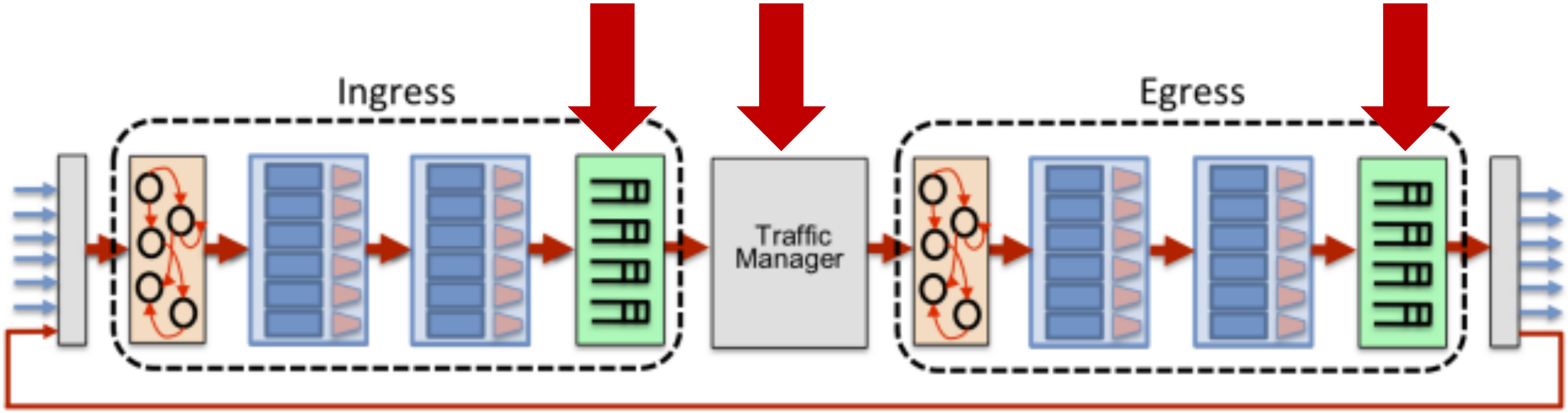
Programmable and Universal Packet Scheduling

ECE/CS598HPN

Radhika Mittal

Scheduling not programmable



Scheduling in standard switches

- Fixed number (8-16) of priority queues.
- Can be configured to support weighted round-robin.

- In practice, few priority levels might be left for common use.

- Not enough to support diverse scheduling policies:
  - Fairness across many more flows.
  - Shortest job first scheduling.
  - …..

Two complementary papers

- Programmable packet scheduling, HotNets'15, SIGCOMM'16
- Universal packet scheduling, HotNets'15, NSDI'16

Two complementary papers

- Programmable packet scheduling, HotNets'15, SIGCOMM'16
- Universal packet scheduling, HotNets'15, NSDI'16

Two complementary papers

- Programmable packet scheduling, HotNets'15, SIGCOMM'16
  - Many slides borrowed from Anirudh Sivaraman.
- Universal packet scheduling, HotNets'15, NSDI'16

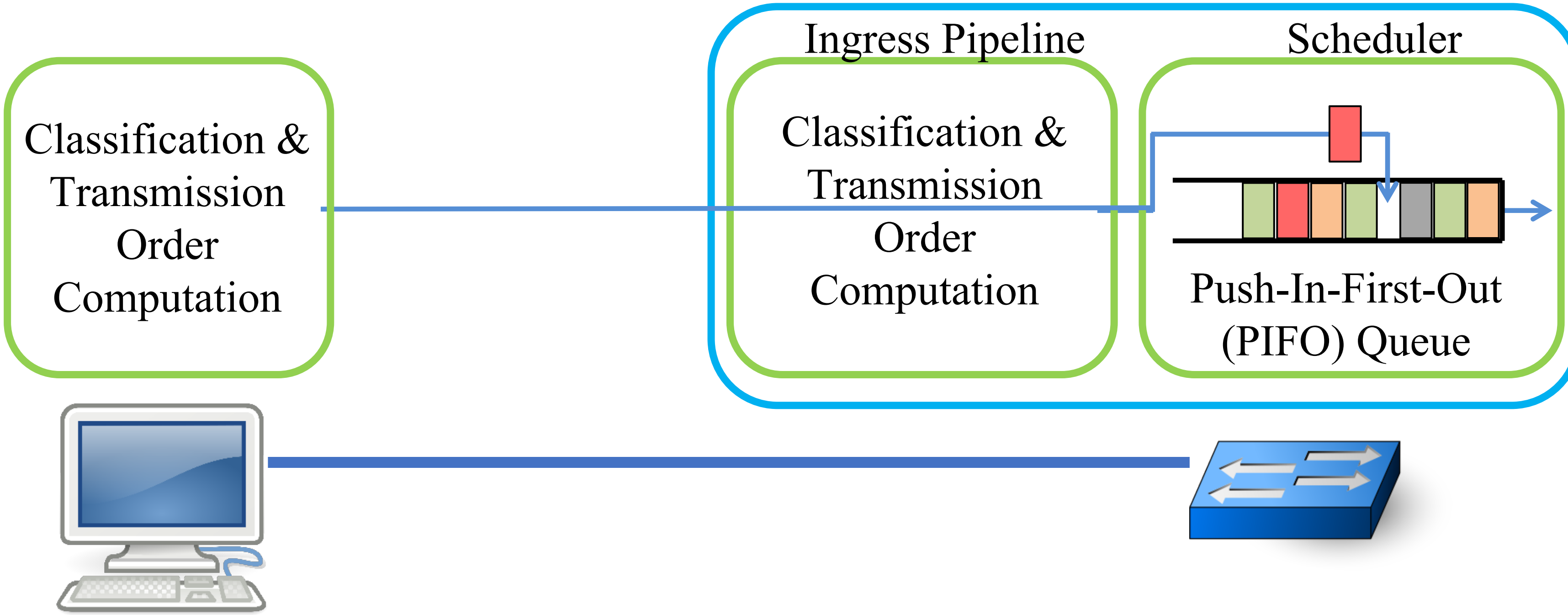
The Push-In First-Out Queue

- Many algorithms determine transmission order at packet arrival
- Often, relative order of packet transmissions doesn't change with future arrivals
- Examples:
  - SJF: Order determined by flow size
  - FCFS: Order determined by arrival time
- Push-in first-out queues (PIFO) is a good abstraction to capture such algorithms.
  - packets are pushed into an arbitrary location based on a priority, and dequeued from the head
- First used as a proof construct by Chuang et. al.

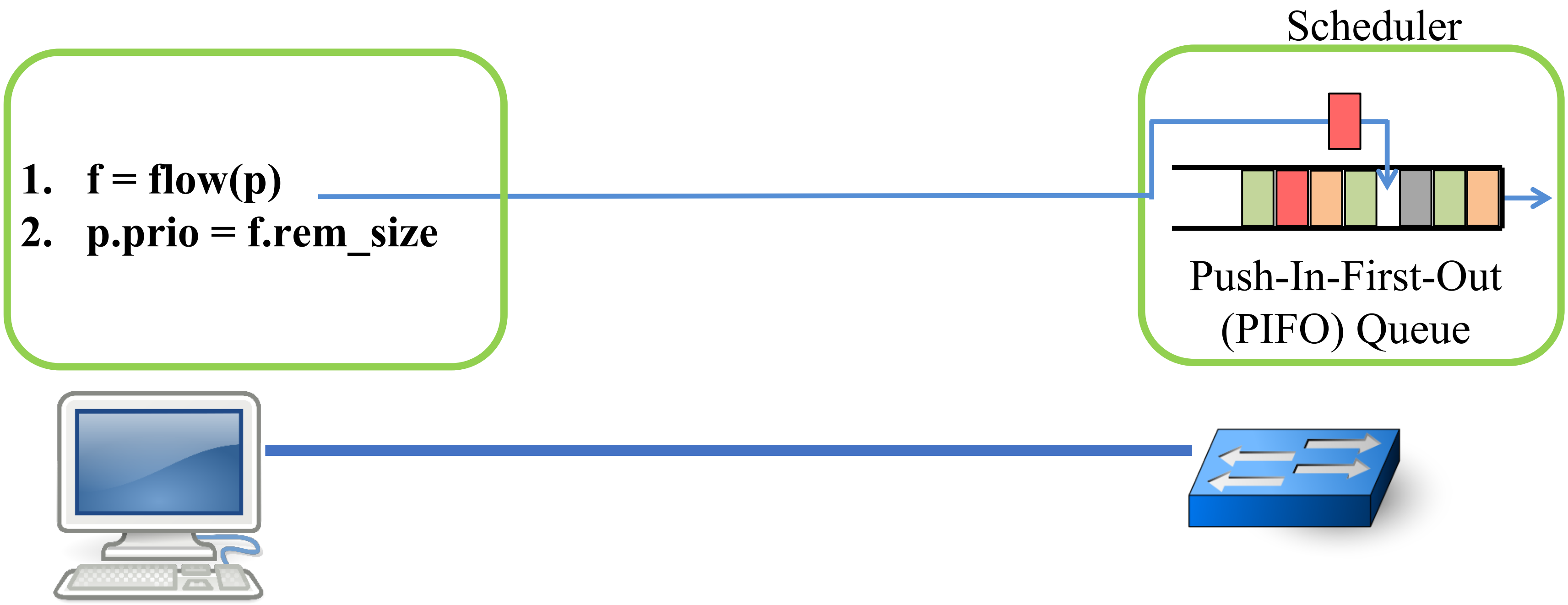
The PIFO abstraction

- PIFO: A sorted array that let us insert an entry (packet or PIFO pointer) based on a programmable priority
  - Entries are always dequeued from the head
  - If an entry is a packet, dequeue and transmit it
  - If an entry is a PIFO, dequeue it, and continue recursively

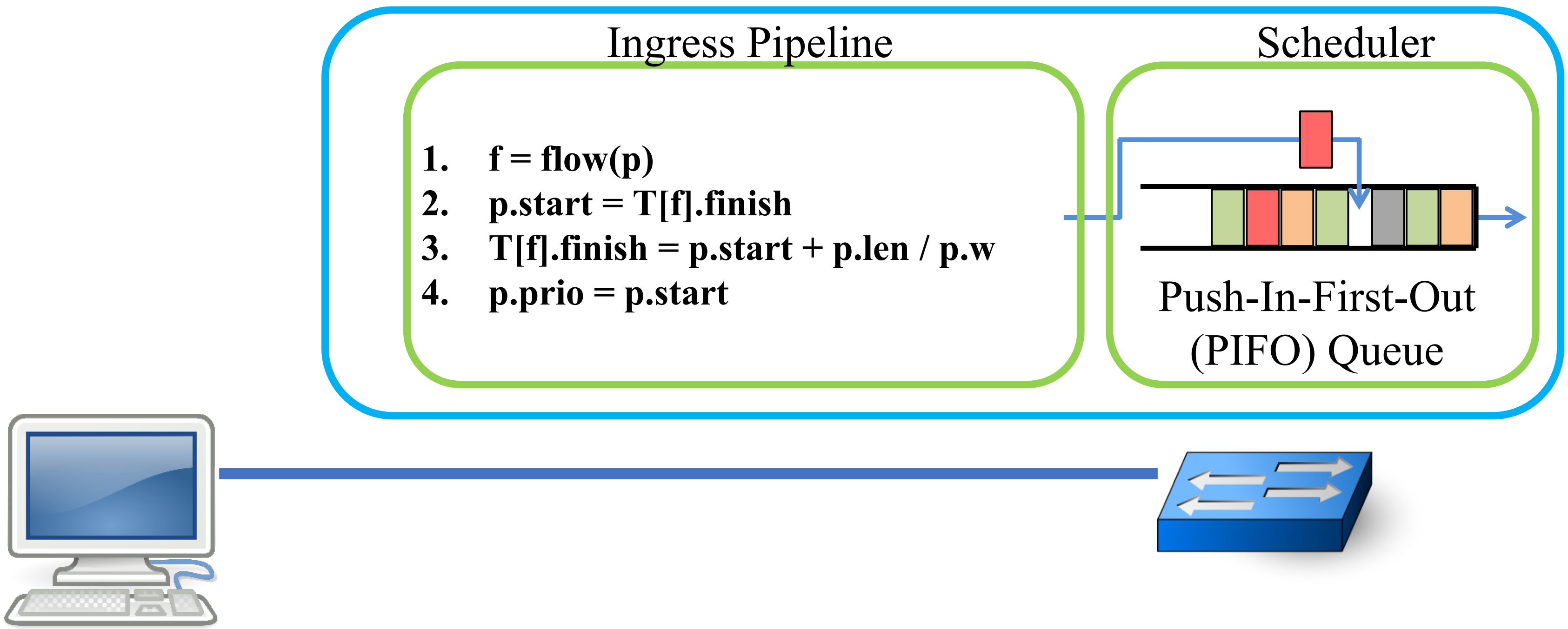
A programmable scheduler



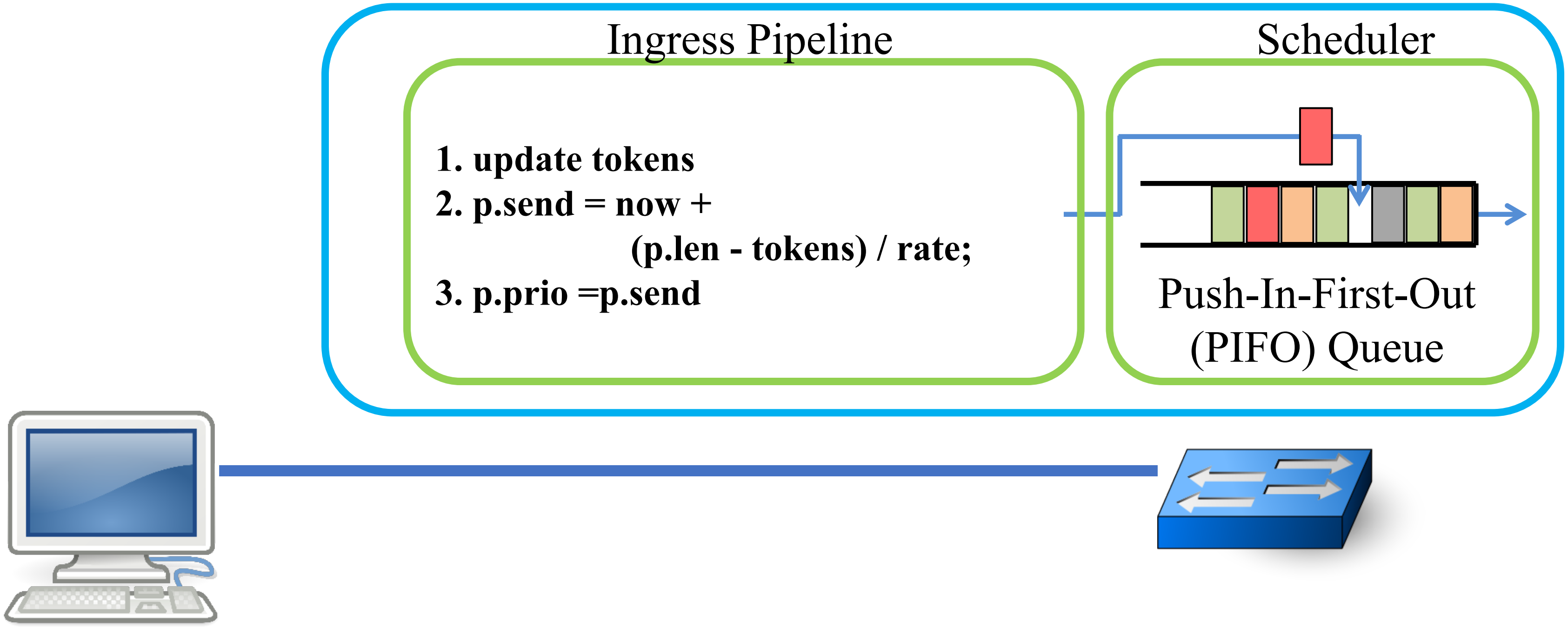
pFabric using PIFO



Weighted Fair Queuing

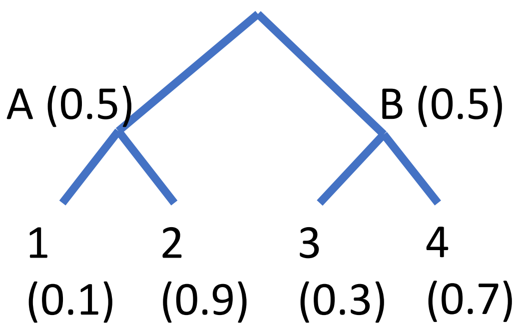


Traffic Shaping

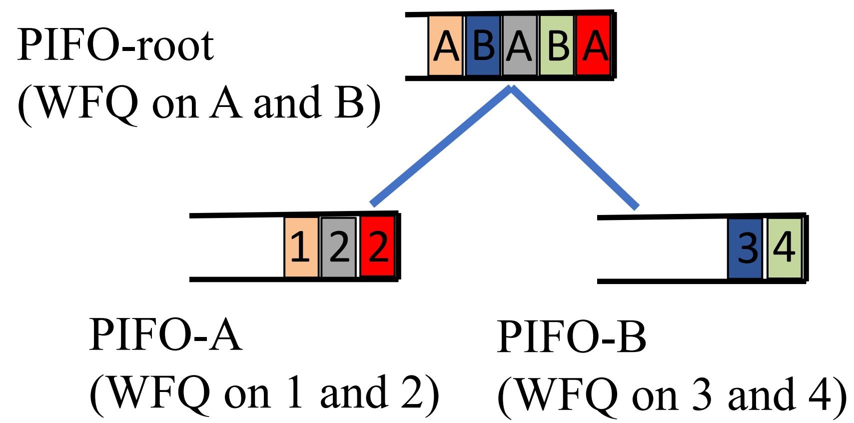


Composing PIFOs

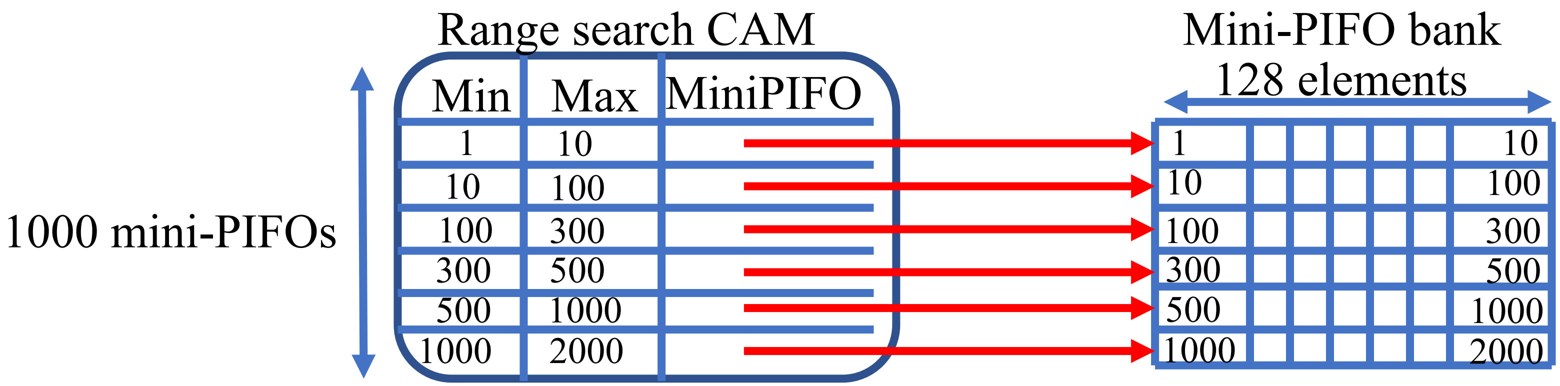
Hierarchical packet-fair queueing (HPFQ)



Composing PIFOs

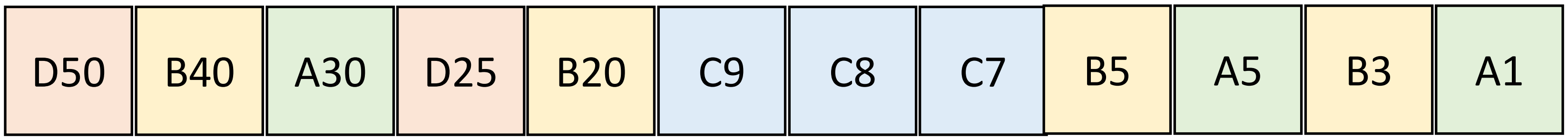


PIFO in hardware

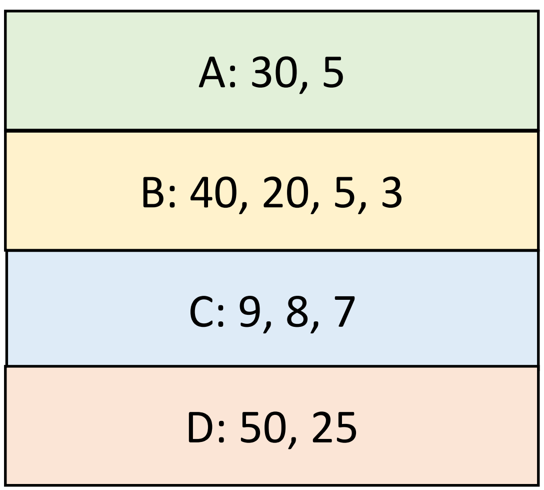


- Meets timing at 1 GHz on a 16 nm node
- 5 % area overhead for 3-level hierarchy

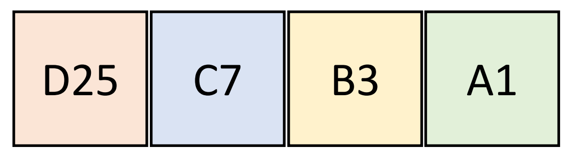
Programmable packet scheduling, SIGCOMM'16



Single array PIFO can be expensive
(lots of comparator circuits required)



Rank Store



Flow scheduler
(fewer comparator circuits required)

Key limitation of the PIFO abstraction?

- What if packet priorities change after enqueuing?
- Need to re-program routers for supporting a new/different algorithm…

Two complementary papers

- Programmable packet scheduling, HotNets'15, SIGCOMM'16
- Universal Packet Scheduling, HotNets'15, NSDI'16

Many Scheduling Algorithms

- **Many different algorithms**
  - FIFO, FQ, virtual clocks, priorities…
- **Many different goals**
  - fairness, small packet delay, small FCT…
- **Many different contexts**
  - WAN, datacenters, cellular…

Many Scheduling Algorithms

- Implemented in *router hardware.*
- *How do we support different scheduling algorithms for different requirements?*
  - Option 1: Change router hardware for each new algorithm
  - Option 2: Implement *all* scheduling algorithms in hardware
  - Option 3: Programmable scheduling hardware

Many Scheduling Algorithms

- Implemented in *router hardware.*
- *How do we support* ***different scheduling algorithms*** *for different requirements?*
    - Option 1: Change router hardware for each new algorithm
    - Option 2: Implement *all* scheduling algorithms in hardware
    - Option 3: Programmable scheduling hardware

Many Scheduling Algorithms

- Implemented in *router hardware.*

- *How do we support different scheduling algorithms for* ***different requirements****?*
  - Option 1: Change router hardware for each new algorithm
  - Option 2: Implement *all* scheduling algorithms in hardware
  - Option 3: Programmable scheduling hardware

We are asking a new question…..

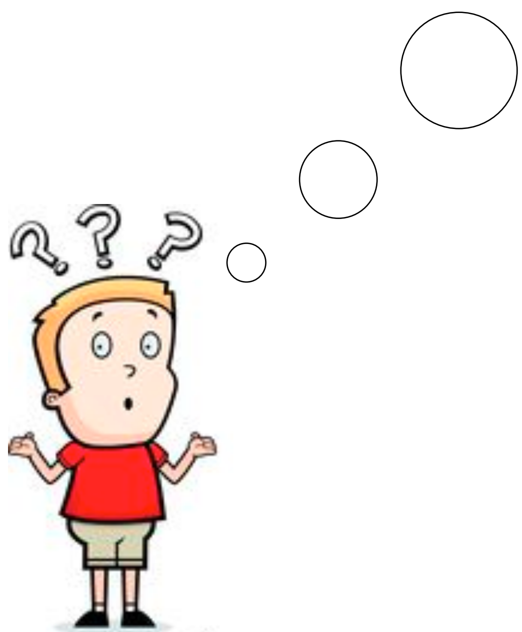
~~*How do we support different scheduling algorithms for different requirements?*~~

Is there a *universal* packet scheduling algorithm?

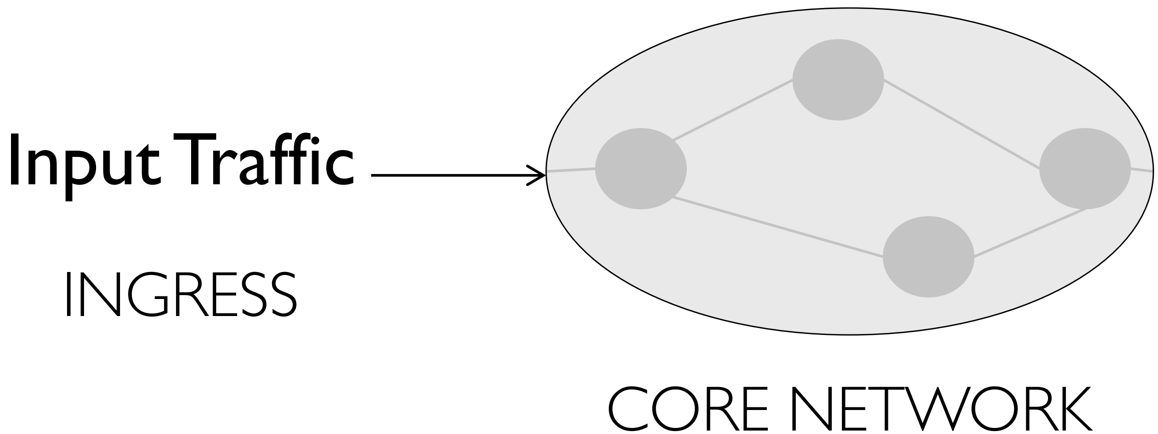
UPS: Universal Packet Scheduling Algorithm

A *single scheduling algorithm* that can imitate the network-wide output produced by ***any*** other algorithm.

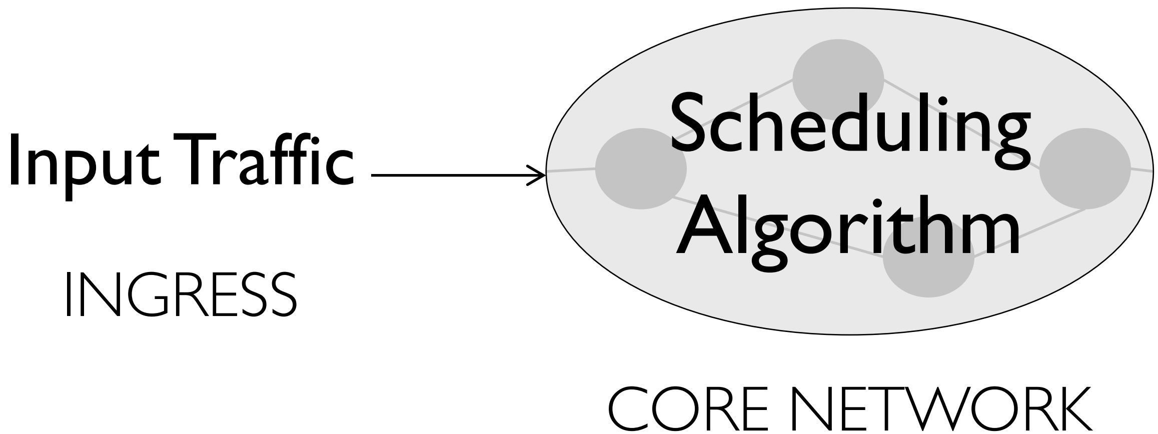
How can a single algorithm imitate all others?



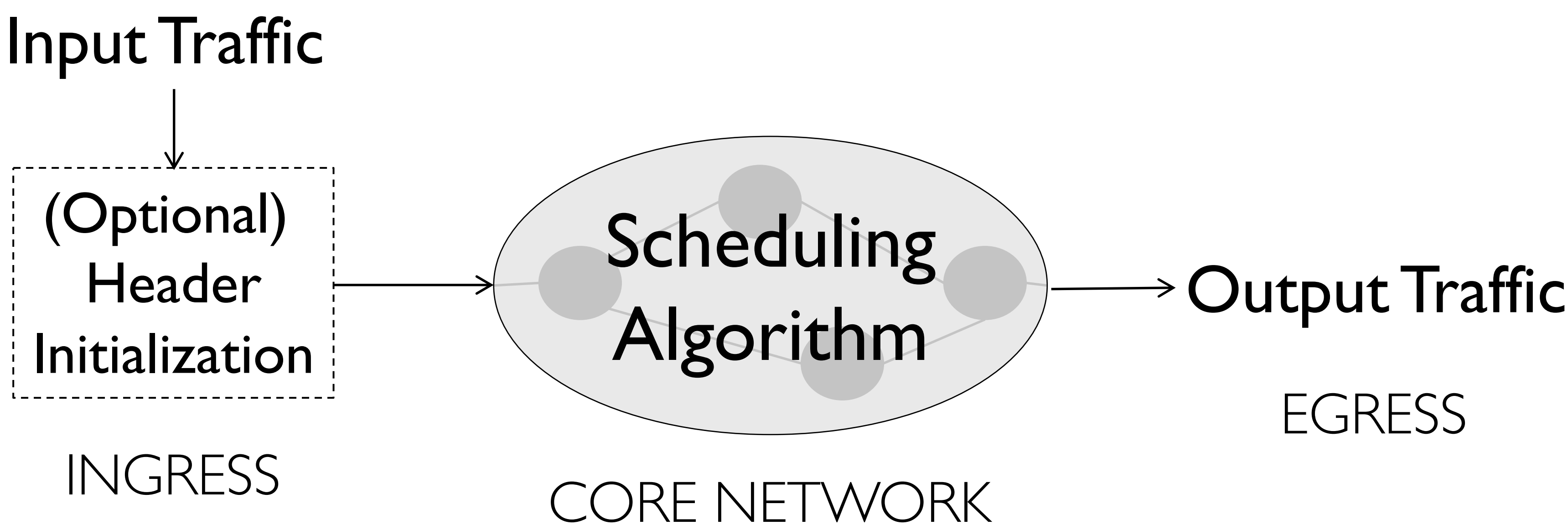
Network Model



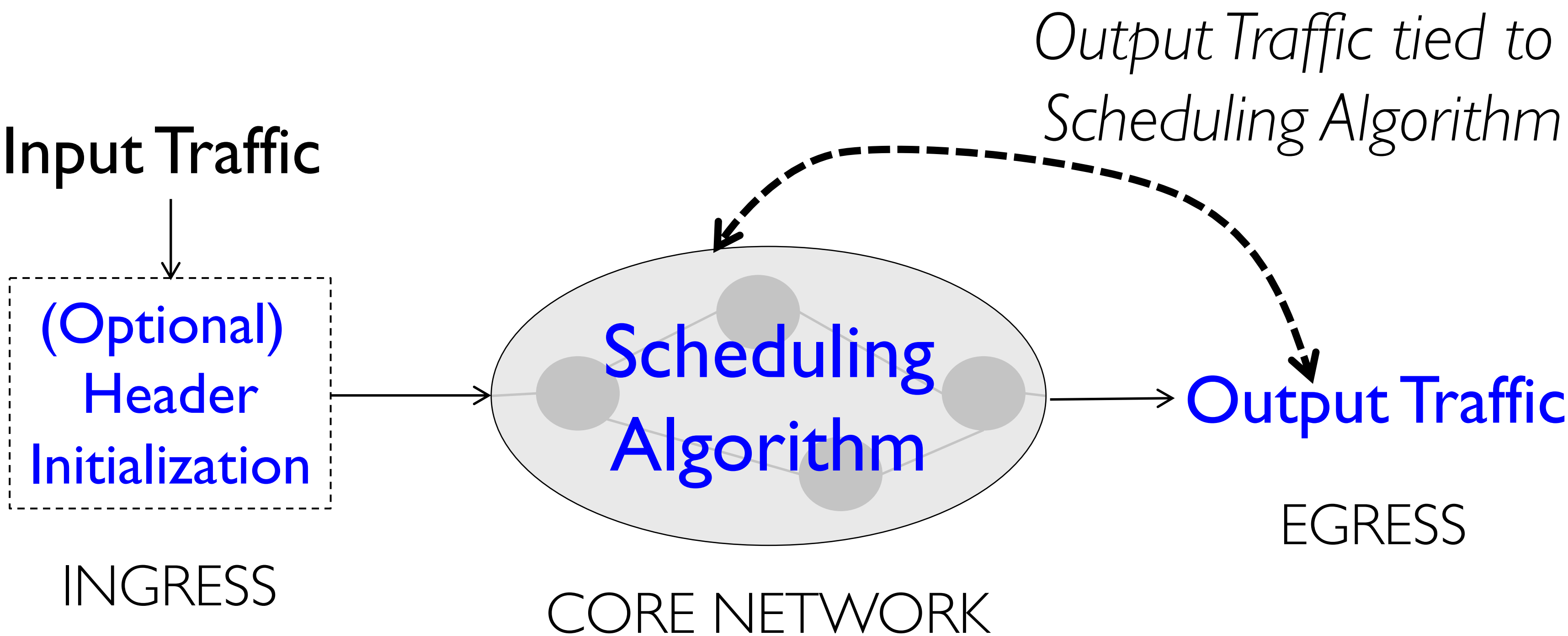
Network Model



Network Model

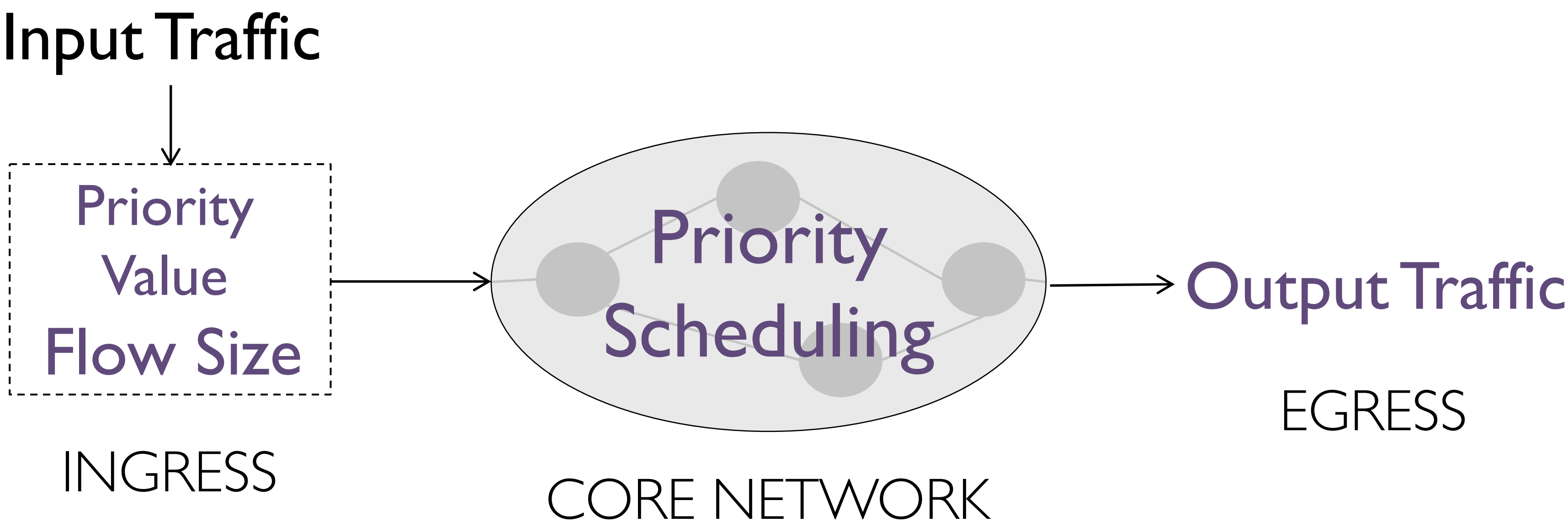


Network Model



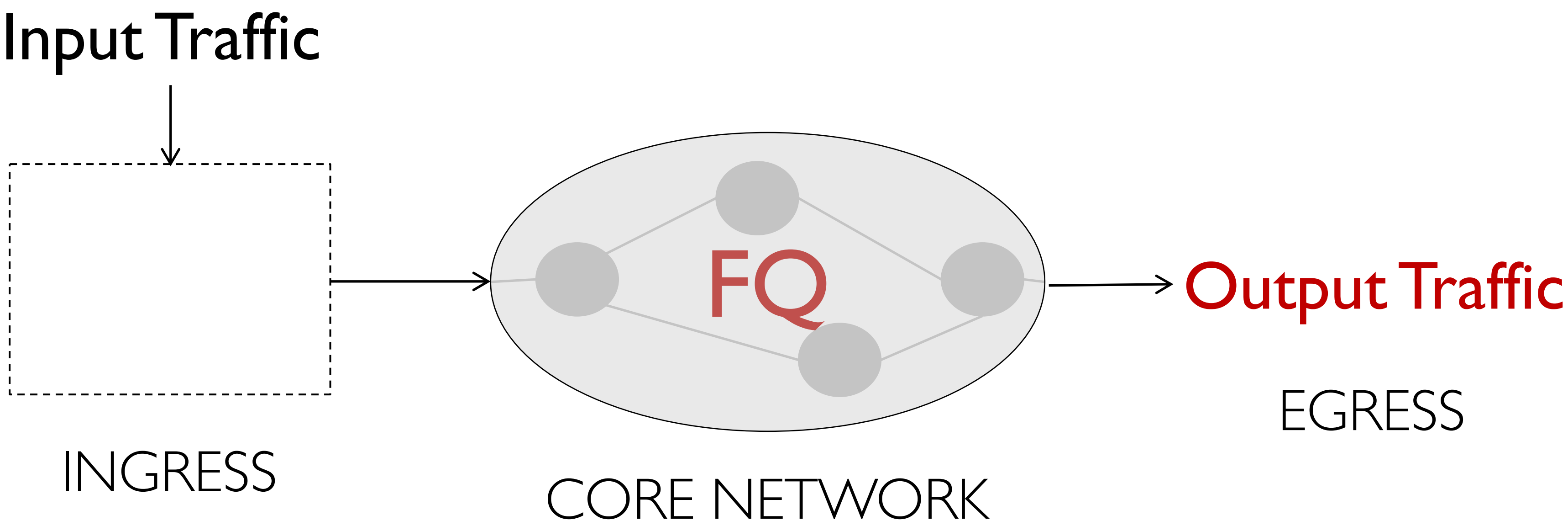
Network Model

Goal: Minimize Mean FCT



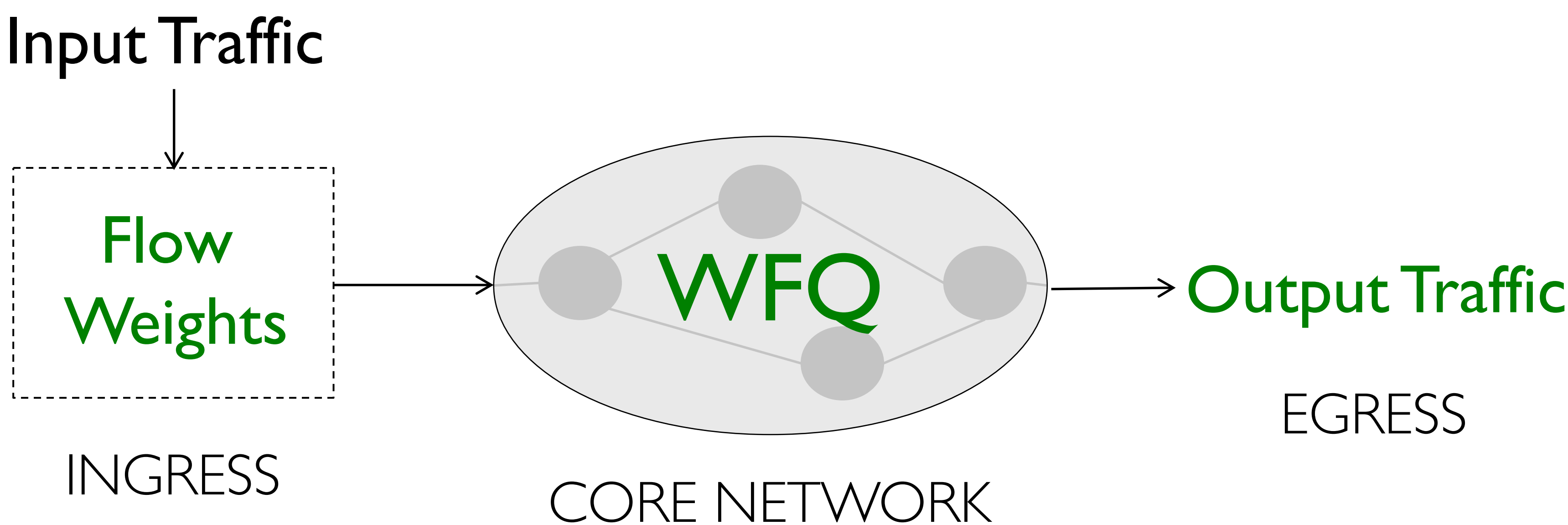
Network Model

Goal: Fairness

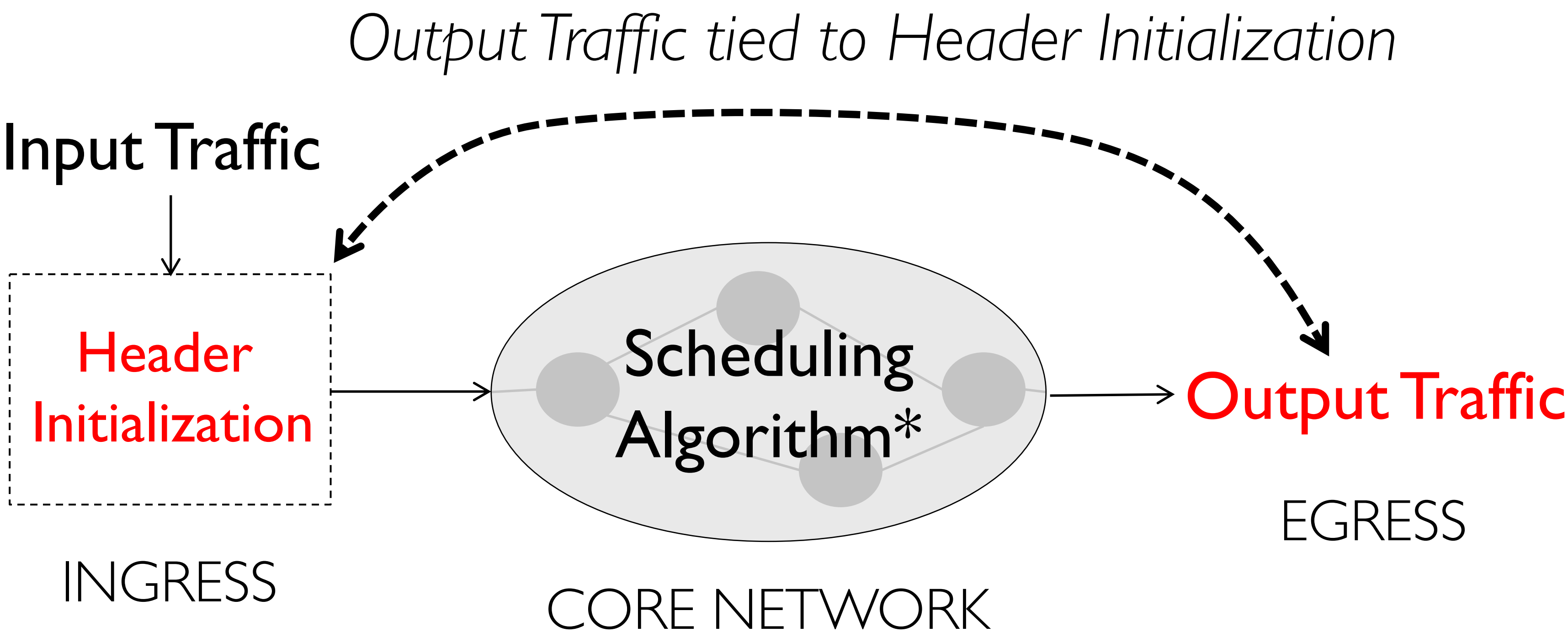


Network Model

Goal: Weighted Fairness

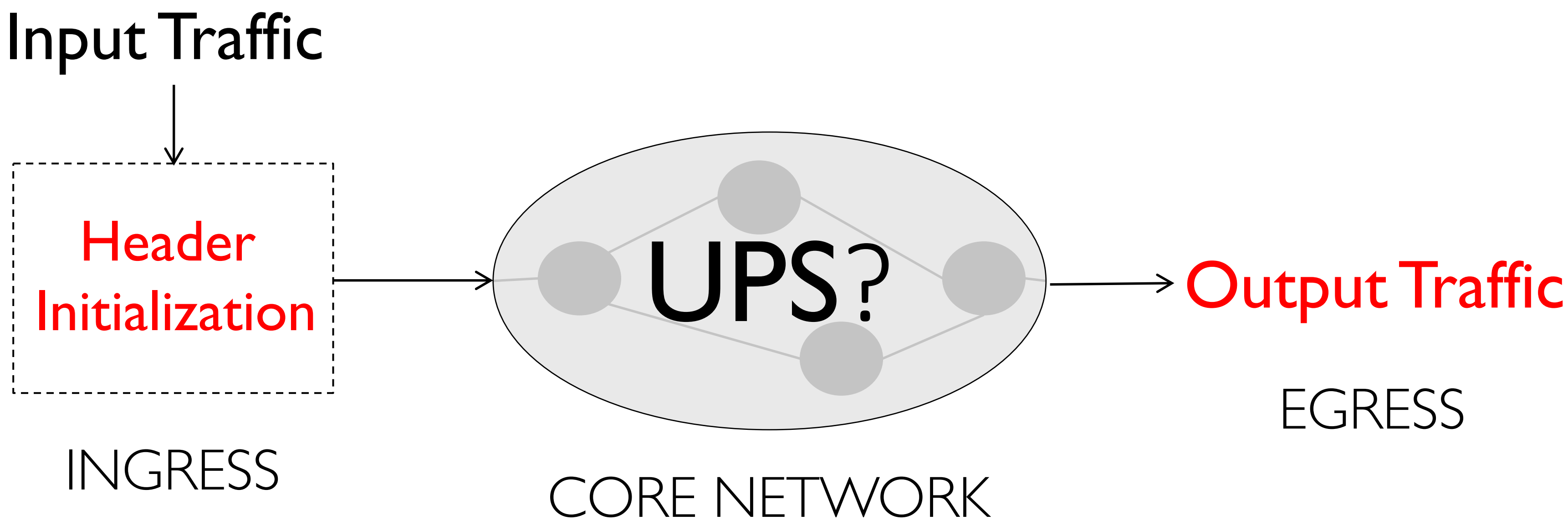


Network Model



* Uses packet header state to make scheduling decisions

Network Model



How do we formally define and evaluate a UPS?

Defining a UPS

Theoretical Viewpoint:

Can it replay a given schedule?

Practical Viewpoint:

Can it achieve a given objective?

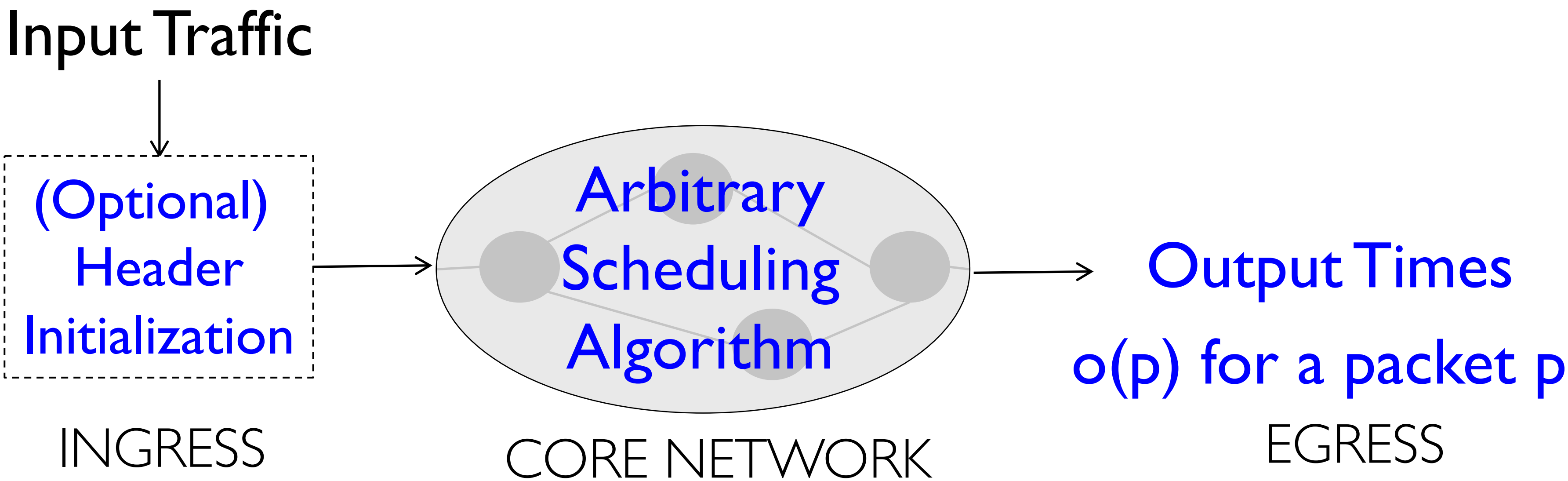
Theoretical Viewpoint

Can it replay a given schedule?

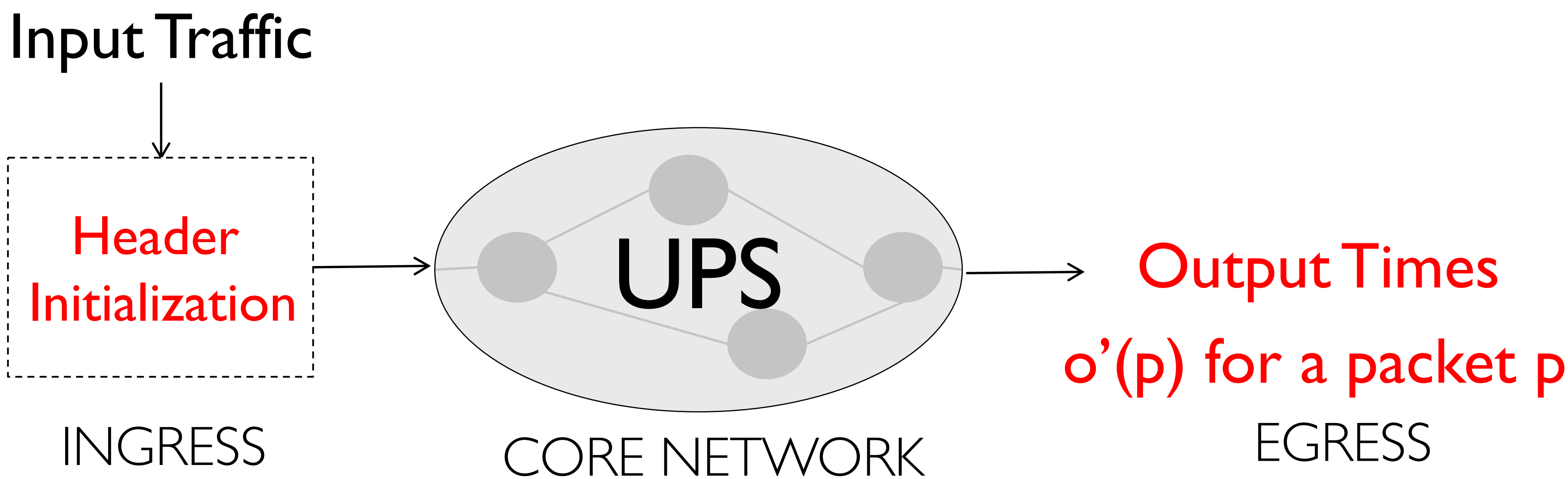
Original Schedule

Only requirement from original schedule:

Output Times are viable

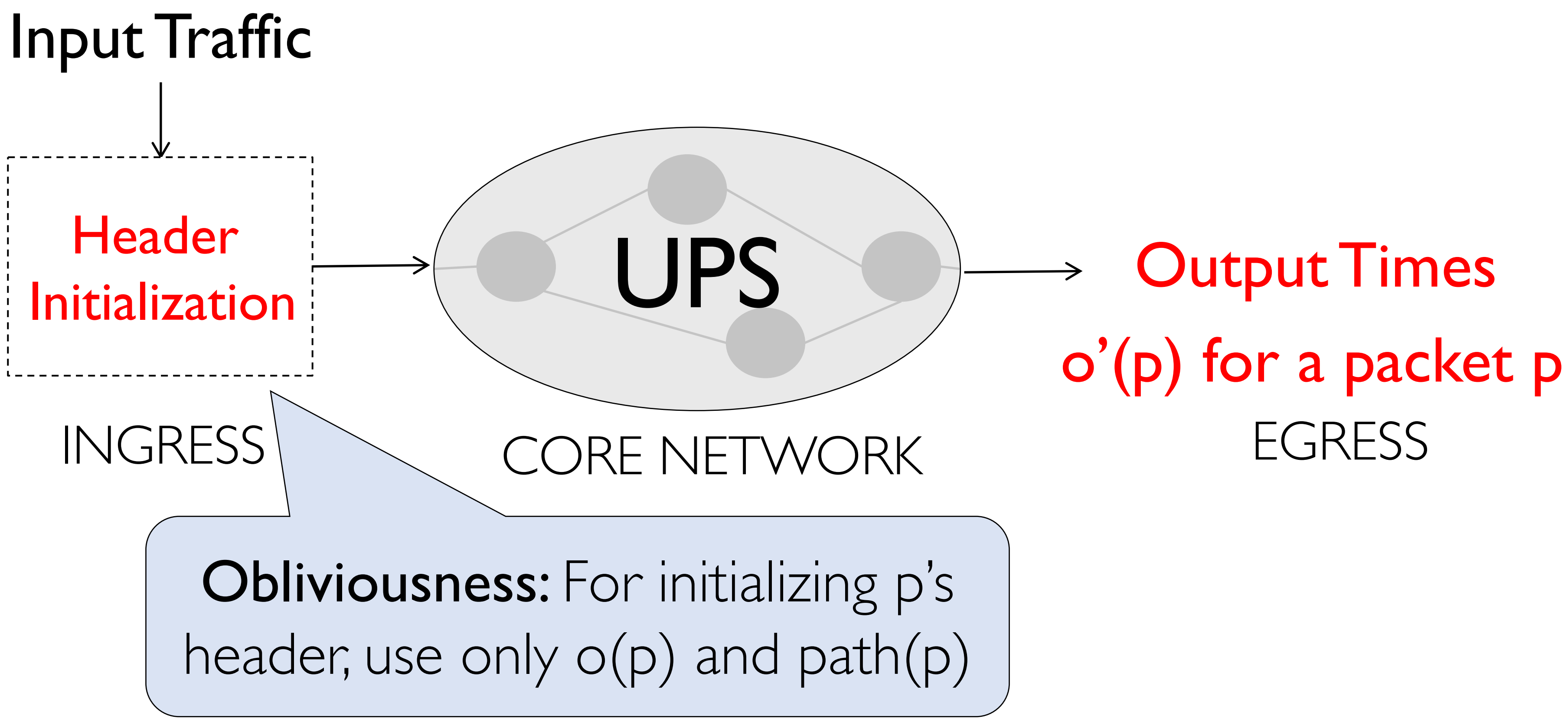


Replaying the Schedule, given o(p)

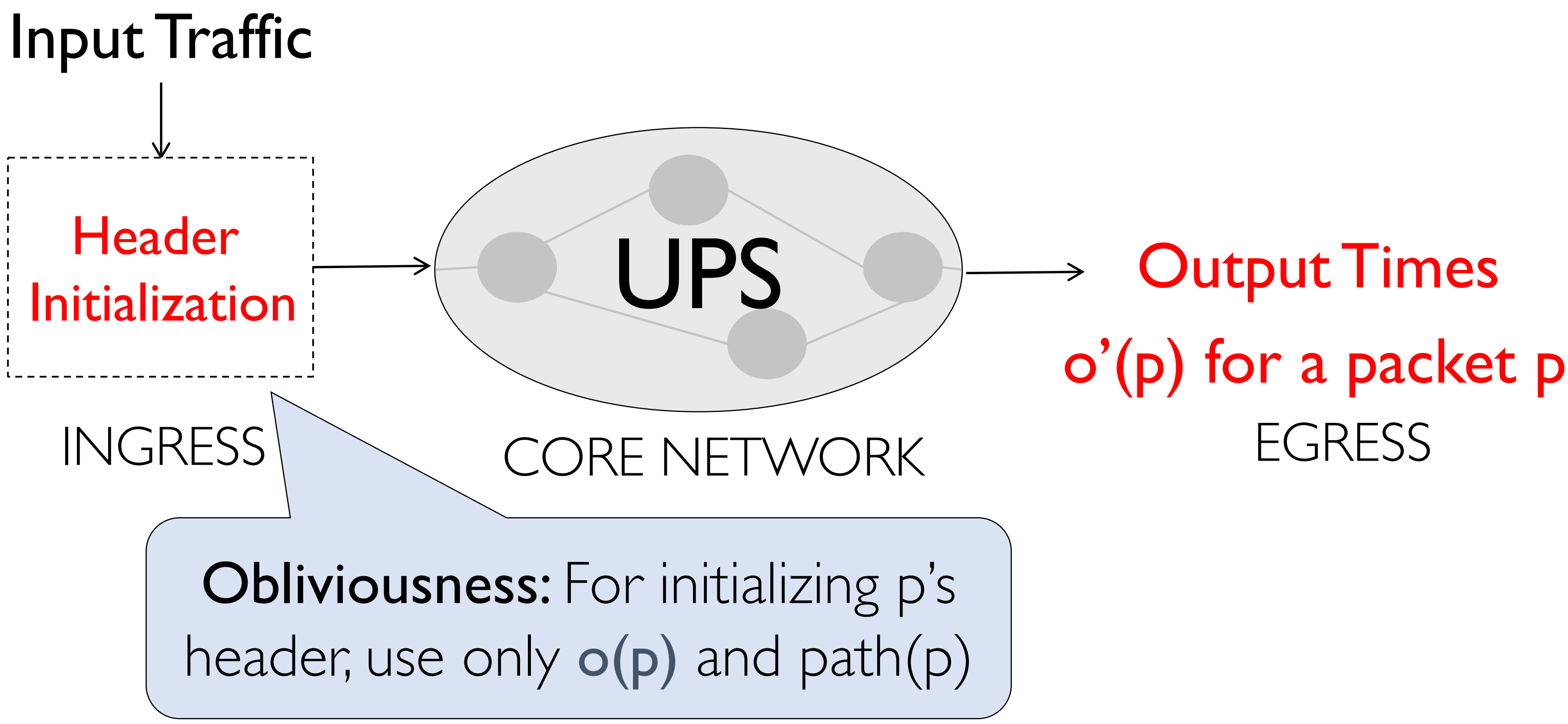


For every packet p, $o'(p) \leq o(p)$

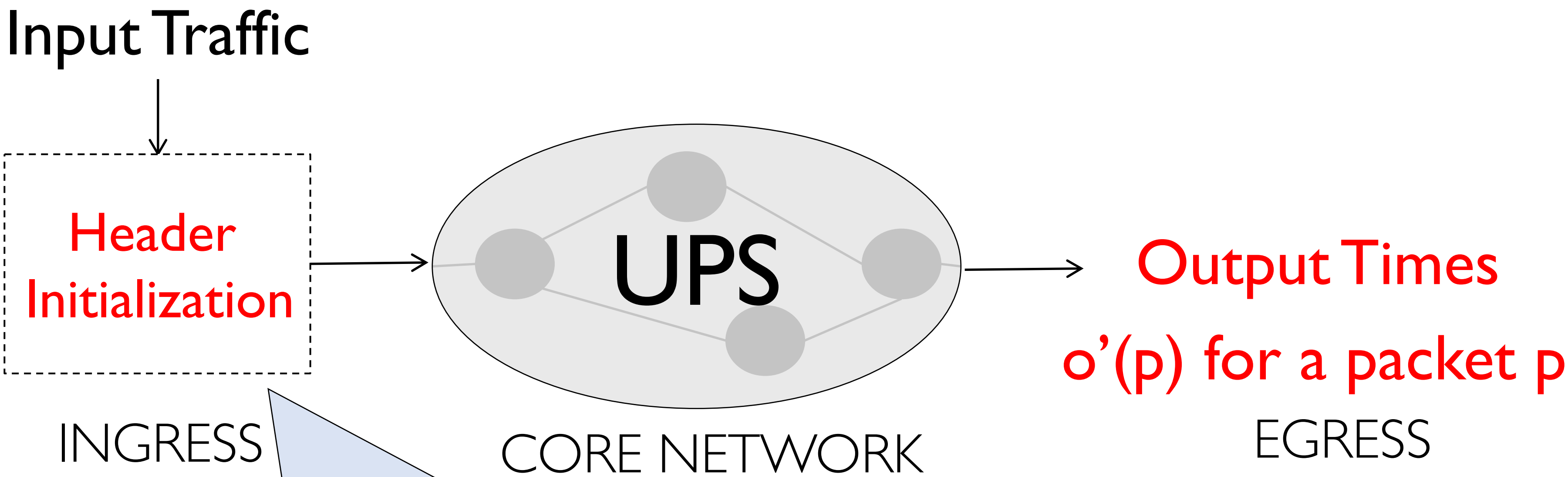
Pragmatic Constraints on a UPS



Pragmatic Constraints on a UPS

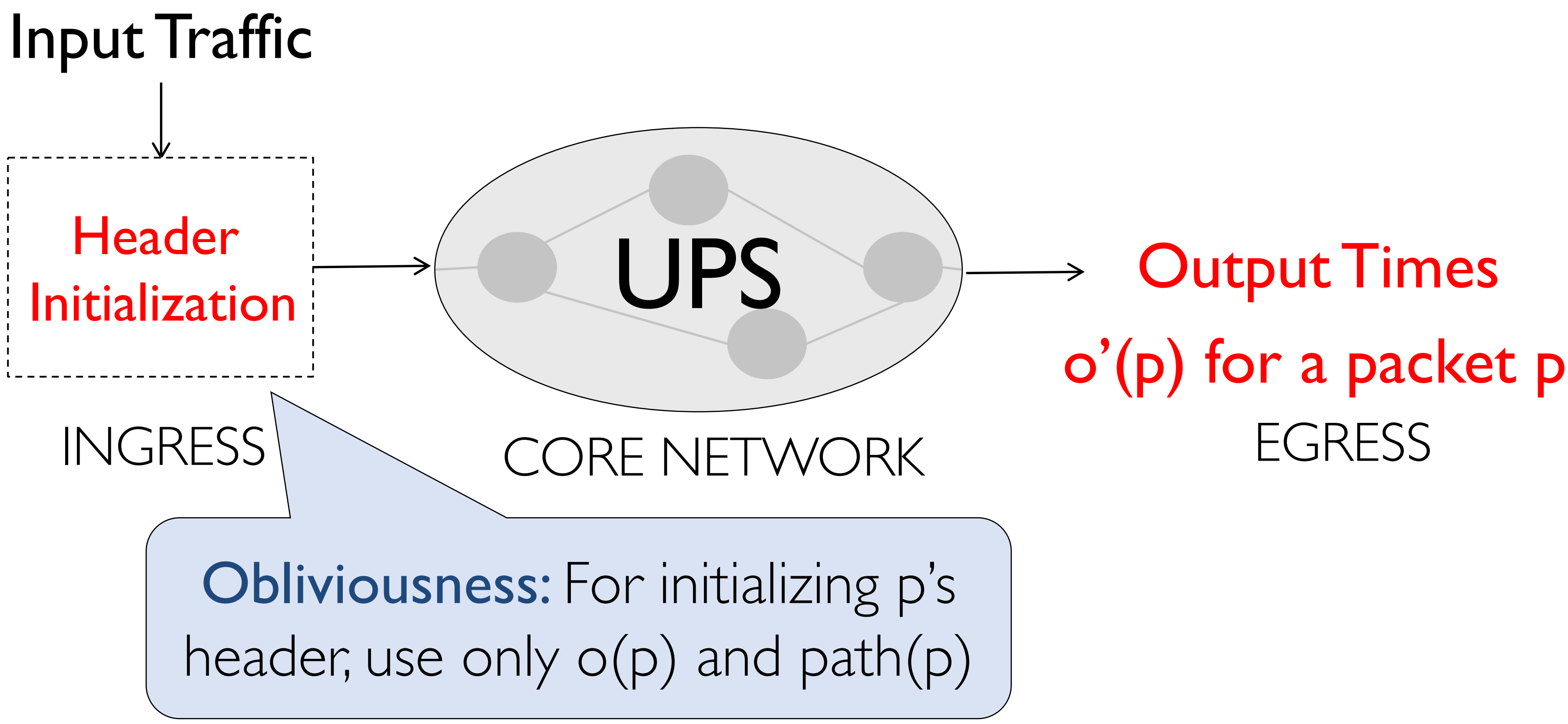


Pragmatic Constraints on a UPS

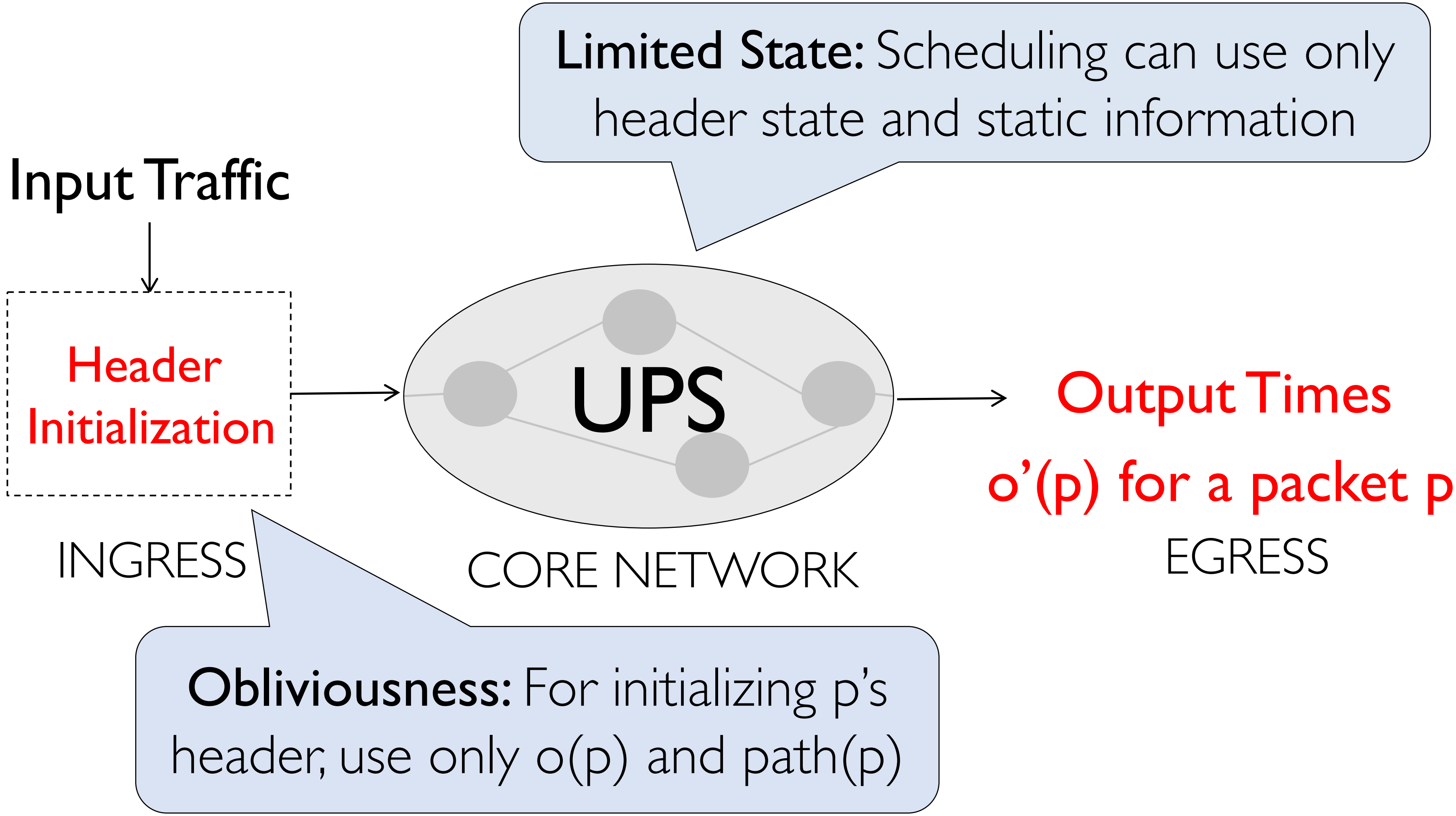


Obliviousness: For initializing p's header, use only o(p) and **path(p)**

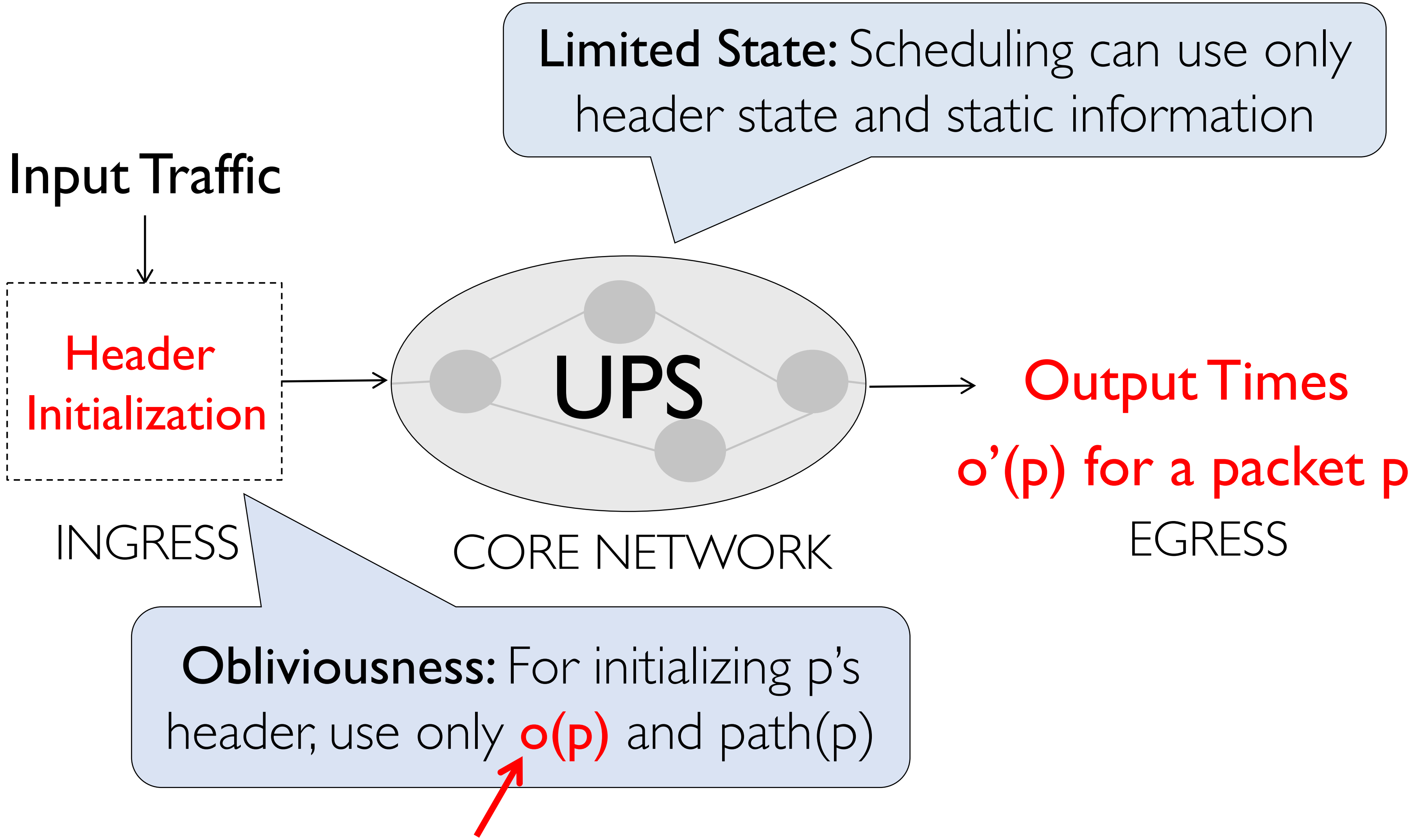
Pragmatic Constraints on a UPS



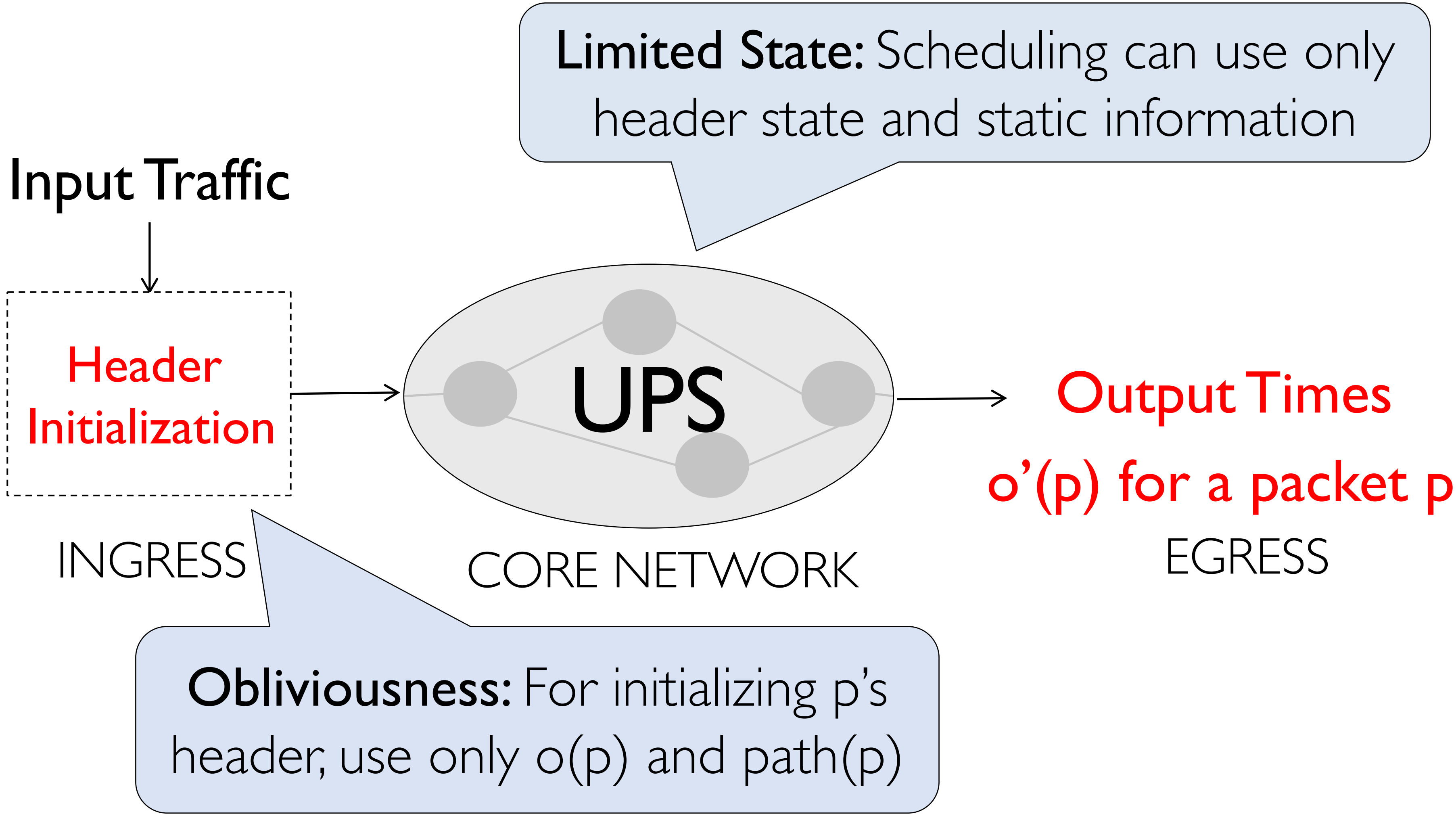
Pragmatic Constraints on a UPS



Pragmatic Constraints on a UPS



We call this Blackbox Initialization



Basic Existence and Non-existence Results

There exists a UPS under *Omniscient Initialization*

when scheduling time at every hop is known

No UPS exists under *Blackbox Initialization*

when only the final output time is known

See NSDI'16 paper for proofs.

How close can we get to a UPS?

Key Result: Depends on congestion points

No. of Congestion Points per Packet	General
1	✓
2	✓
3	✗

See NSDI'16 paper for proofs.

Can we achieve this upper bound?

Can we achieve this upper bound?

Yes, LSTF!

Least Slack Time First

- Packet header initialized with a slack value
  - slack = maximum tolerable queuing delay
- At the routers
  - Schedule packet with least slack time first
  - Update the slack by subtracting the wait time

Key Results

No. of Congestion Points per Packet	General	LSTF
1	✓	✓
2	✓	✓
3	✗	✗

See NSDI'16 paper for proofs.

Not all algorithms achieve upper bound

No. of Congestion Points per Packet	General	LSTF	Priorities
1	✓	✓	✓
2	✓	✓	✗
3	✗	✗	✗

See NSDI'16 paper for proofs.

How well does LSTF perform empirically?

Empirically, LSTF is (almost) universal

- ns-2 simulation results on realistic network settings
  - Less than 3% packets missed their output times
  - Less than 0.1% packets are late by more than one transmission time

Summarizing the theoretical viewpoint

- Evaluate the ability to replay a schedule, given its final output times

- Analytical Results:
  - No UPS exists
  - LSTF comes as close to a UPS as possible

- Empirical Results: LSTF is *almost* universal!

Practical Viewpoint

Can it achieve a given objective?

Achieving various network objectives

- Slack assignment based on heuristics
- Comparison with state-of-the-art
- Three objective functions
  - Tail packet delays
  - Mean Flow Completion Time
  - Fairness

Tail Packet Delays

Slack Assignment: Same slack for all packets

State-of-the-art: FIFO, FIFO+

Results:

- Identical to FIFO+.
- Smaller tail packet delays compared to FIFO.

Mean Flow Completion Time

Slack Assignment: Proportional to flow size

State-of-the-art: SJF, SRPT

Results:

- Mean FCTs comparable to both SJF and SRPT.

Fairness

Slack Assignment: Inspired by Virtual Clocks

$$\text{slack}(p_0) = 0$$

$$\text{slack}(p_i) = \max(0, \text{slack}(p_{i-1}) + (1/r_{est}) - (i(p_i) - i(p_{i-1}))$$

r_{est} = Estimate of fair share rate

State-of-the-art: Fair Queuing (FQ)

Results:

- Eventual convergence to fairness for long-lived flows.
- FCTs roughly comparable to FQ for short-lived flows.
  - Higher sensitivity to fair share rate estimate (r_{est})

Results Summary

- Theoretical results show that
  - There is no UPS under blackbox initialization
  - LSTF comes as close to a UPS as possible
  - Empirically, LSTF is very close

- LSTF can be used in practice to achieve a variety of network-wide objectives.

Implication

- Less need for many different scheduling algorithms.

- Can just use LSTF, with varying initializations.

Limitations

- Policies for which the required information is not available during header initialization at the ingress.
  - When relative ordering between two packets changes after enqueuing them.
  - Class-based weighted fairness.

Follow-up work along similar lines…

- Fast, Scalable, Programmable Packet Scheduler (SIGCOMM'19)
  - Increase flexibility and scalability of PIFO through a different design.
- SP-PIFO (NSDI'20)
  - Most switches have only 8-16 queues. What's the best we can do with existing switch hardware?
- Programmable Calendar Queues (NSDI'20)
  - A different abstraction to approximate different scheduling algorithms using fixed number of FIFO queues.
- Programmable packet scheduling with a single queue (SIGCOMM'22)
  - Approximate PIFO using a single FIFO queue.
  - Datacenter buffers are shallow, so only need to decide which packets should be enqueued vs which ones should be dropped.

Logistics

- First progress report is due on Friday (Oct 13) at11:59pm.
  - See course website (assignment tab) for submission instructions.

- Need to end OH at 3pm today.
  - Supplemental OH till 3pm on Friday.