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., HotN

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., HotN

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., HotN

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., HotN

We are asking a new question…..

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

Is there a *universal* packet scheduling algorithm?

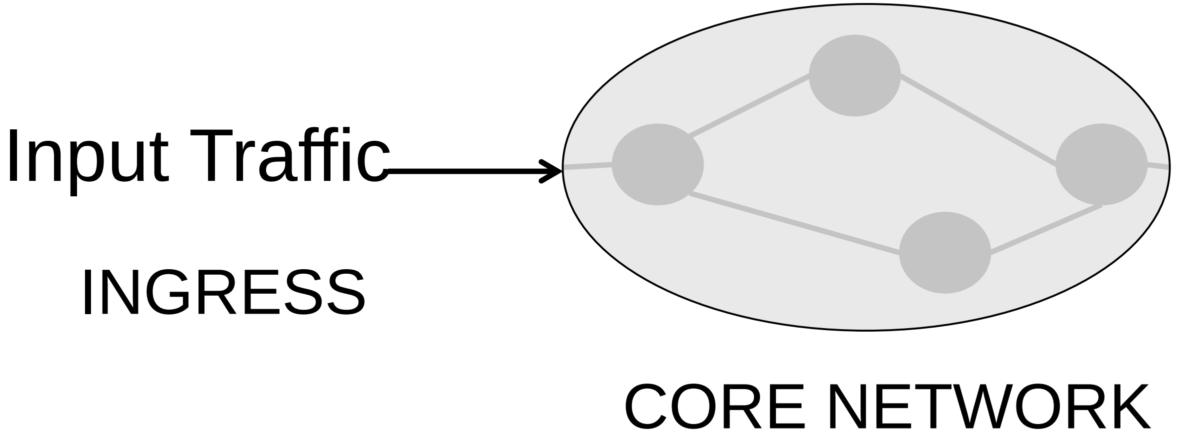
UPS: Universal Packet Scheduling Algorithm

A *single scheduling algorithm* that

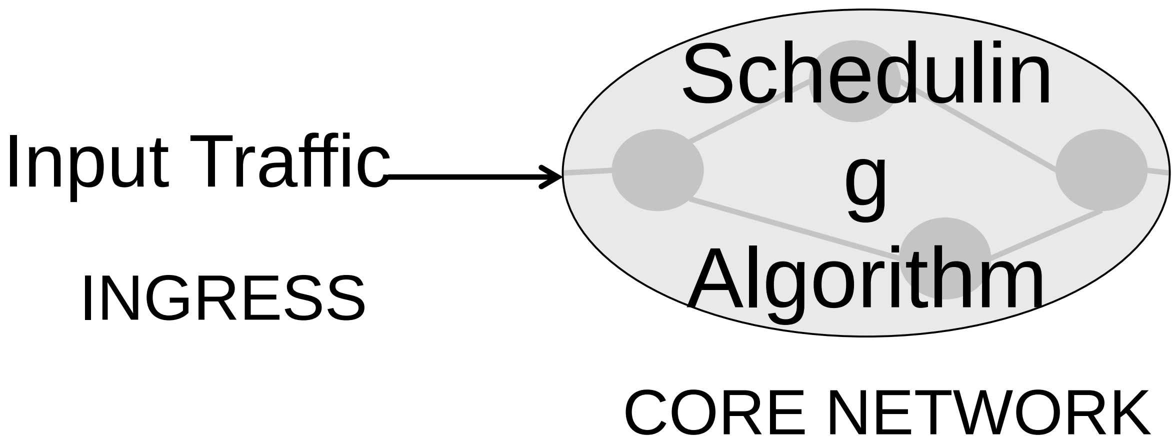
can imitate the network-wide output produced by ***any*** other algorithm.

How can a single algorithm imitate all others?

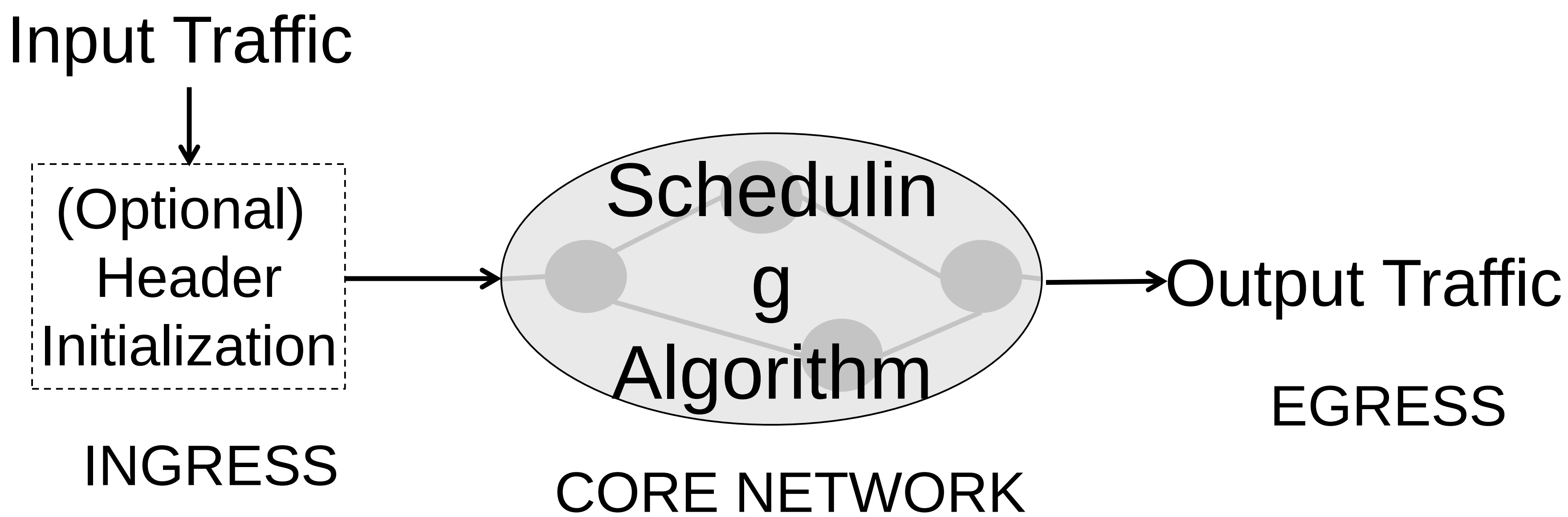
Network Model



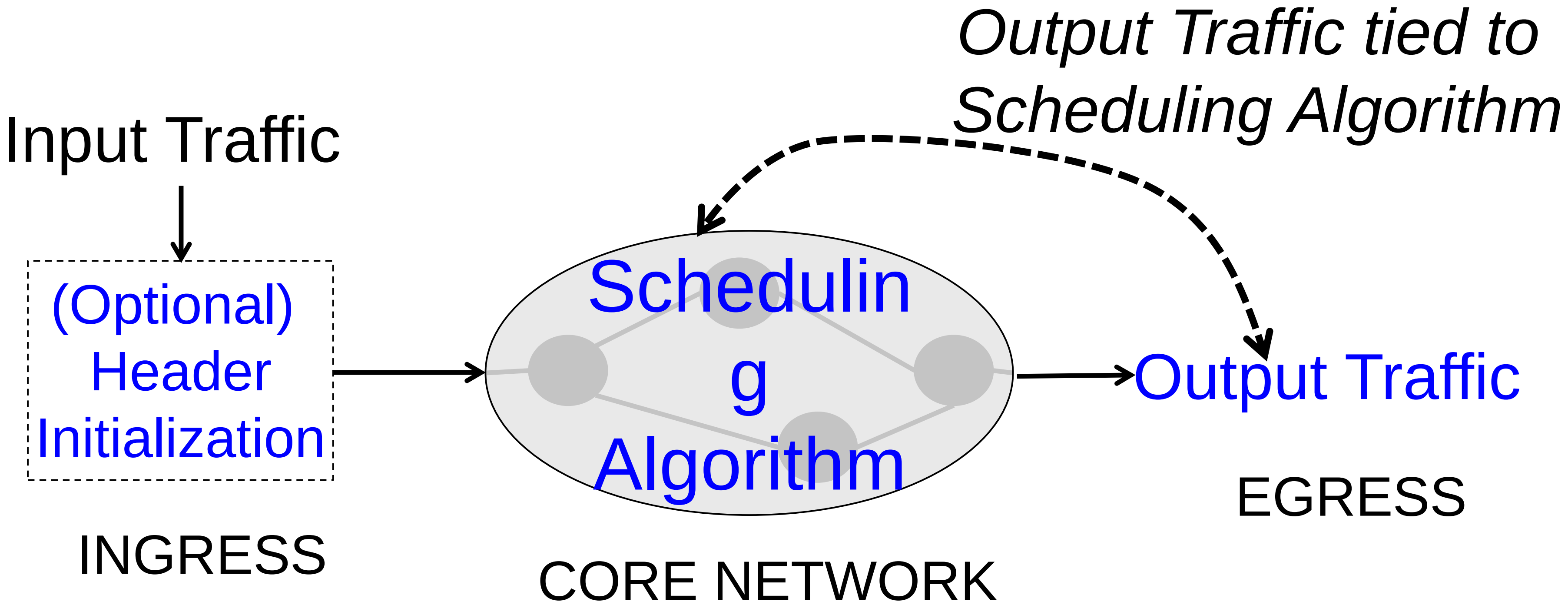
Network Model



Network Model

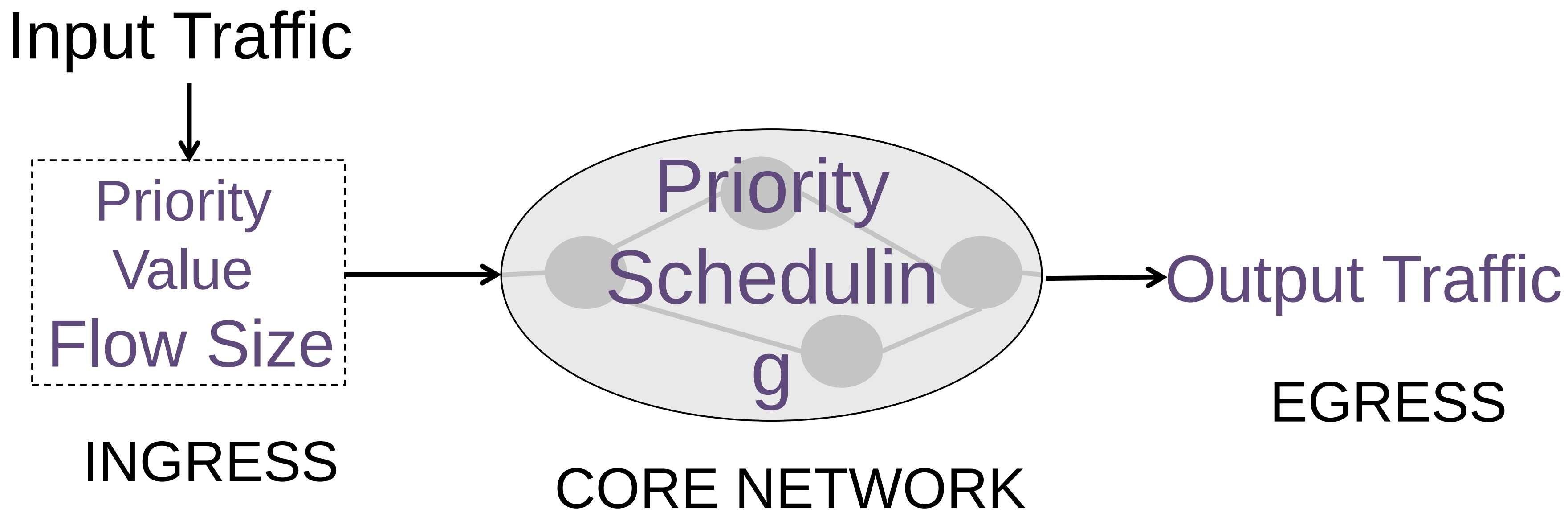


Network Model



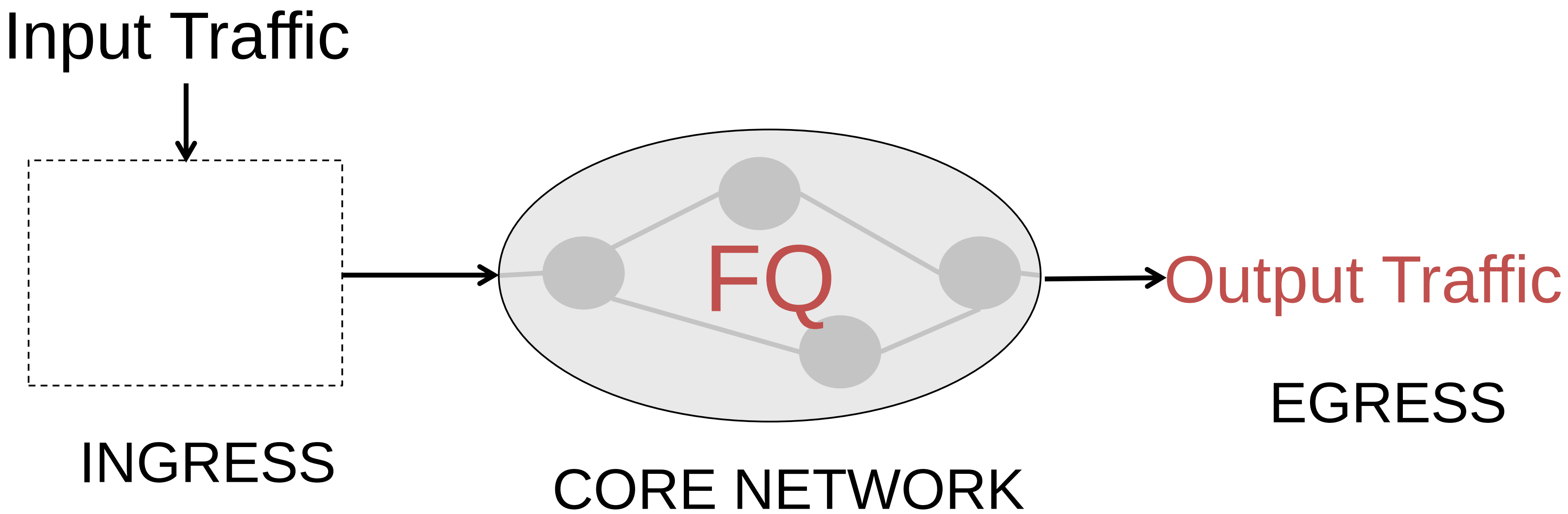
Network Model

Goal: Minimize Mean FCT



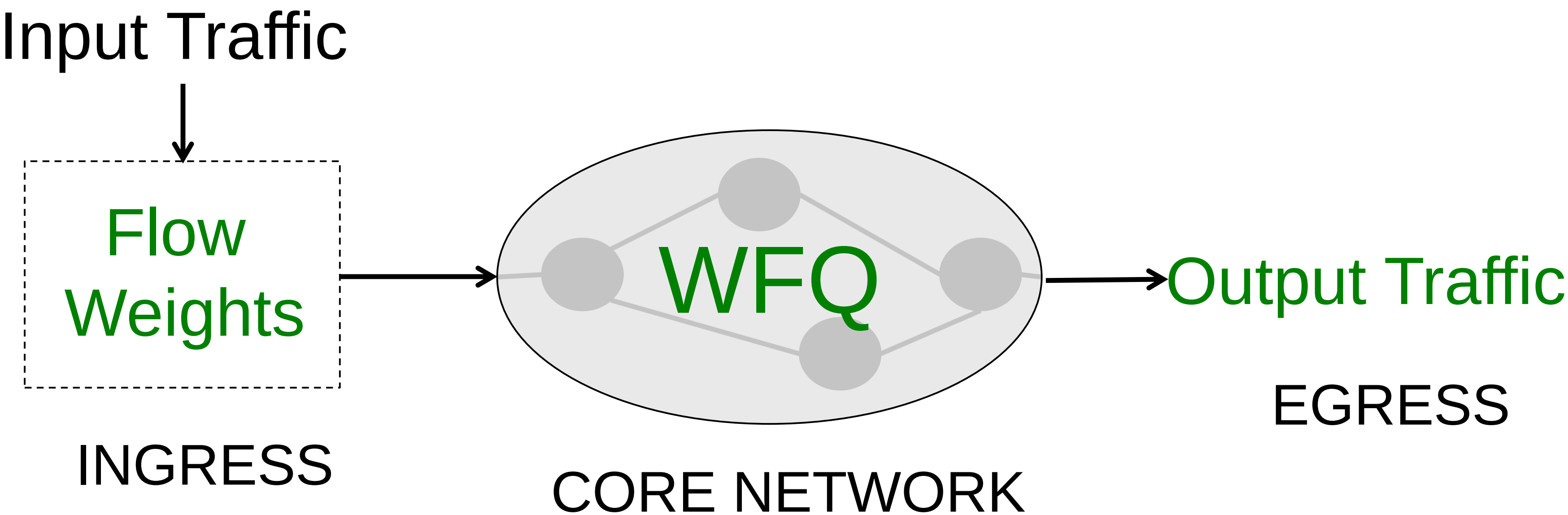
Network Model

Goal: Fairness

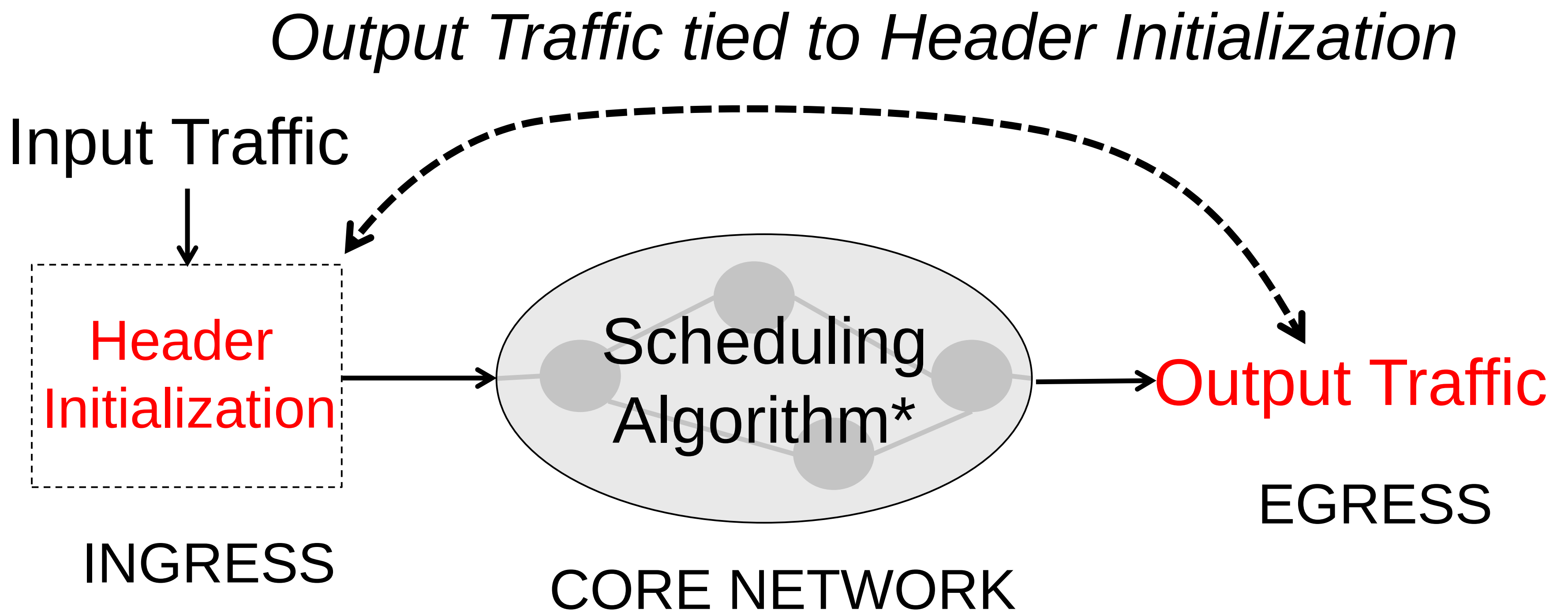


Network Model

Goal: Weighted Fairness

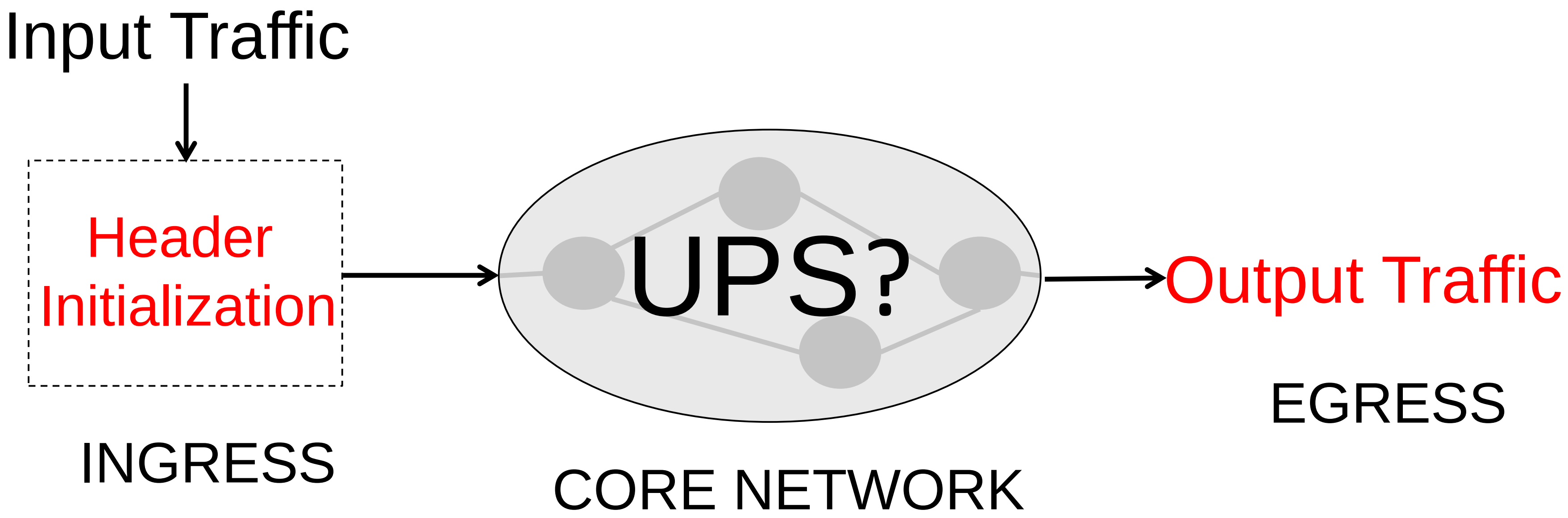


Network Model



* Uses packet header state to make scheduling decis

Network Model



Greater processing capability in the edge than in the core.

As per on prior SDN-based architecture designs.

How do we formally define and evaluate a UPS?

Defining a UPS

Theoretical Viewpoint:

Can it replay a given schedule?

Practical Viewpoint:

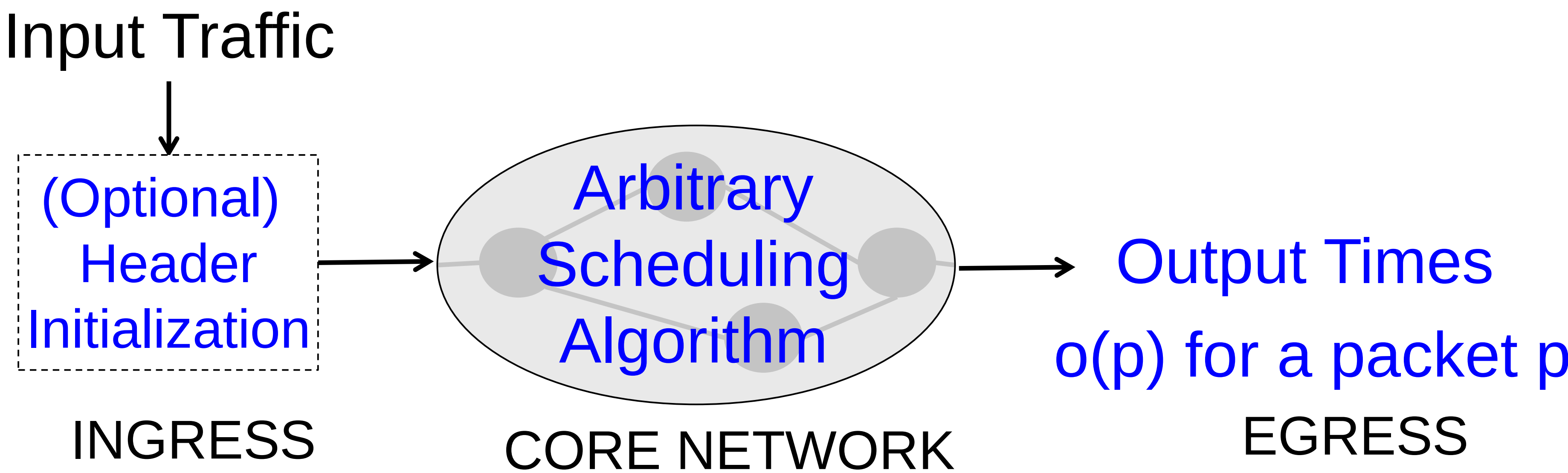
Can it achieve a given objective?

Theoretical Viewpoint

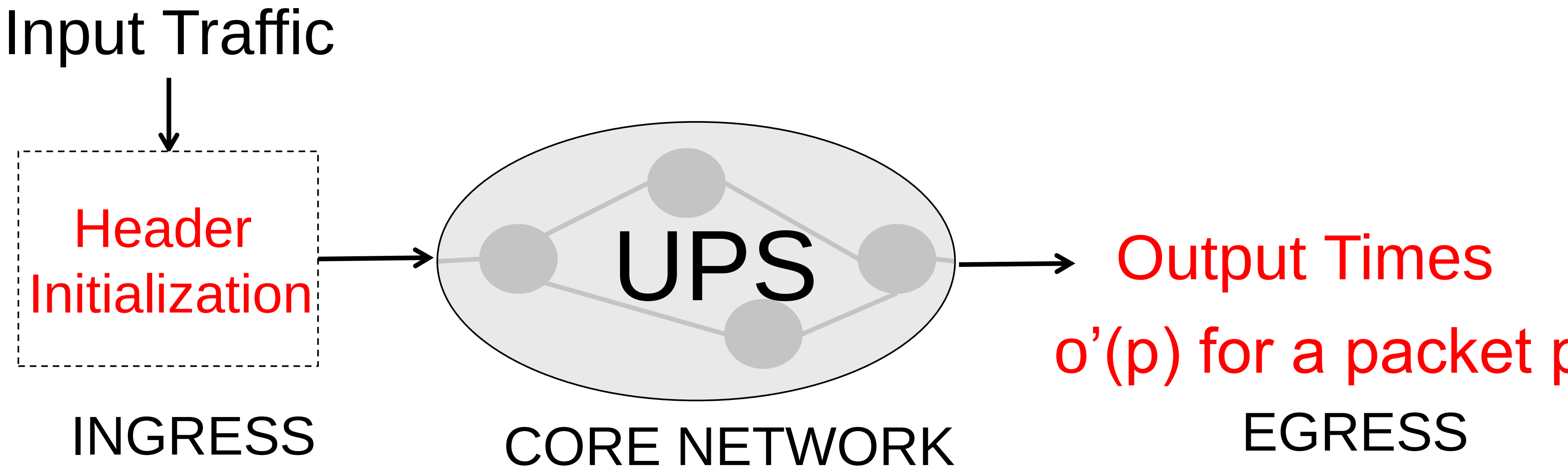
Can it replay a given schedule?

Original Schedule

Only requirement from original schedule:
Output Times are viable

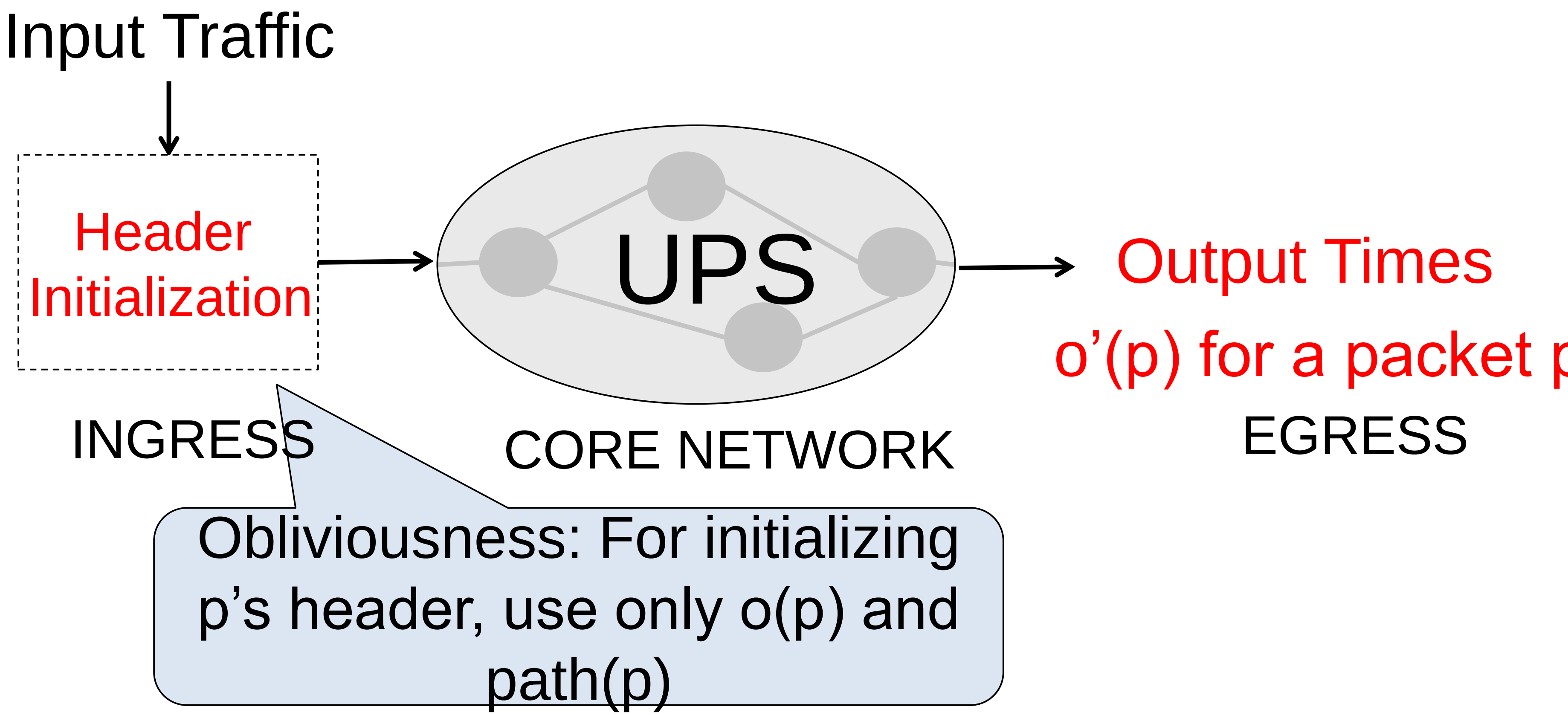


Replaying the Schedule, given o(p)

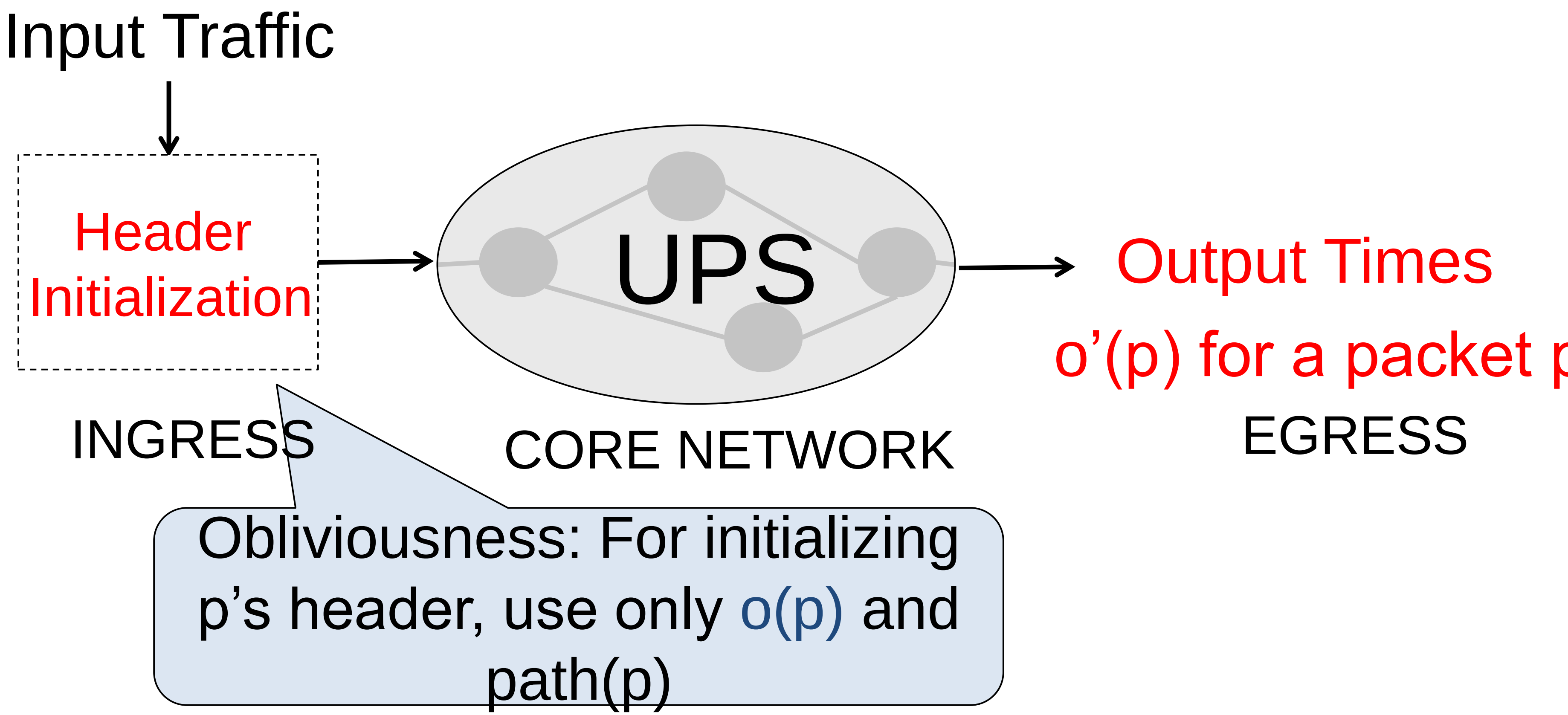


For every packet p, o'(p) ≤ o(p)

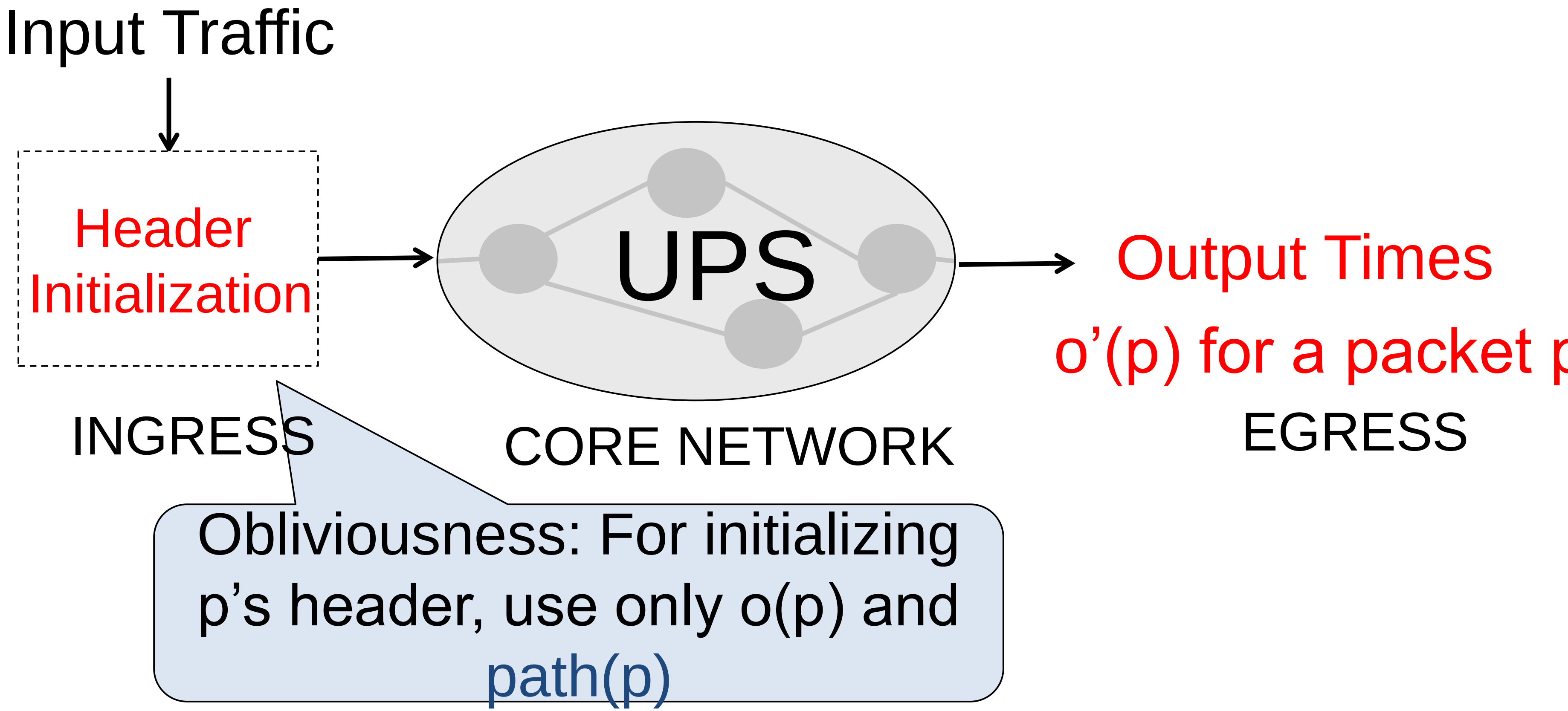
Pragmatic Constraints on a UPS



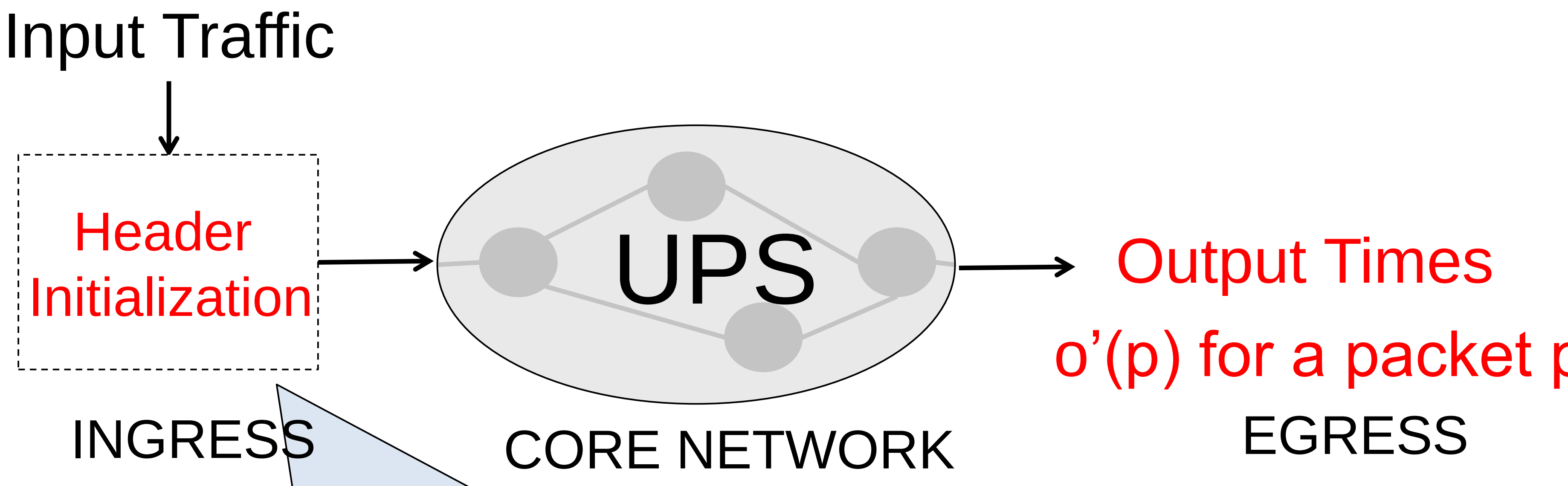
Pragmatic Constraints on a UPS



Pragmatic Constraints on a UPS

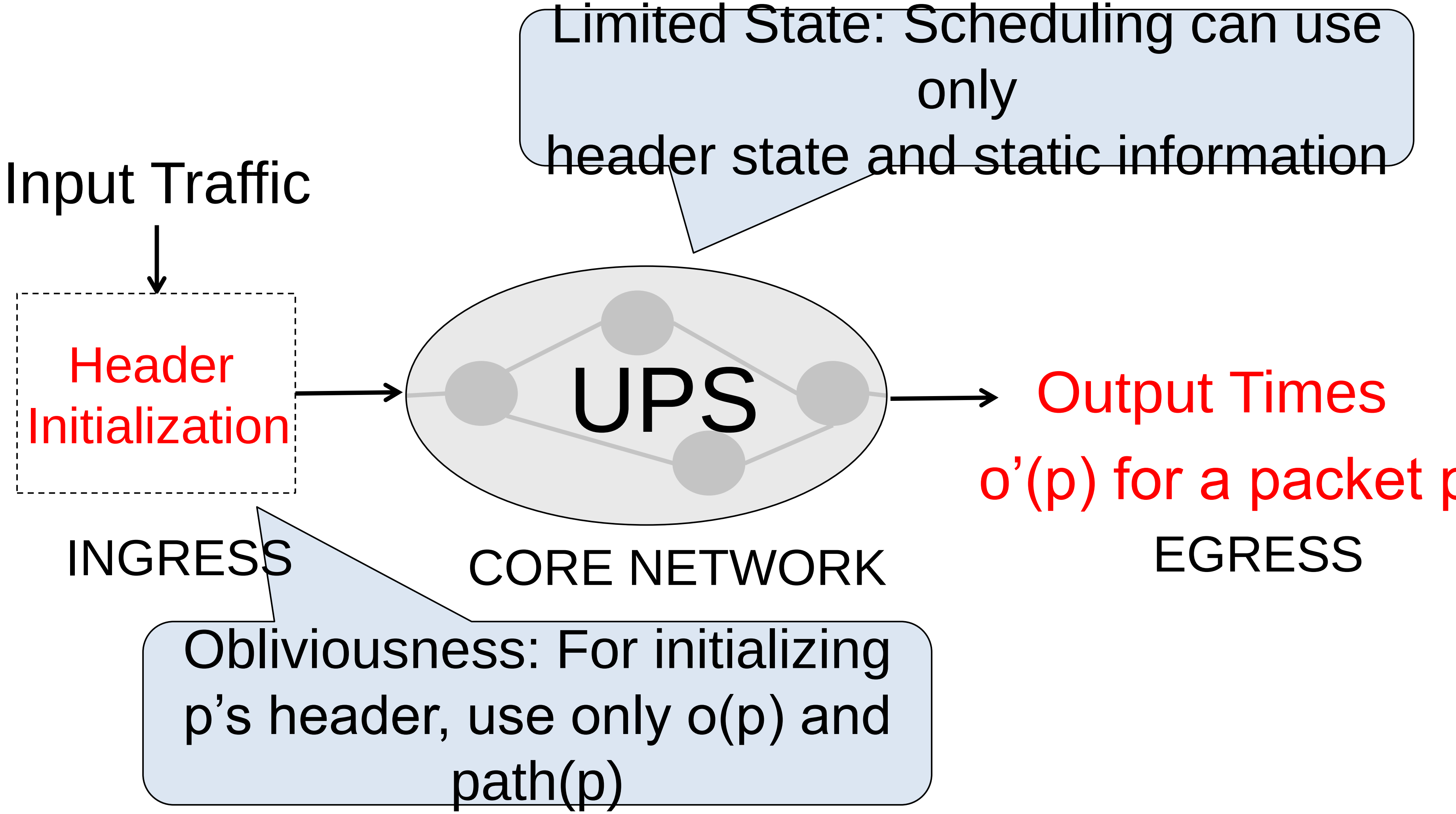


Pragmatic Constraints on a UPS

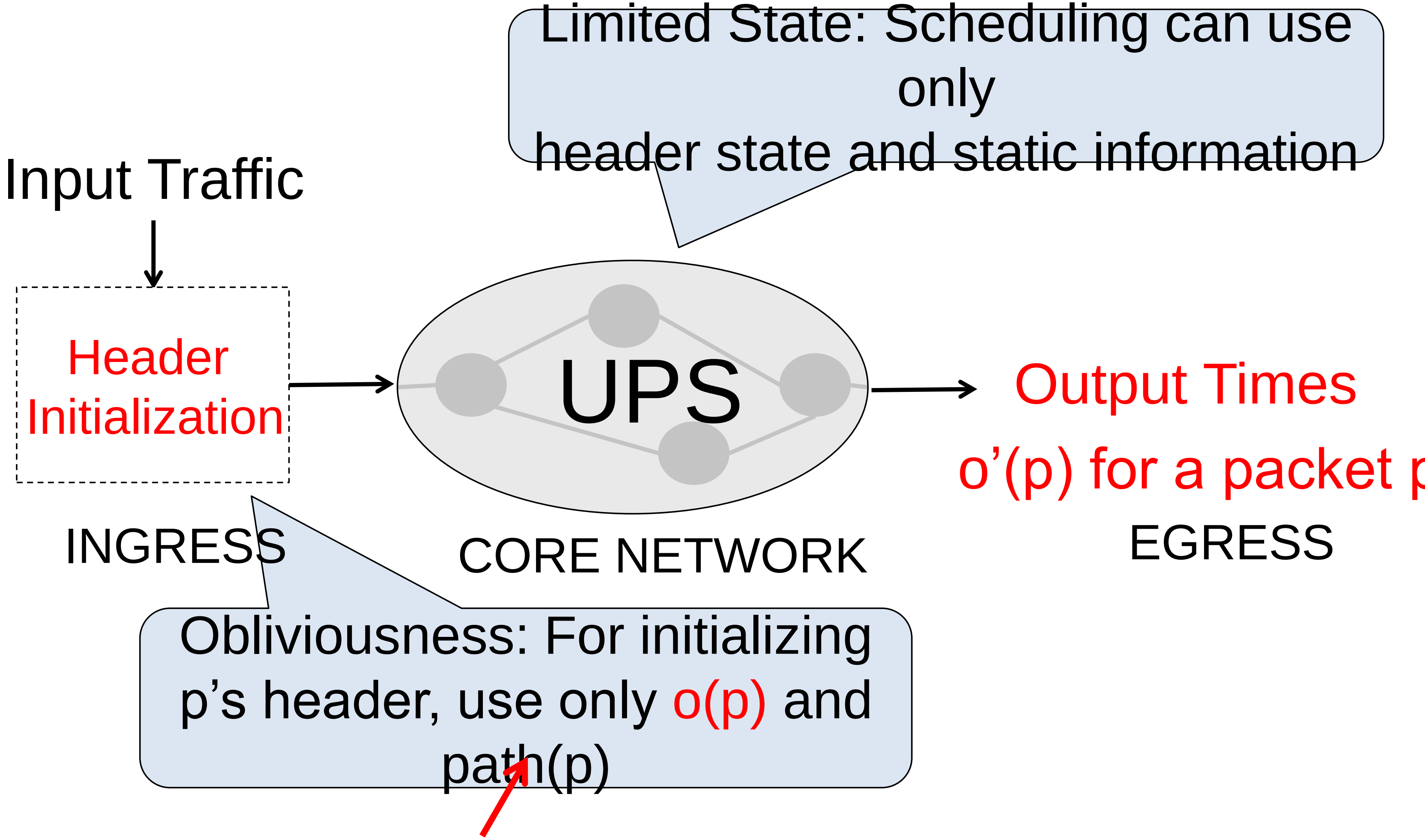


Obliviousness: For initializing p's header, use only o(p) and path(p)

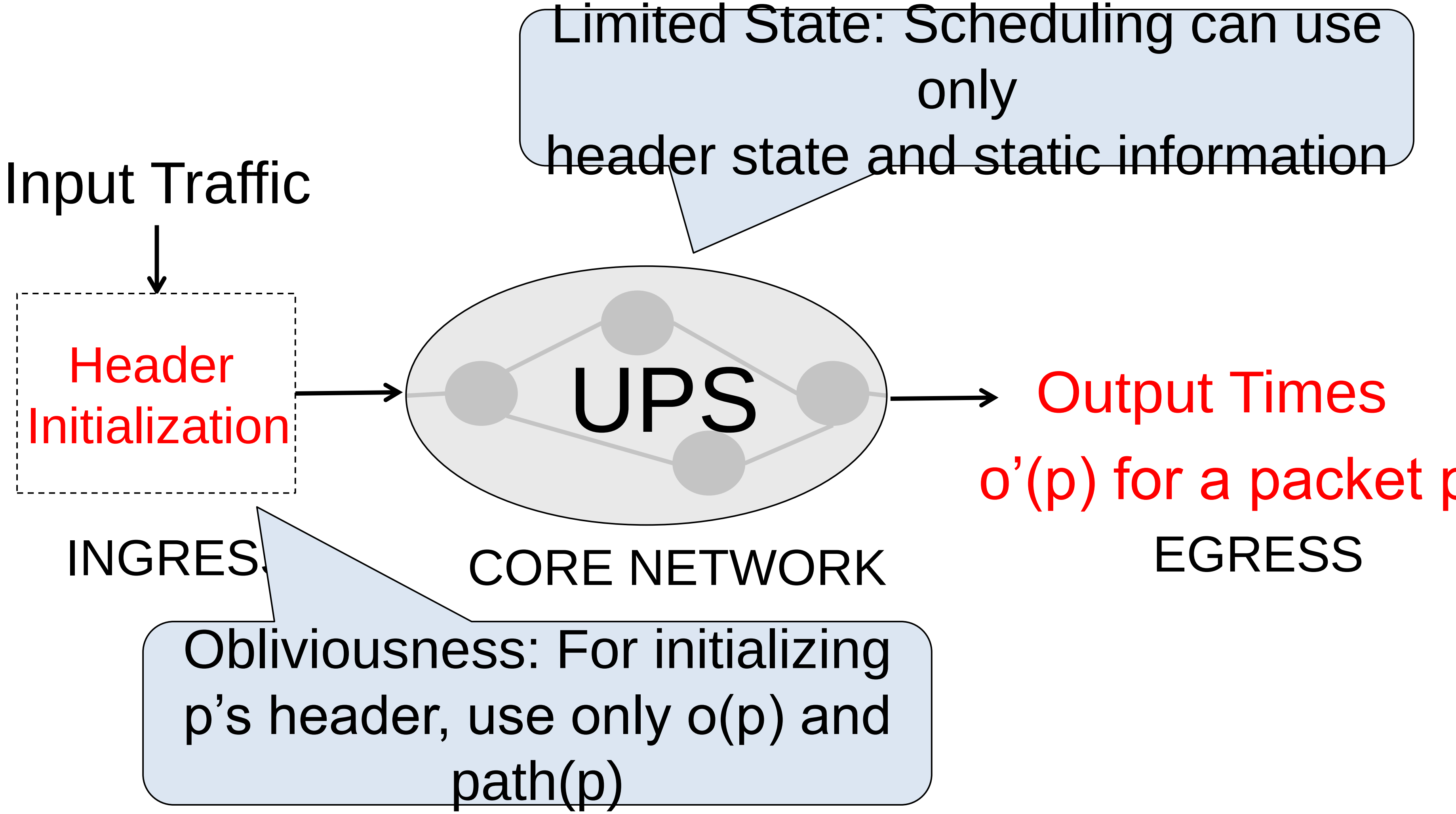
Pragmatic Constraints on a UPS



Pragmatic Constraints on a UPS



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

How close can we get to a UPS?

Key Result: Depends on congestion points

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

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	✗	✗

Not all algorithms achieve upper bound

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


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

Practical Viewpoint

Can it achieve a given objective?

Achieving various network objectives

- Slack assignment based on heuristics
- Three objective functions
  - Tail packet delays
  - Mean Flow Completion Time
  - Fairness
- We also show how LSTF can facilitate AQM from the edge.
- *See NSDI'16 paper for details!*

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 slack initializations.

There are still some interesting 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?
- 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.ed

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

Thank You!