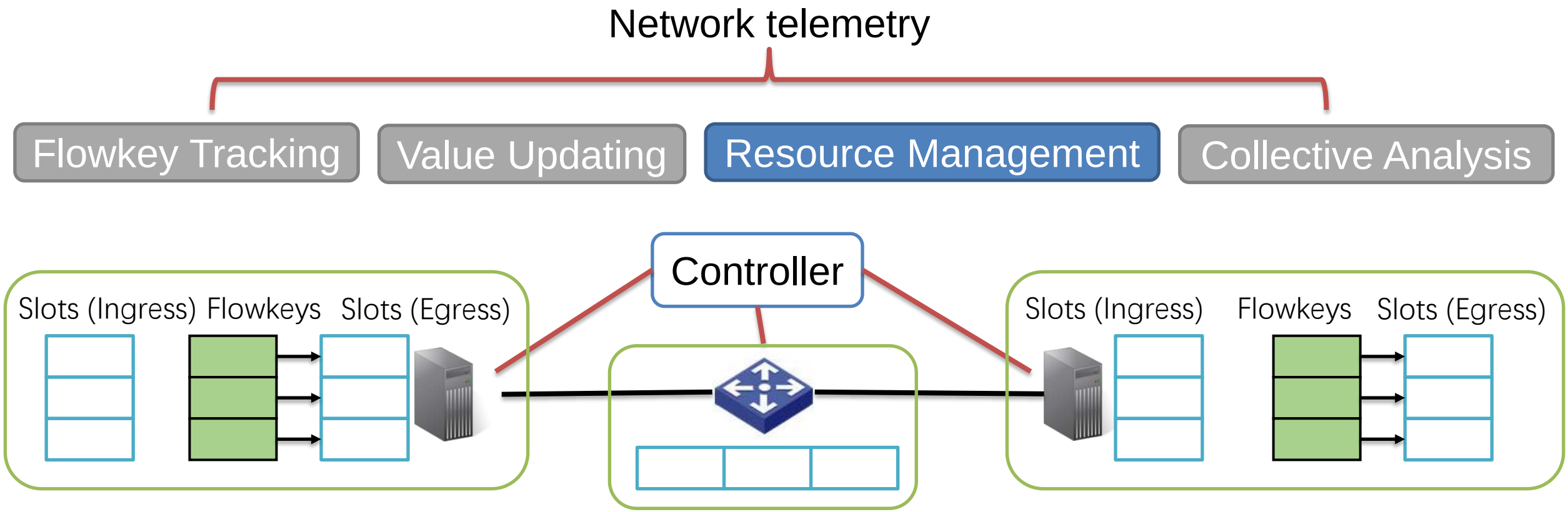


Mapping at End-Host

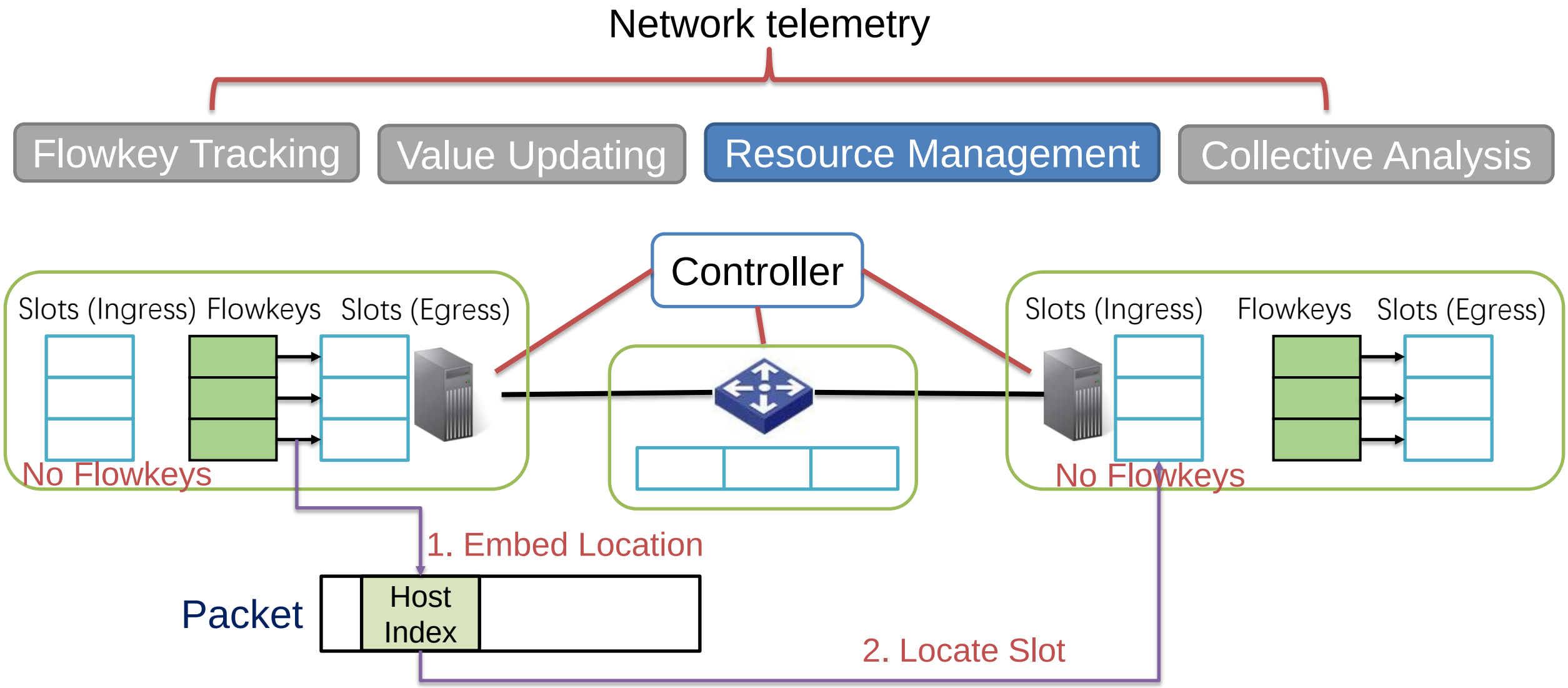


Different strategies of slot maping in end-hosts and switches

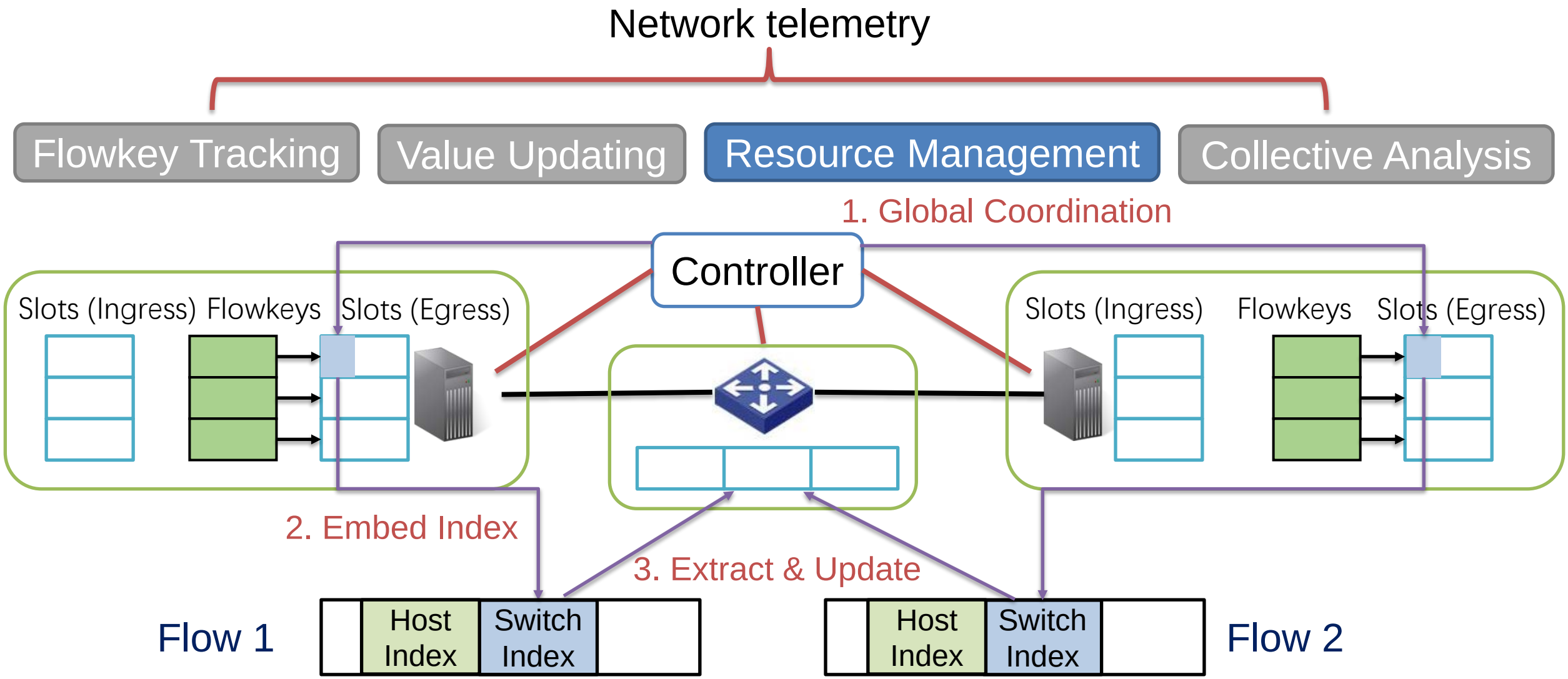
Mapping at End-Host (Egress)



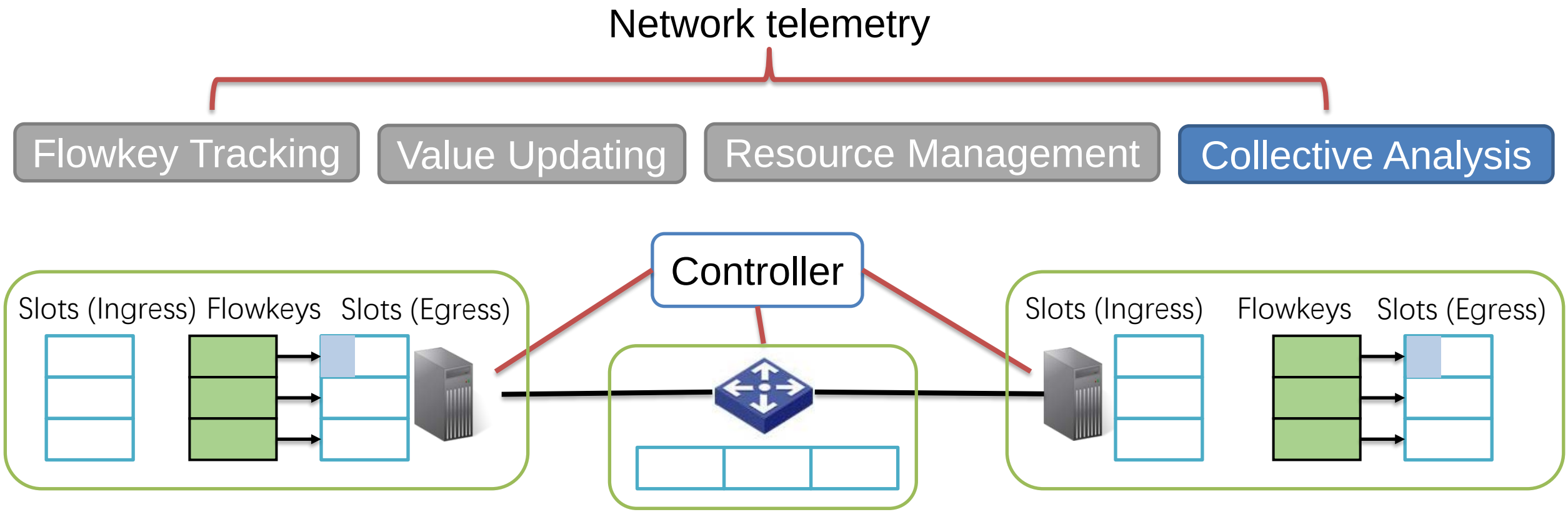
Mapping at End-Host (Ingress)



Mapping at Switch



Collective Analysis

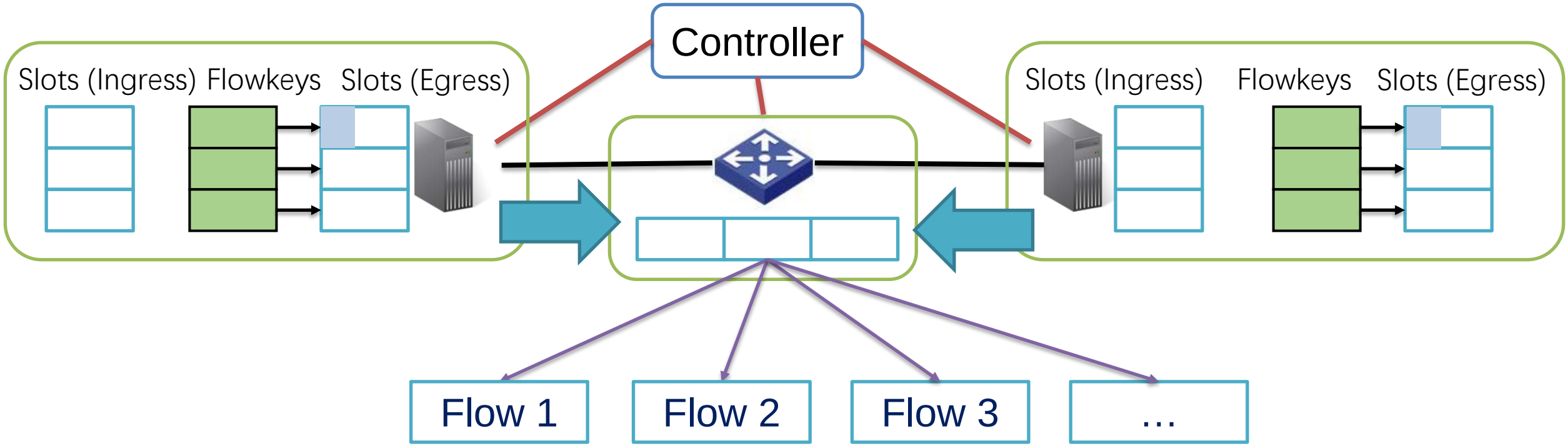


Collect results from end-host and switches to form final flow statistics

Collective Analysis

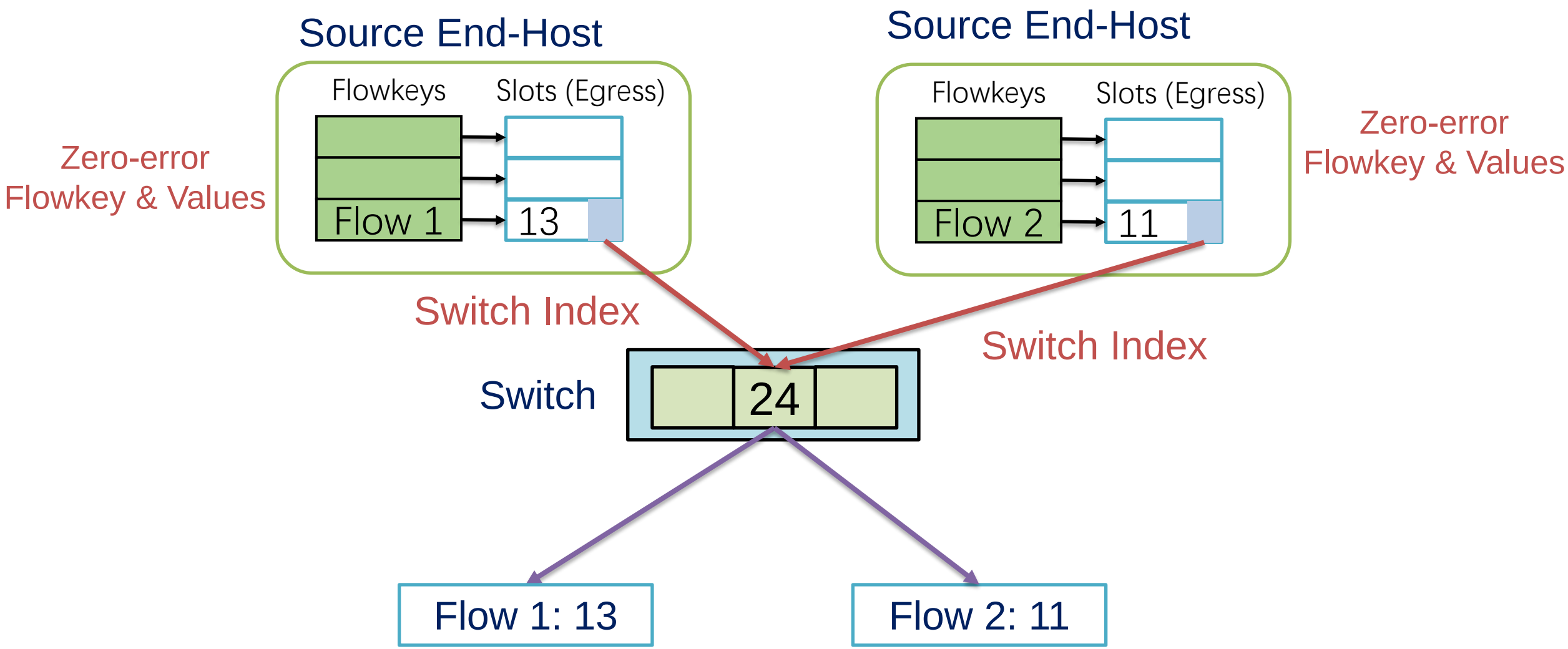
Network telemetry

Flowkey Tracking | Value Updating | Resource Management | Collective Analysis



Exploit end-host information to decompose switch slots

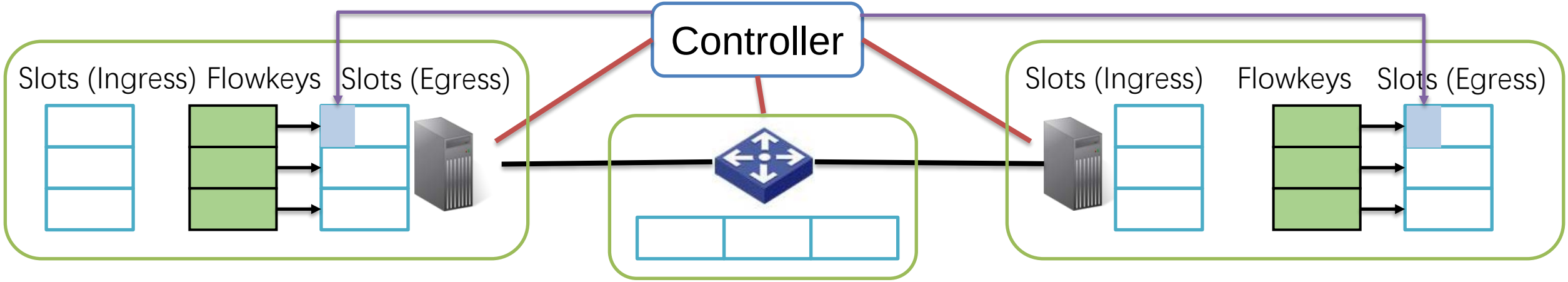
Collective Analysis (Detail)



Putting It Together

Network telemetry

Flowkey Tracking | Value Updating | Resource Management | Collective Analysis



End-Hosts: Hash Table

Exact Per-flow Tracking ☺

Affordable Operations ☺

Switches: Shared Slots

Low Memory Usage ☺

Simple Updates ☺

Controller: Global Info.

Switch Index Mapping ☺

Collective Analysis ☺

OminMon

Re-architect network telemetry by distributed design

Question 1

Coordinate different entities for network telemetry?

Question 2

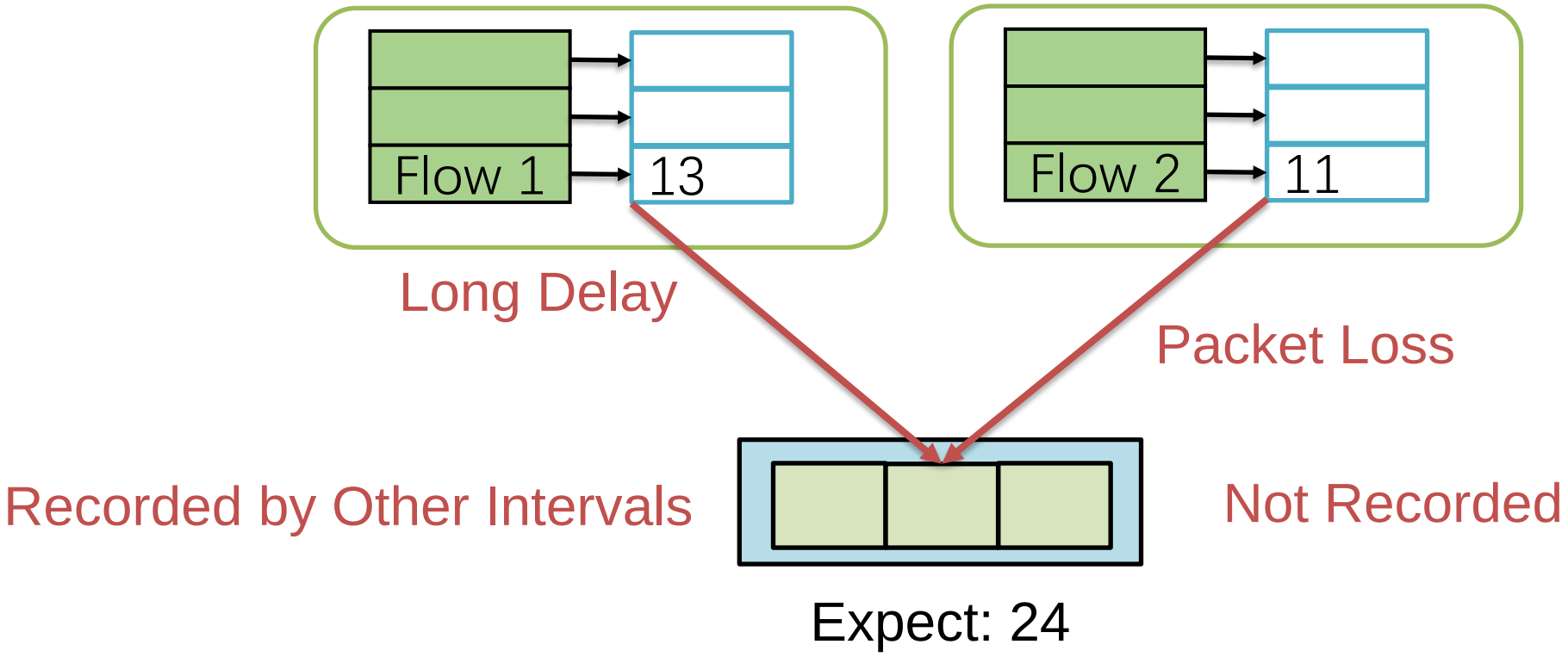
Reliable guarantees for the coordination?

Unreliable Events

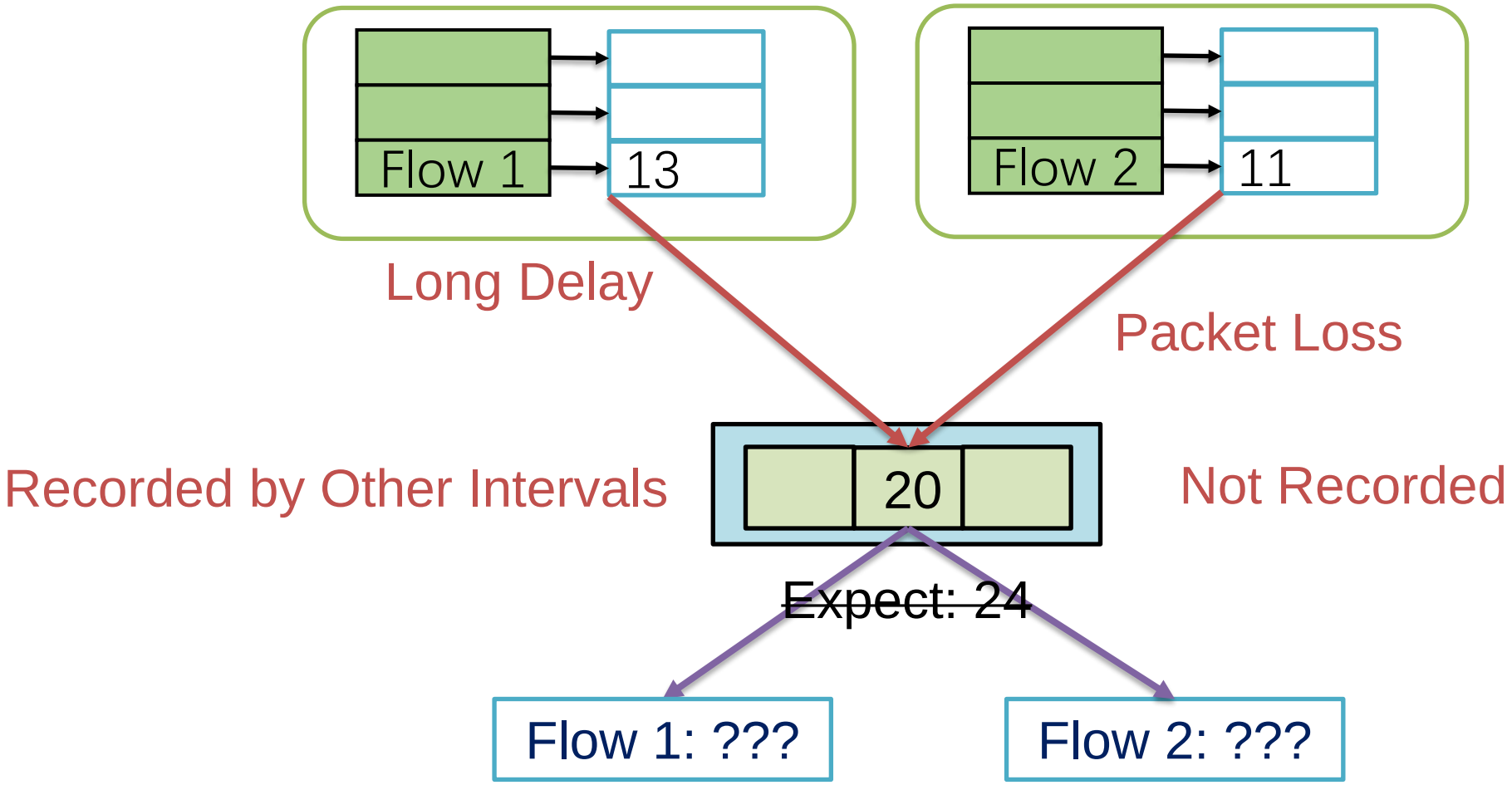
- Lack of global clock
  - Devices reside in different intervals

- Packet loss
  - Flow values are missing in some devices

Impact of Unreliable Events



Impact of Unreliable Events



Reliability Guarantees

Lack of global clock

- Hybrid consistency model
  - Network-wide synchronization
- Guarantees
  - Each packet is included in the same intervals by all devices
  - All end-hosts reside in the same time intervals in most time

Packet Loss

- Loss inference
  - Linear system with DCN-specific optimizations
  - Flow mapping algorithm
- Guarantees
  - Per-switch, per-flow loss interference in common cases

More details in the paper

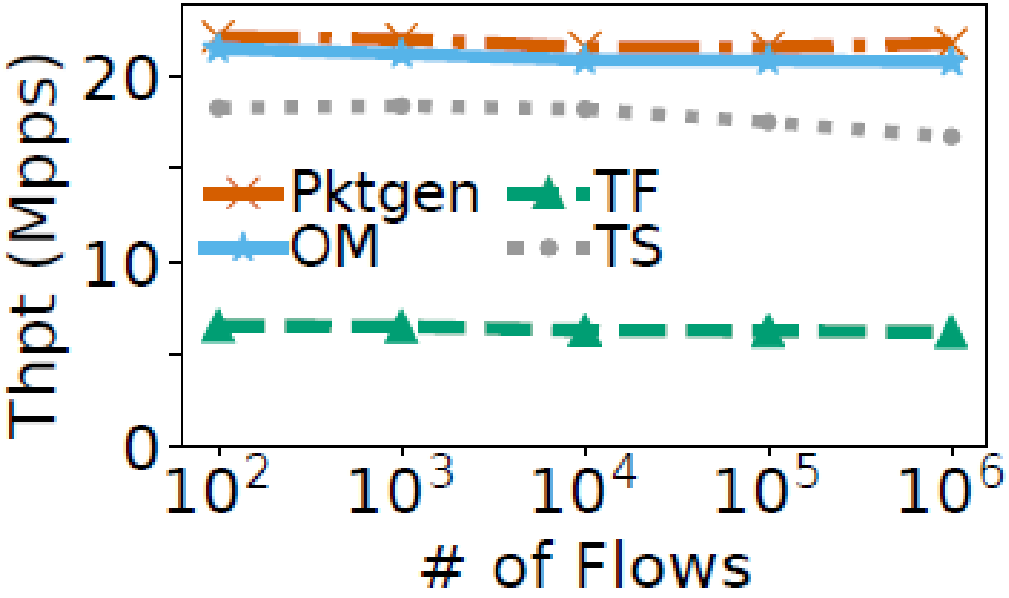
Implementation

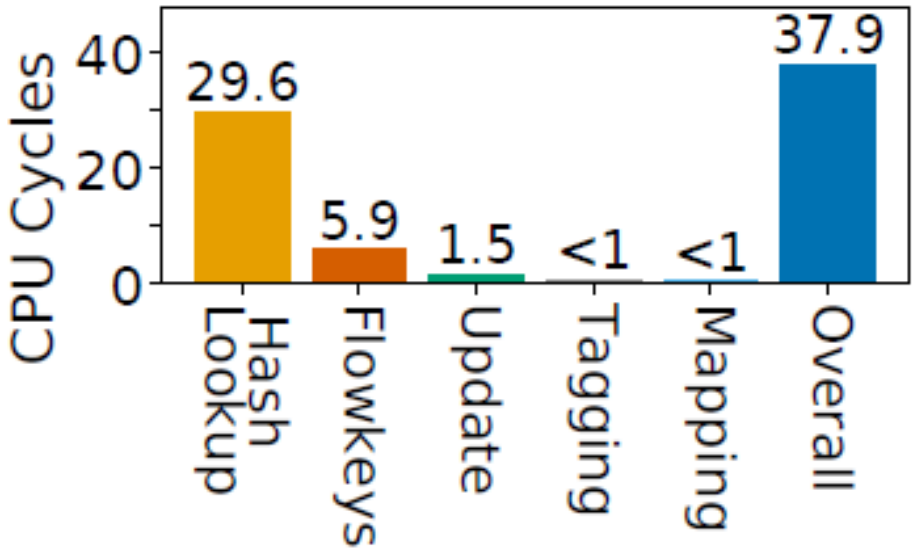
- Testbed
  - End-hosts: DPDK
  - Switch: P4
  - Controller: C++

- Simulator: 8-ray fat-tree

Host Overheads

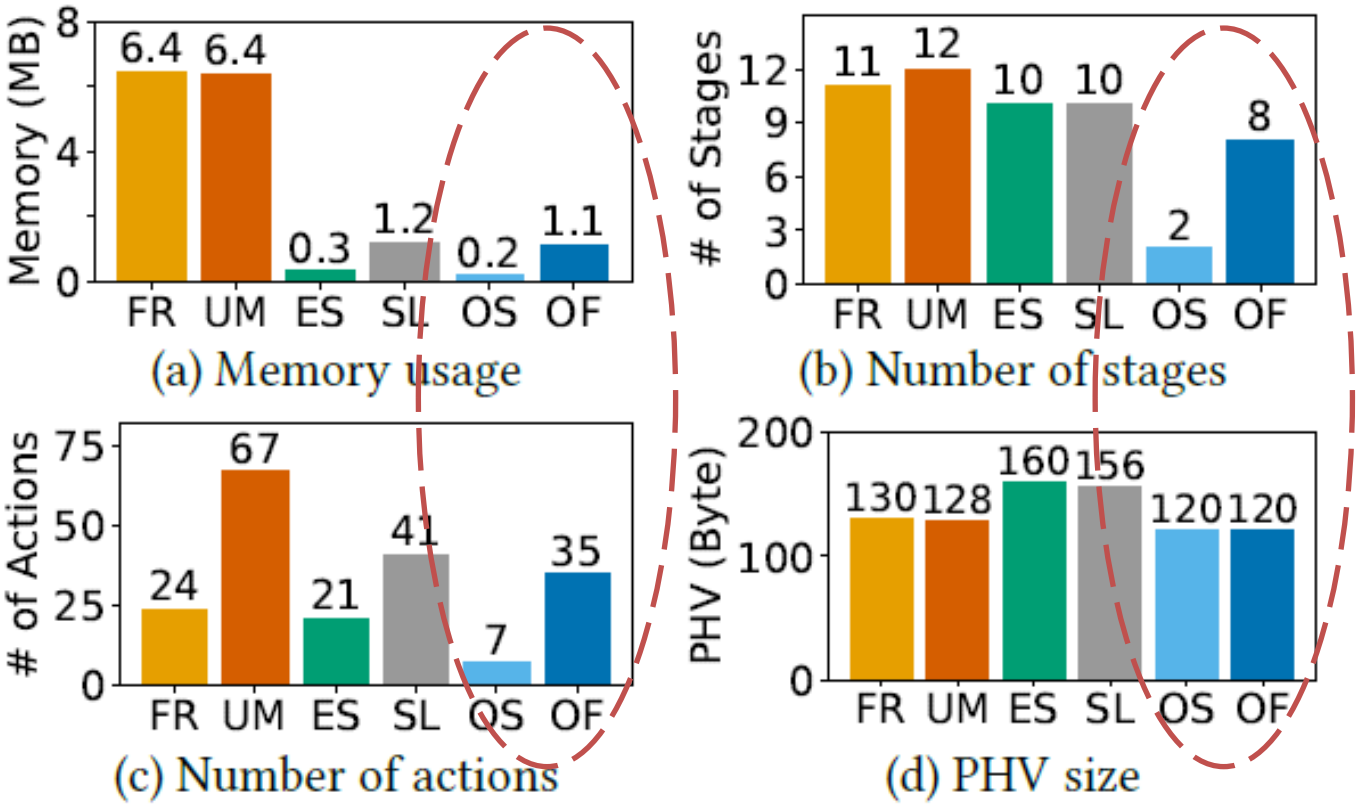
- <10% overheads when adding telemetry functionalities to PktGen
- Hash lookup dominates the overheads





Switch Overheads

- Less resources than sketch-based techniques
- Only OmniMon achieves zero errors



Sketch techniques

FR: FlowRadar (NSDI 16)
UM: UmniMon (SIGCOMM 16)
ES: Elastic Sketch (SIGCOMM 18)
SL: SketchLearn (SIGCOMM 18)
Each sketch only monitors packet count

OmniMon

OS: OmniMon that monitors only packet count
OF: OmniMon that monitors 9 statistics

More Results

- Controller overheads
- Synchronization efficiency
- Accountability
- Scalability
- User case: anomaly detection
- Use case: network failure diagnosis
- Use case: load balance evaluation

Conclusion

- OmniMon architecture: split-and-merge design
  - Four partial operations
  - Network-wide coordination
- Consistency guarantee
  - Network-wide synchronization with hybrid consistency model
- Accountability guarantee
  - Packet loss inference with linear systems
  - Flow mapping algorithm
- Prototype: DPDK + P4
- Results: compare with 11 state-of-the-art solutions in various aspects

Source Code Available: https://github.com/N2-Sys/Omnimon