

Universal Packet Scheduling

Radhika Mittal, Rachit Agarwal,

Sylvia Ratnasamy, Scott Shenker

UC Berkeley

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*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotNets 2015

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*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotNets 2015

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*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotNets 2015

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*

*Towards Programmable Packet Scheduling, Sivaraman et. al., HotNets 2015

We are asking a new question…..

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

Is there a *universal* packet scheduling algorithm?

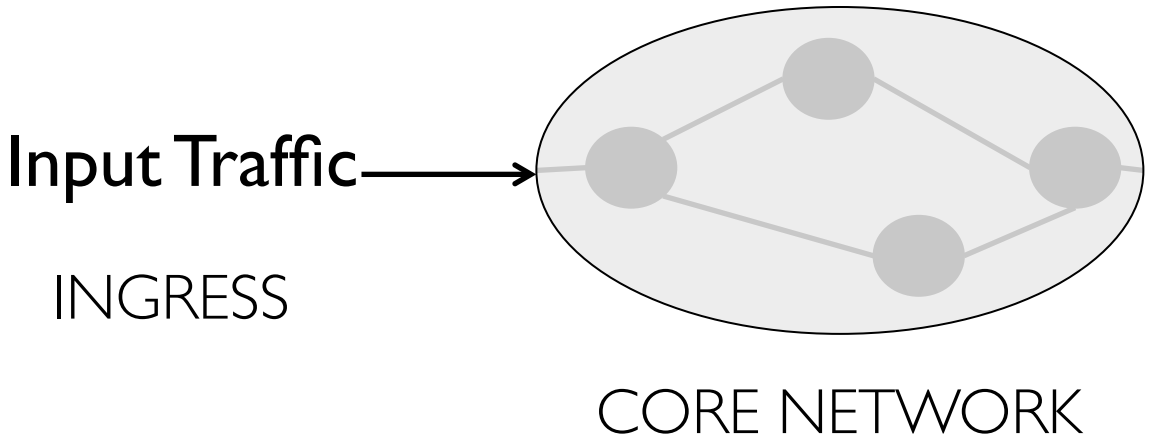
UPS: Universal Packet Scheduling Algorithm

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

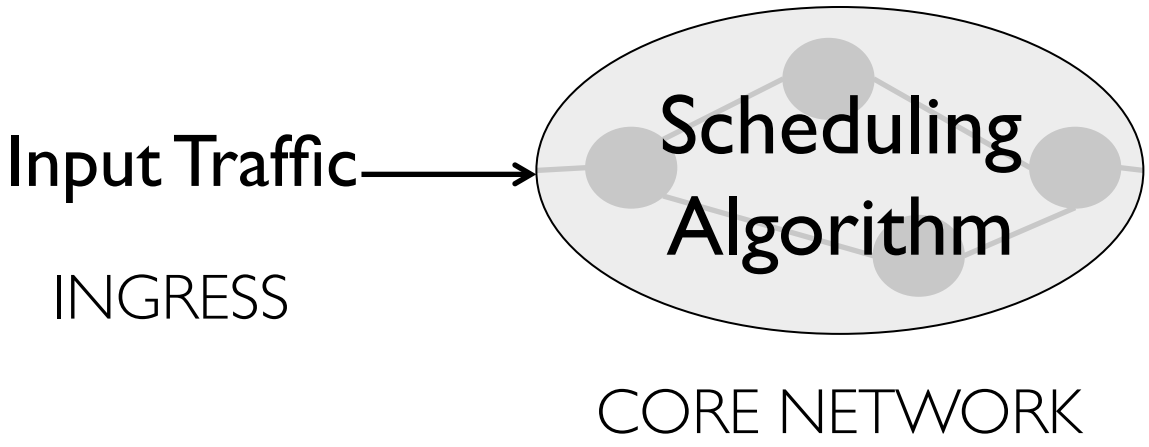
How can a single algorithm imitate all others?



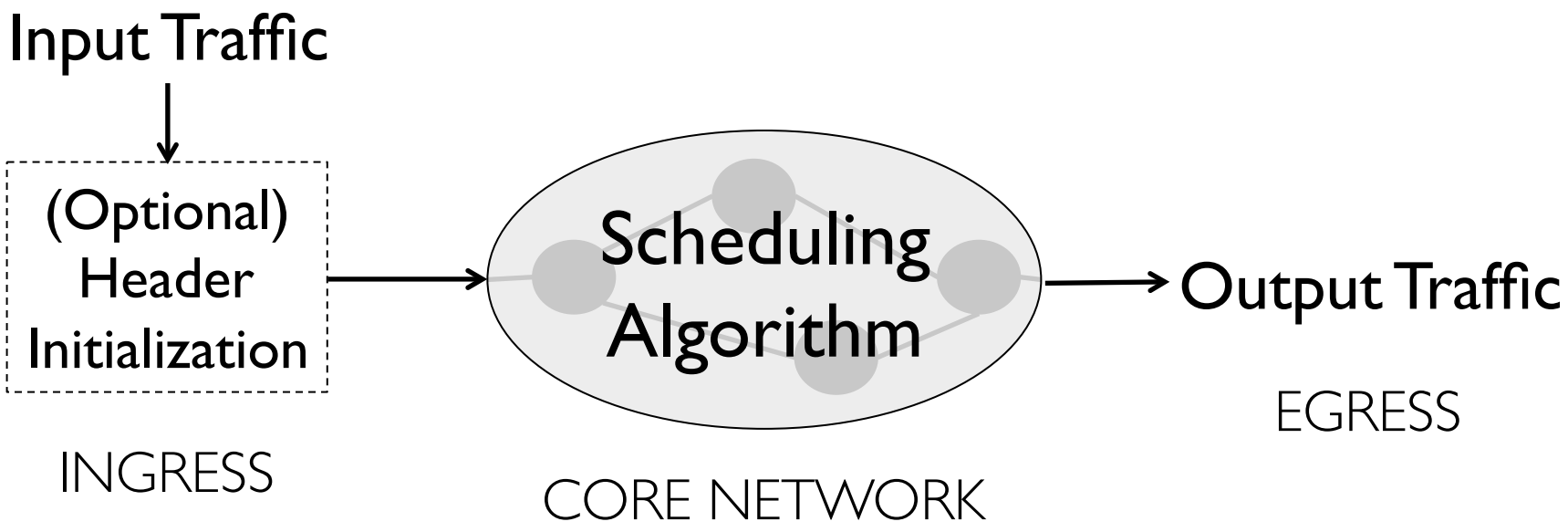
Network Model



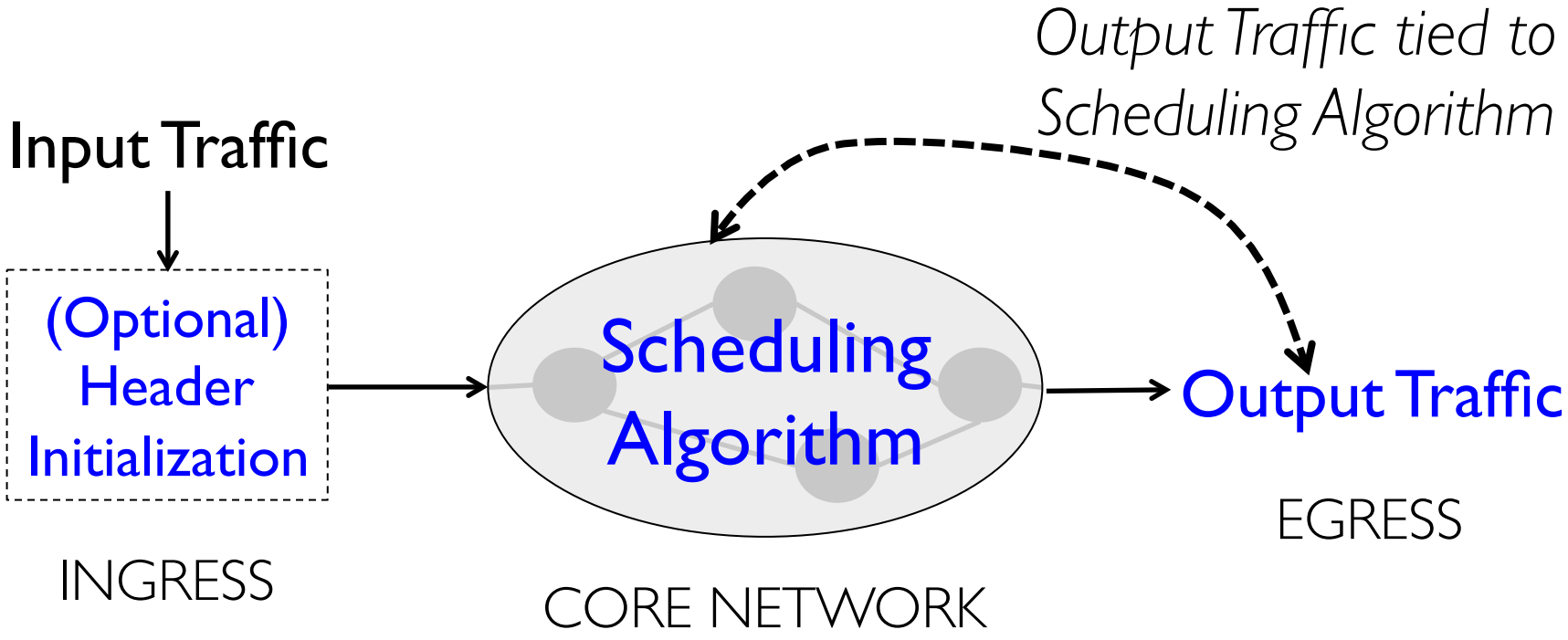
Network Model



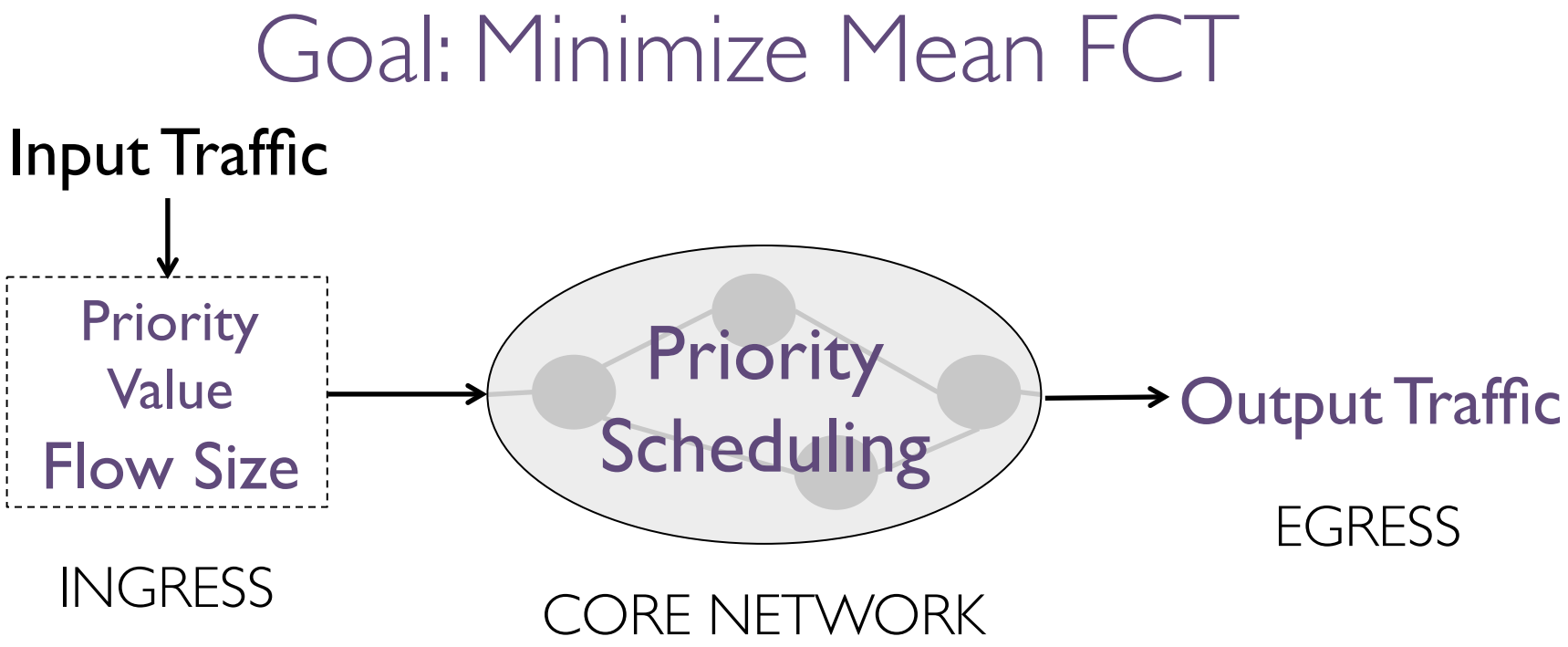
Network Model



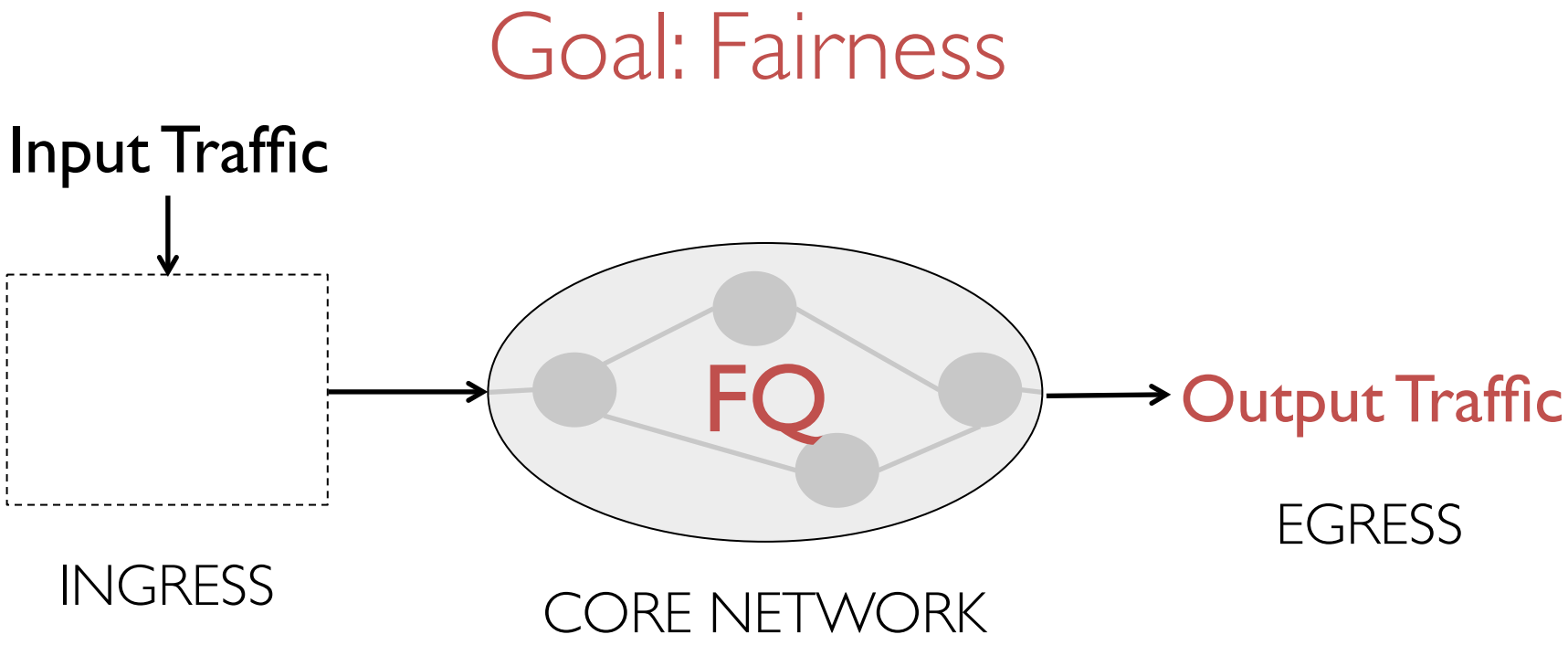
Network Model



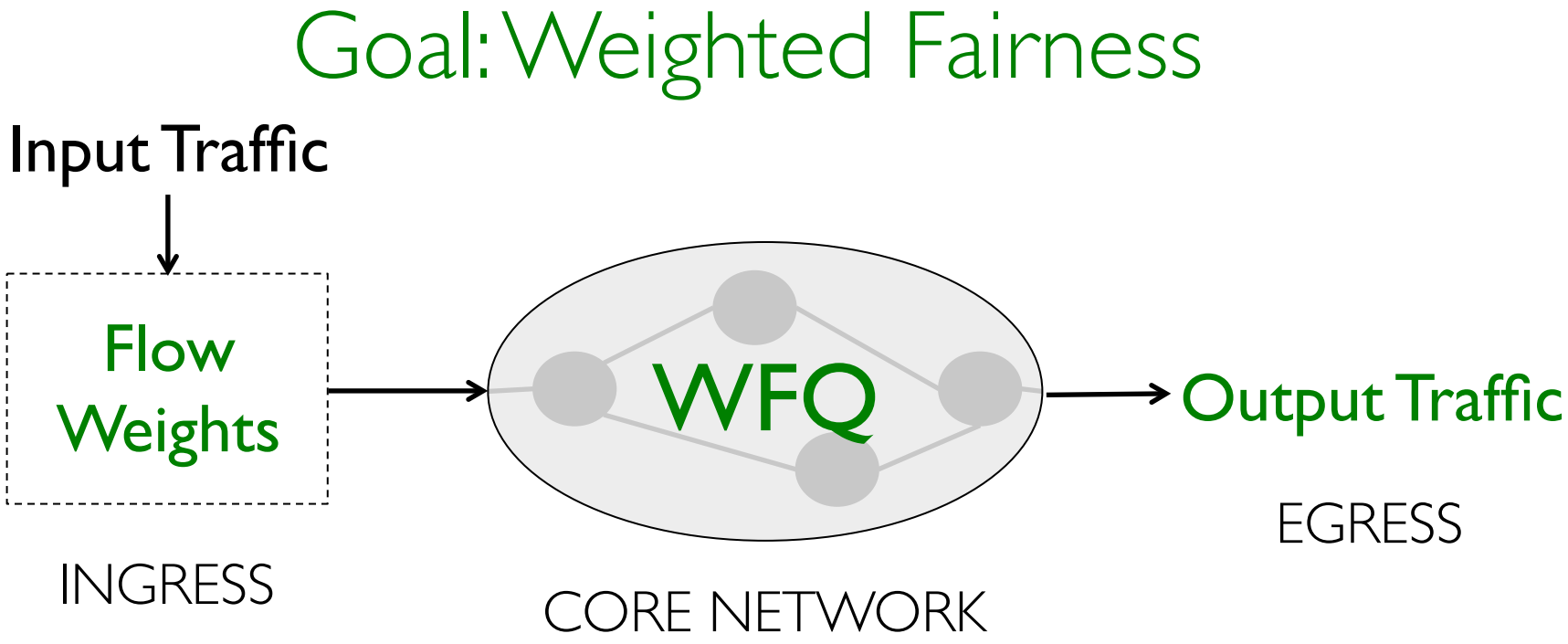
Network Model



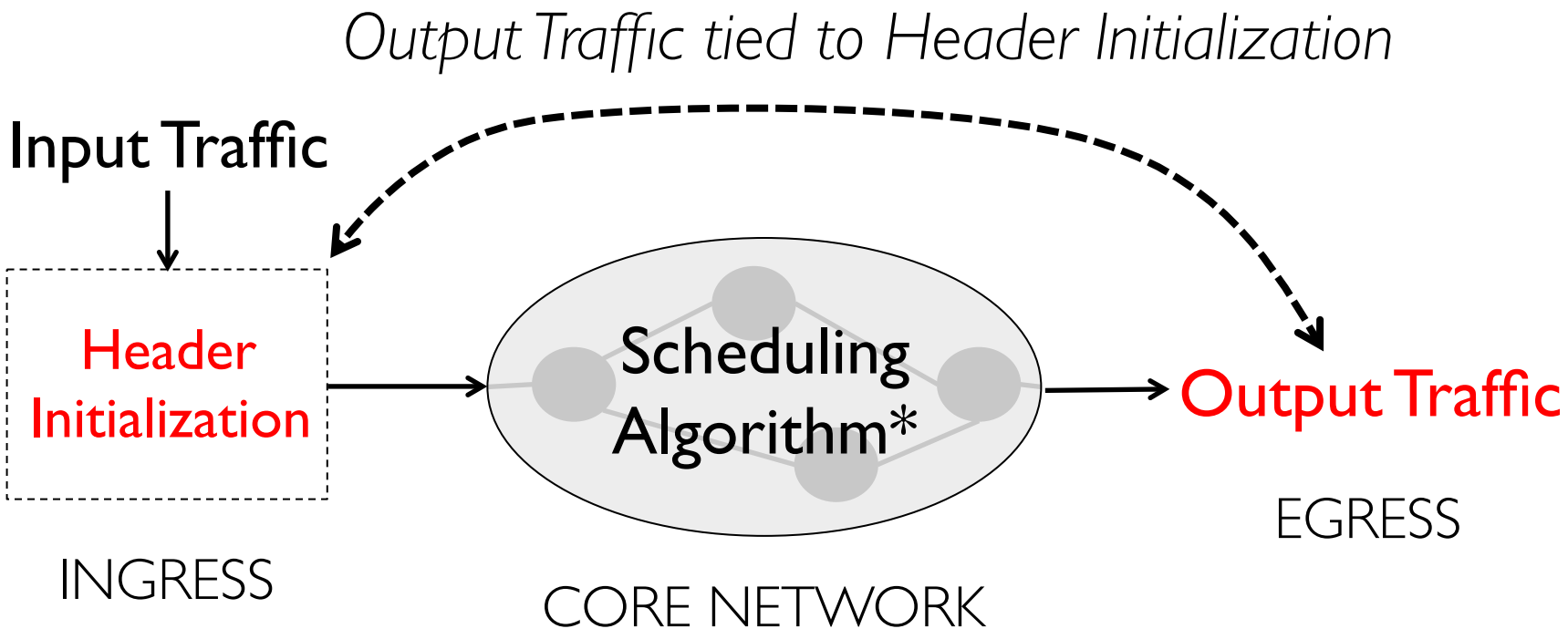
Network Model



Network Model

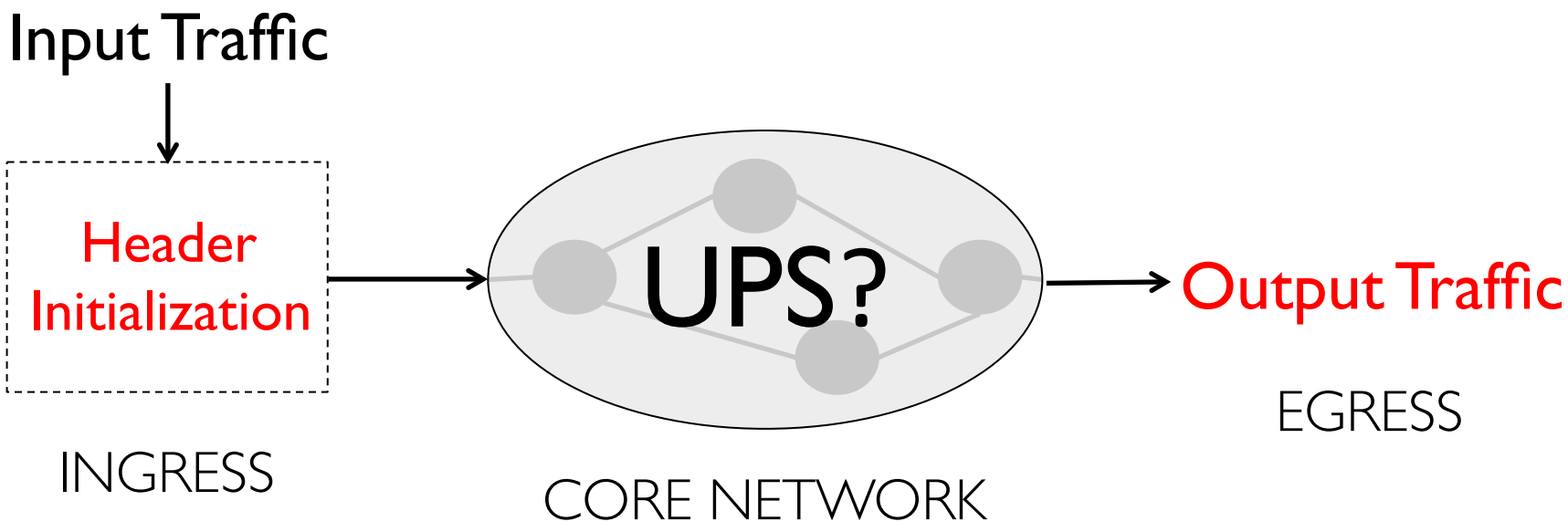


Network Model

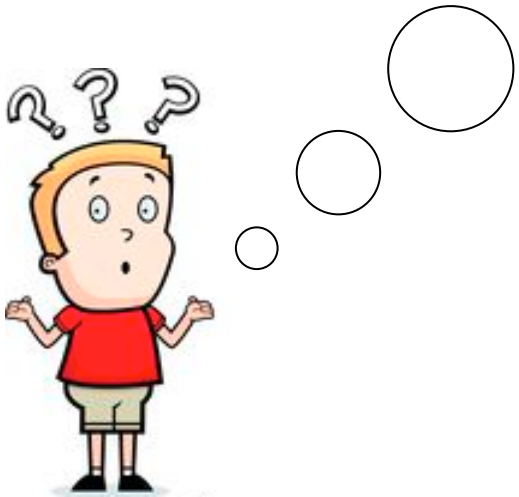


* Uses packet header state to make scheduling decisions

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

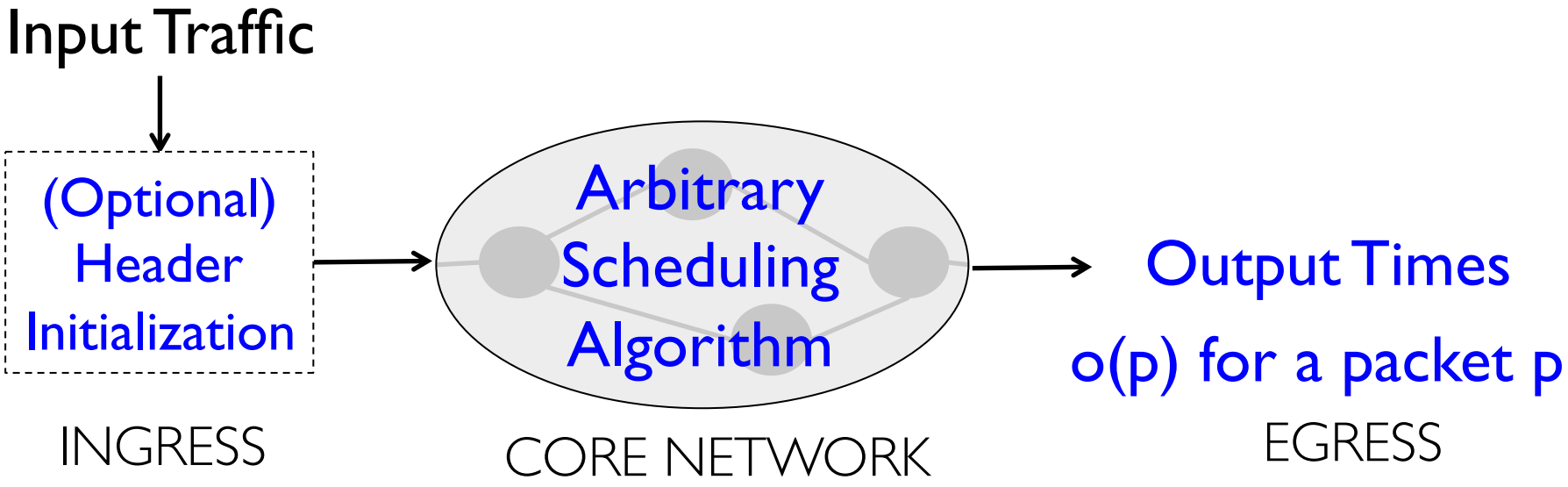
Can it replay a given schedule?



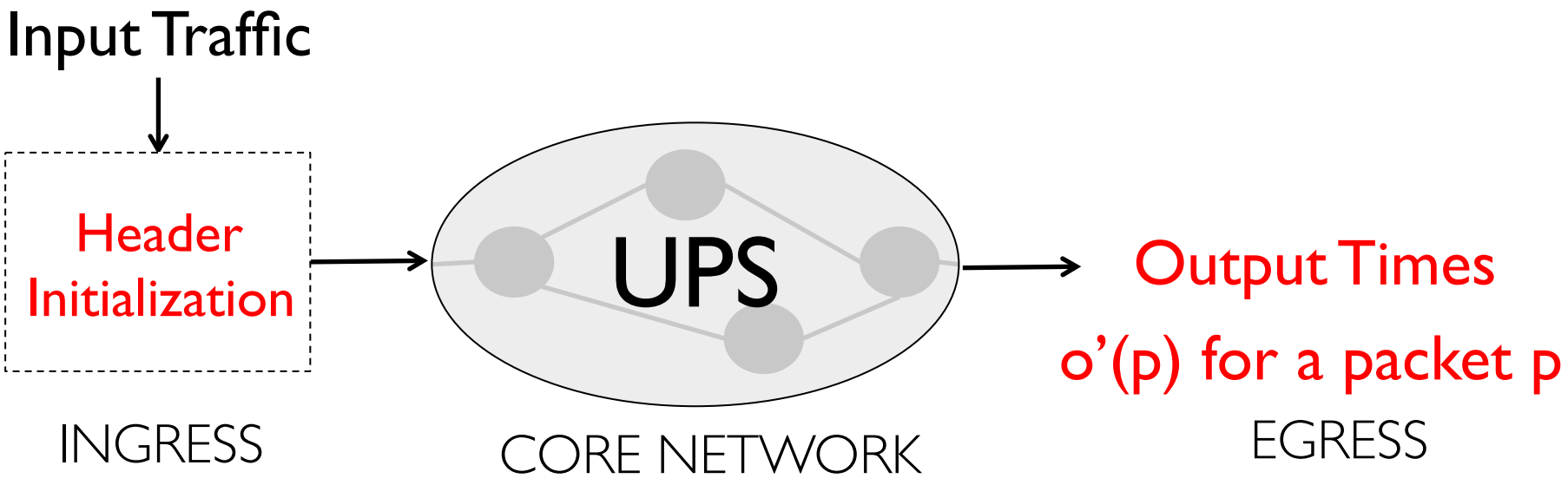
Original Schedule

Only requirement from original schedule:

Output Times are viable

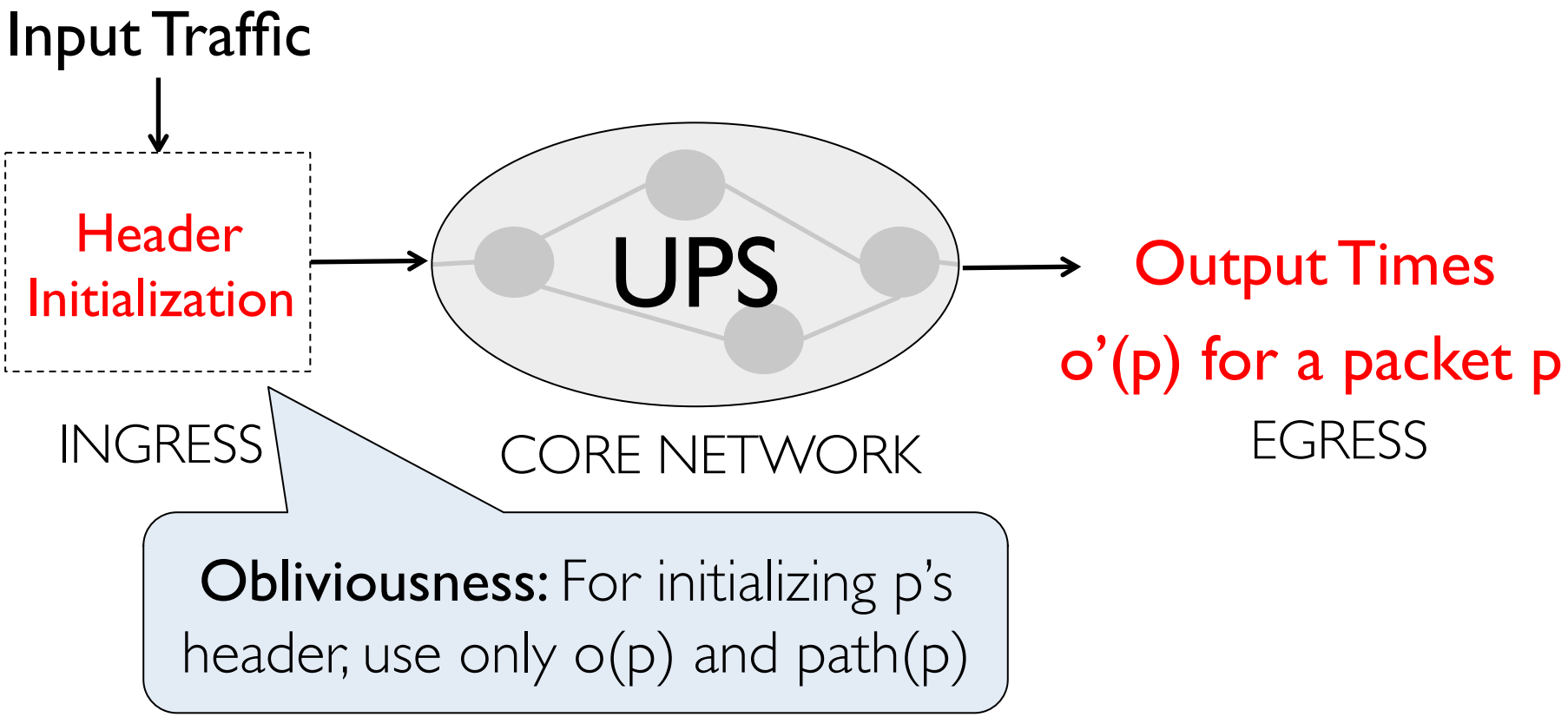


Replaying the Schedule, given o(p)

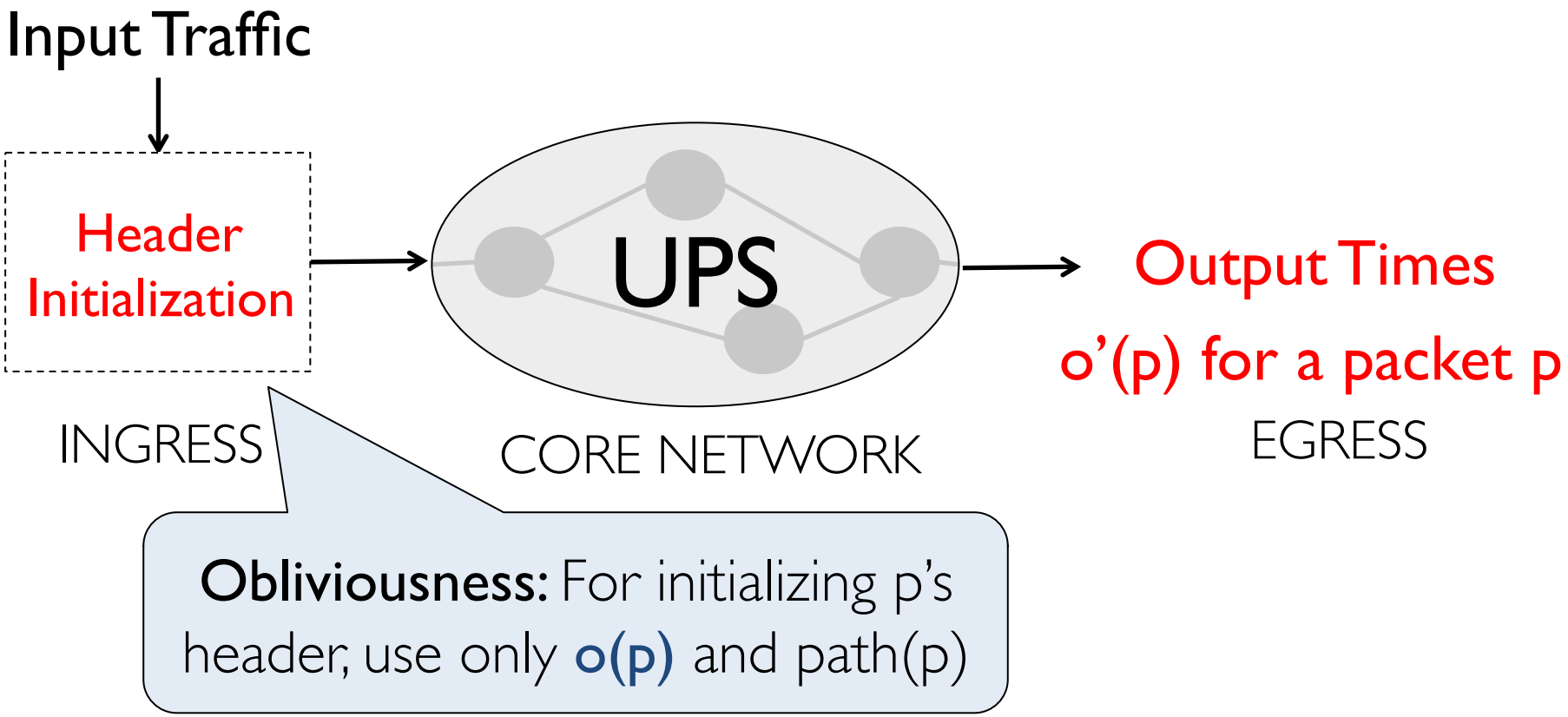


For every packet p, $o'(p) \leq o(p)$

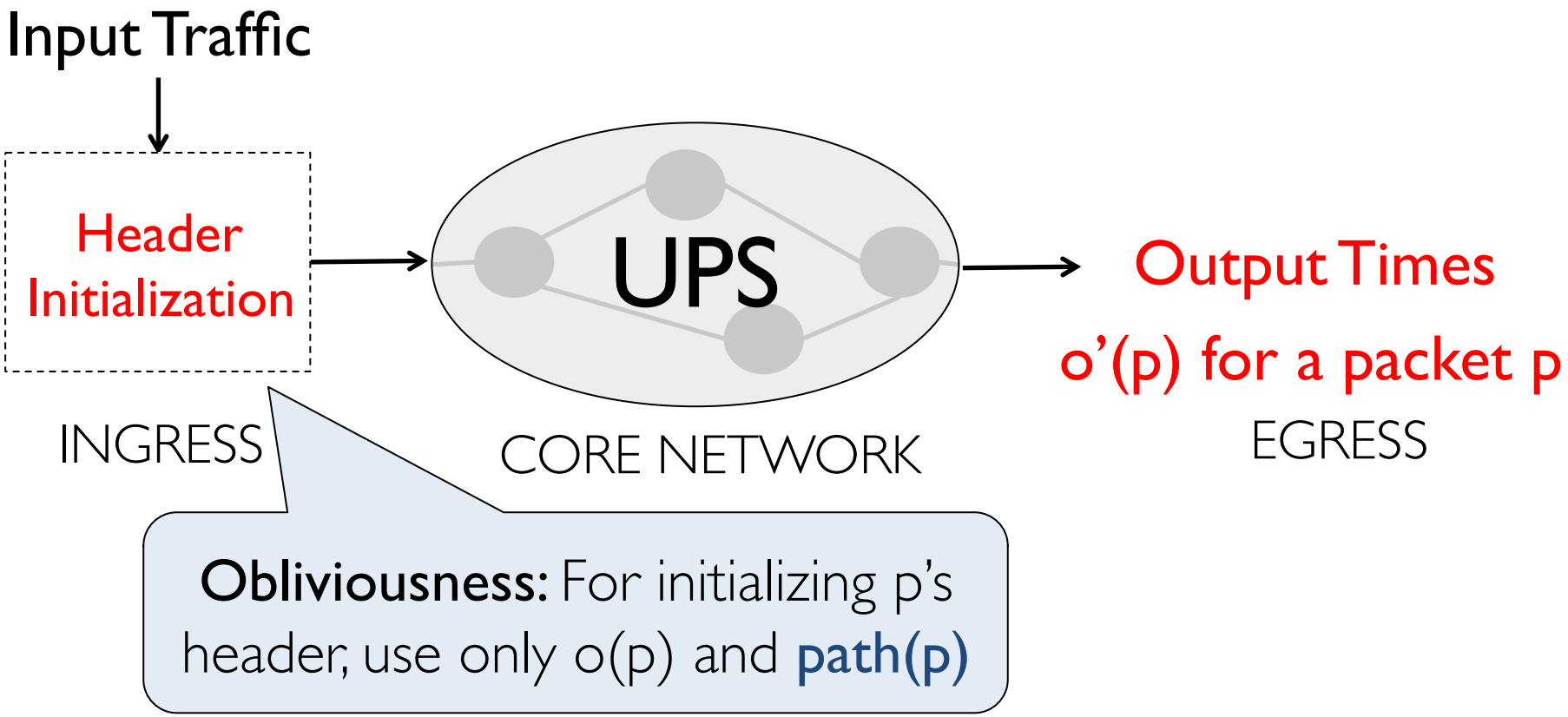
Pragmatic Constraints on a UPS



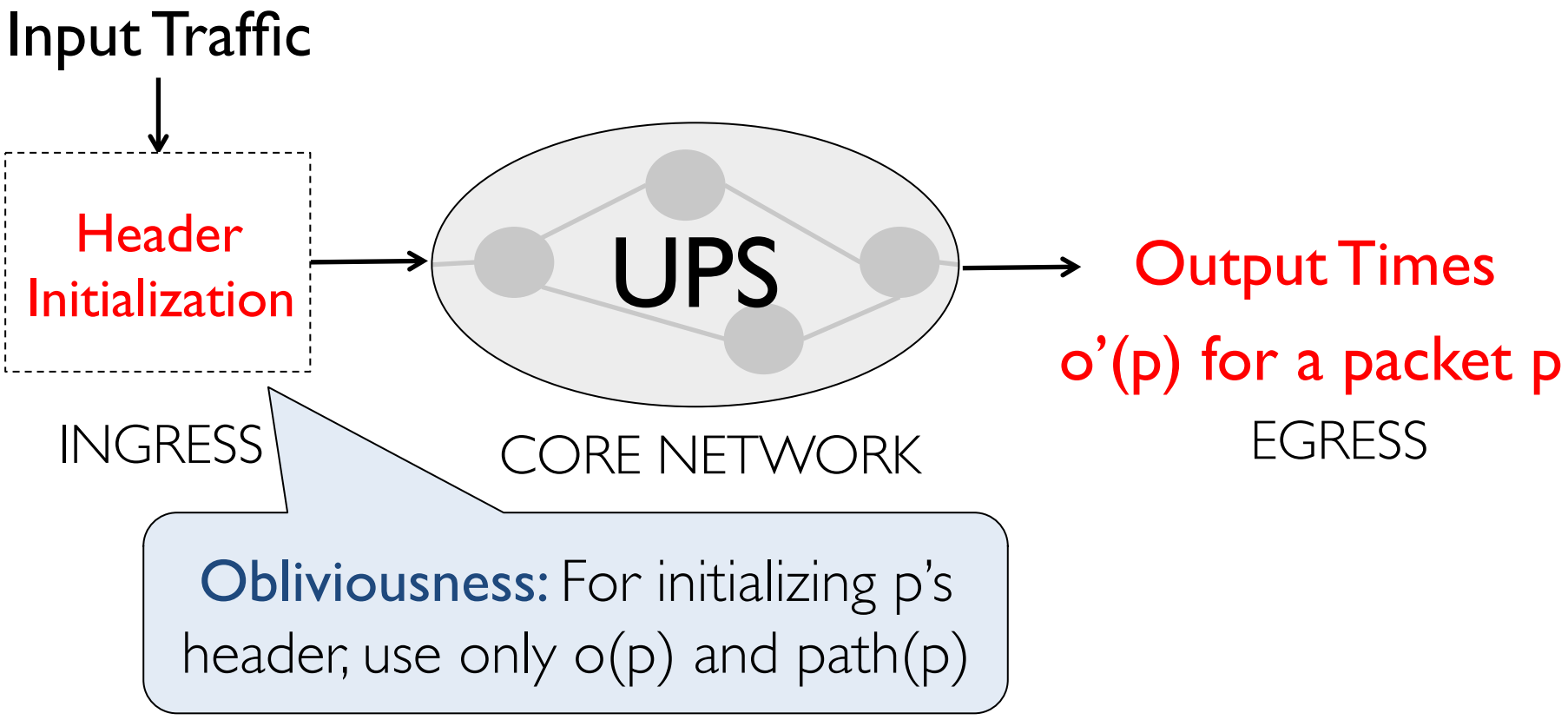
Pragmatic Constraints on a UPS



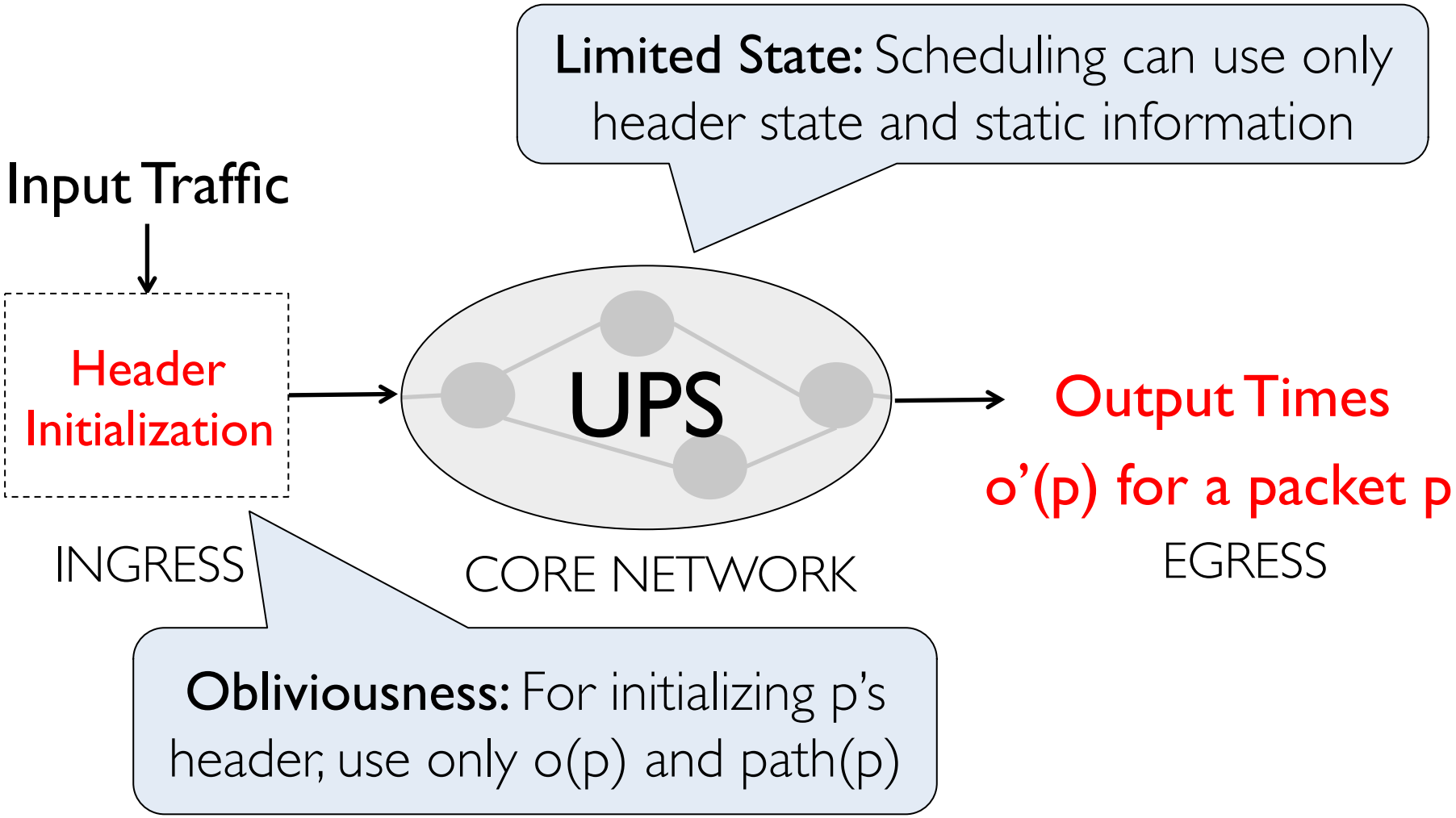
Pragmatic Constraints on a UPS



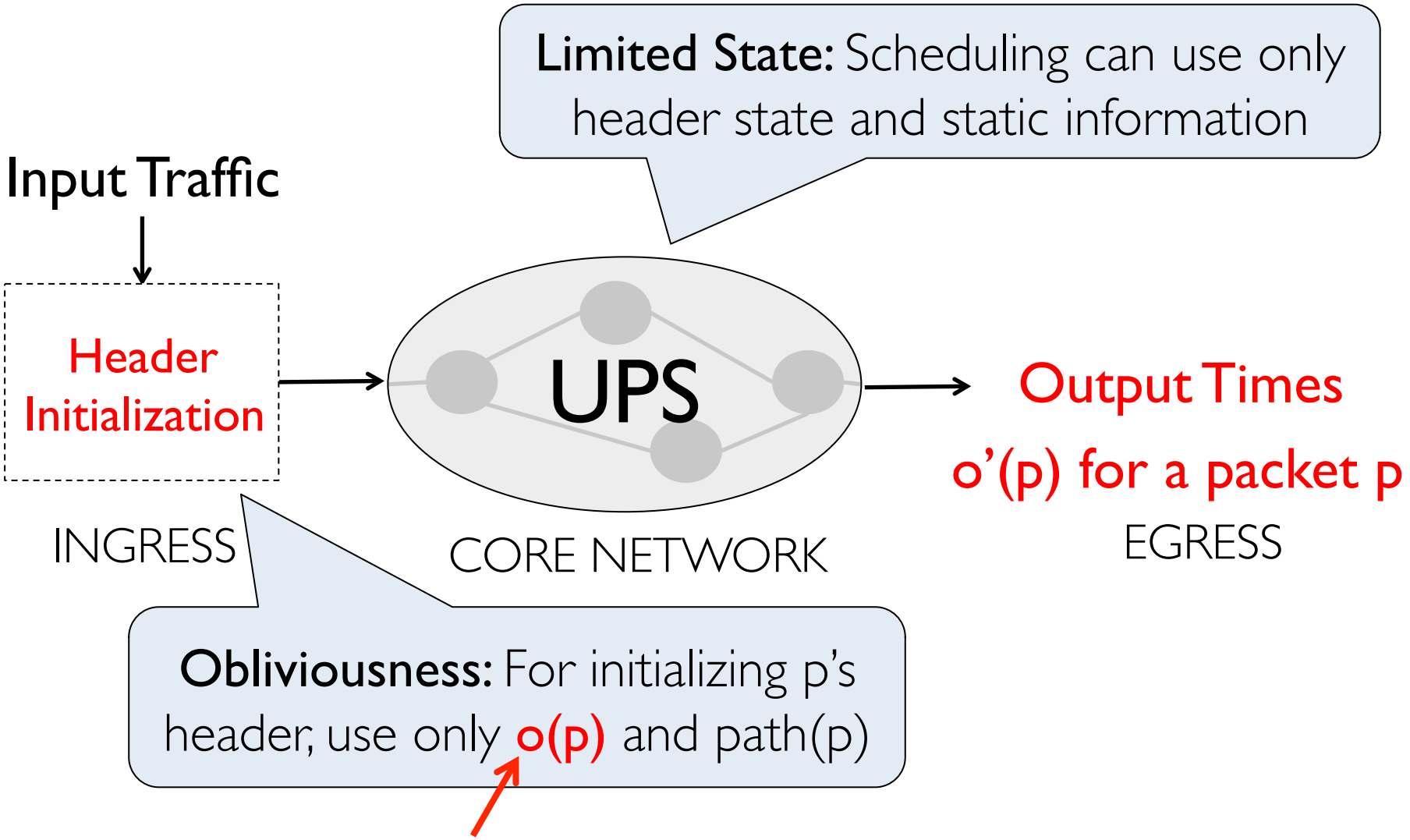
Pragmatic Constraints on a UPS



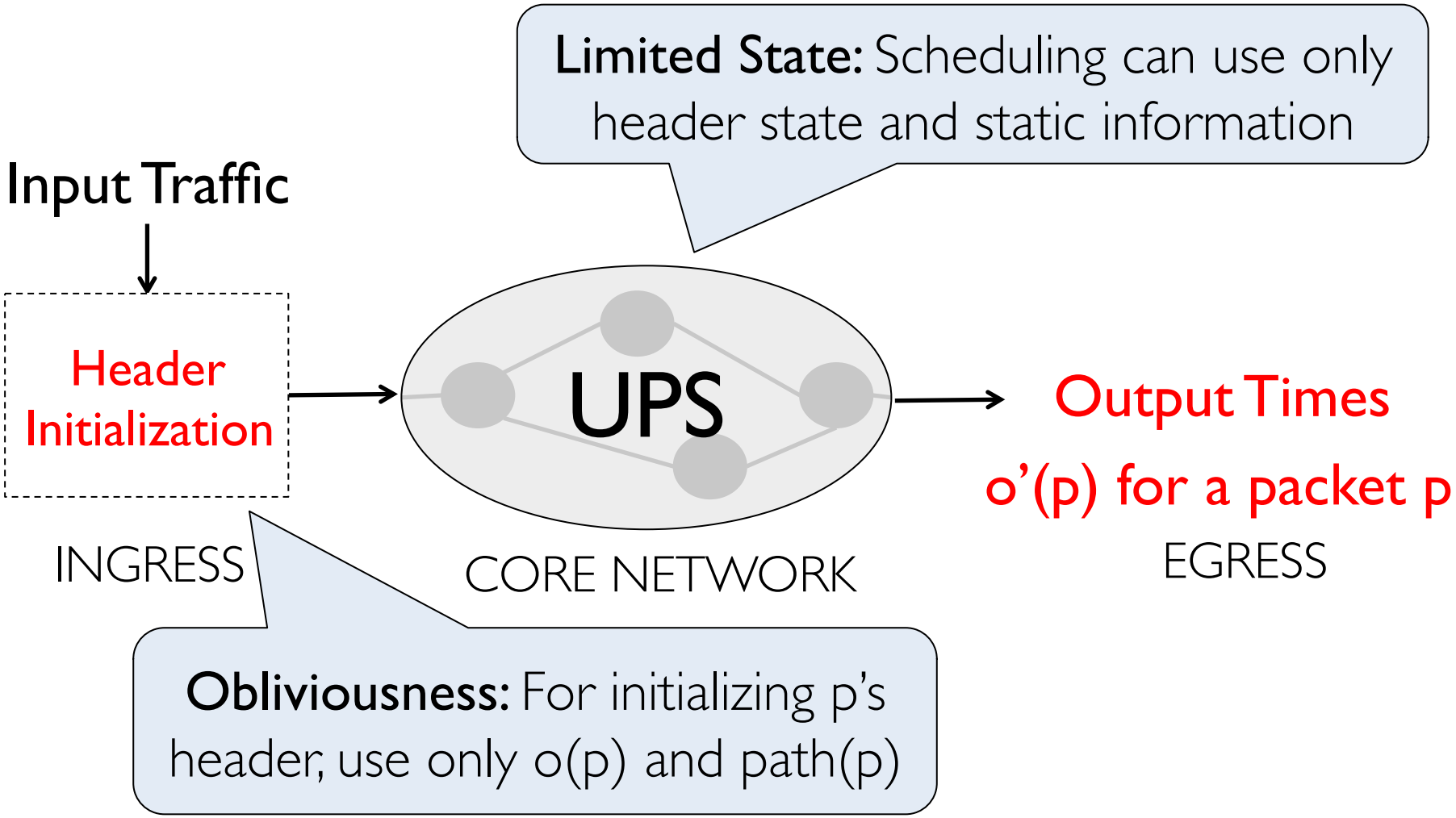
Pragmatic Constraints on a UPS



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 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 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 paper for proofs.

Not all algorithms achieve upper bound

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

See 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

$slack(p_0) = 0$

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

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})

Active Queue Management (AQM)

- Routers sends feedback in the form of dropping or marking appropriate packets.
- LSTF facilitates AQM from the edge:
  - It does not matter *where* the packets are dropped or marked.
  - *Used slack* value can be used for deciding *which* packets are to be dropped or marked.
- Performs comparable to FQ-CoDel and DCTCP (ECN).

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 and queue management algorithms.

- Can just use LSTF, with varying initializations.

There are still a
bunch of
open questions!



Open Questions



What is the least amount of information needed to achieve universality?



Are there tractable bounds for the degree of lateness with LSTF?



How do we achieve multiple objectives simultaneously?



What is the class of objectives that can be achieved with LSTF *in practice*?

Conclusion

- 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.

Contact: radhika@eecs.berkeley.edu

Code: http://netsys.github.io/ups/

Thank You!