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

We are asking a new question…..

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

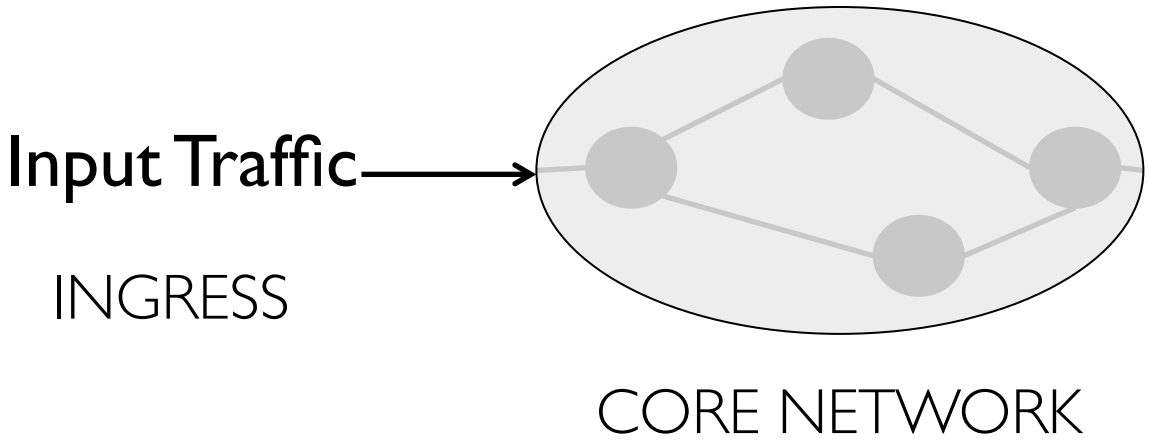
Is there a *universal* packet scheduling algorithm?

UPS: Universal Packet Scheduling Algorithm

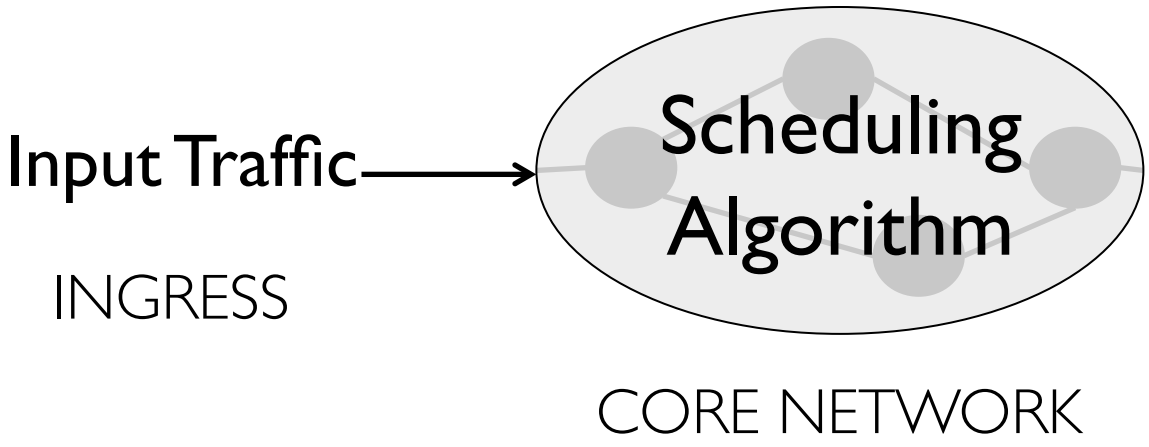
A *single scheduling algorithm* that can imitate the network-wide output produced by ***any*** other algorithm.

How can a single algorithm imitate all others?

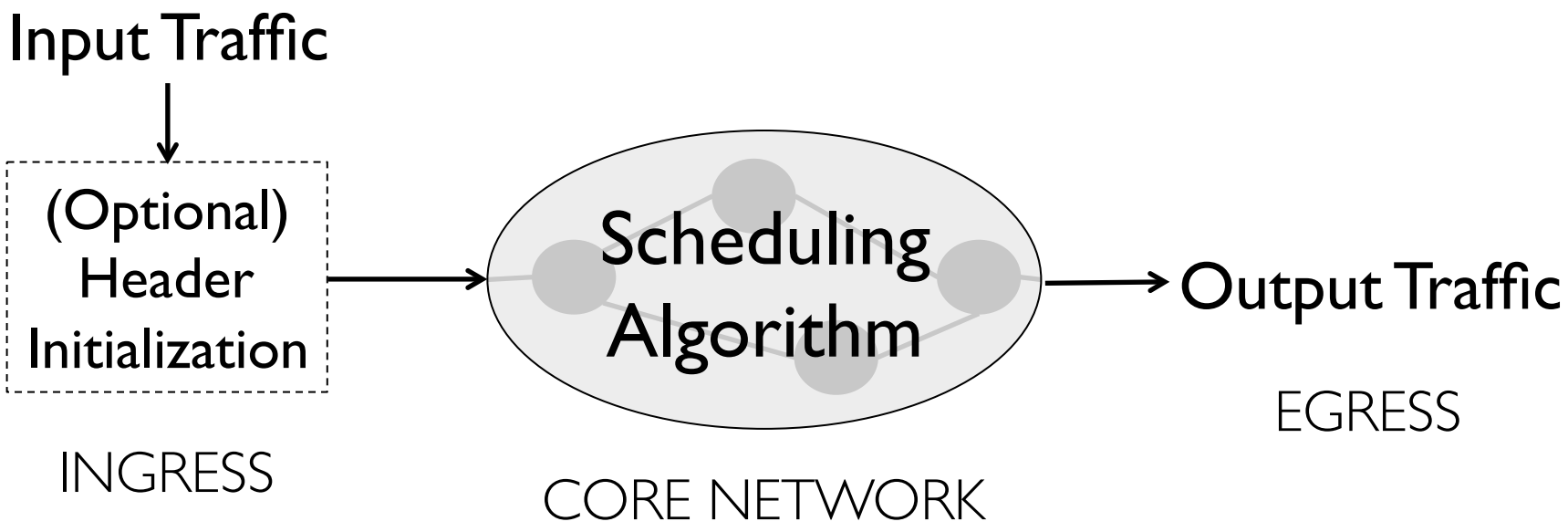
Network Model



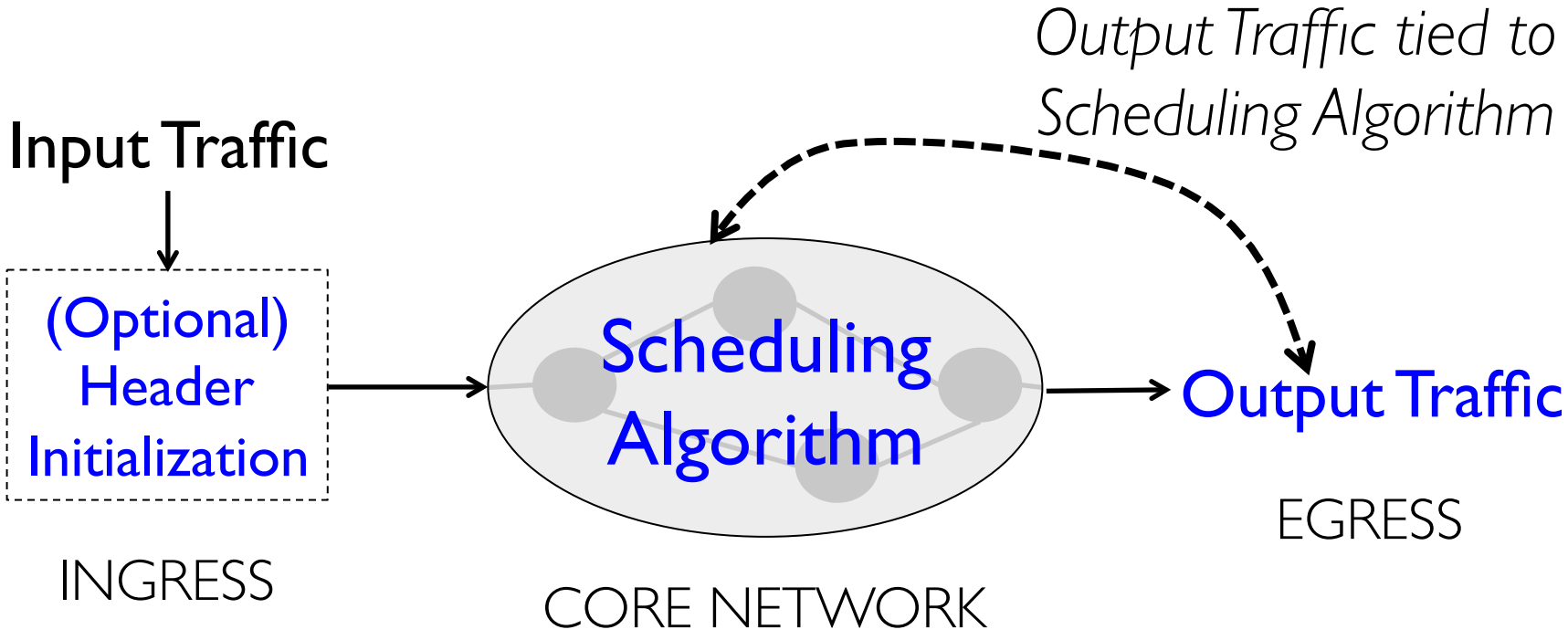
Network Model



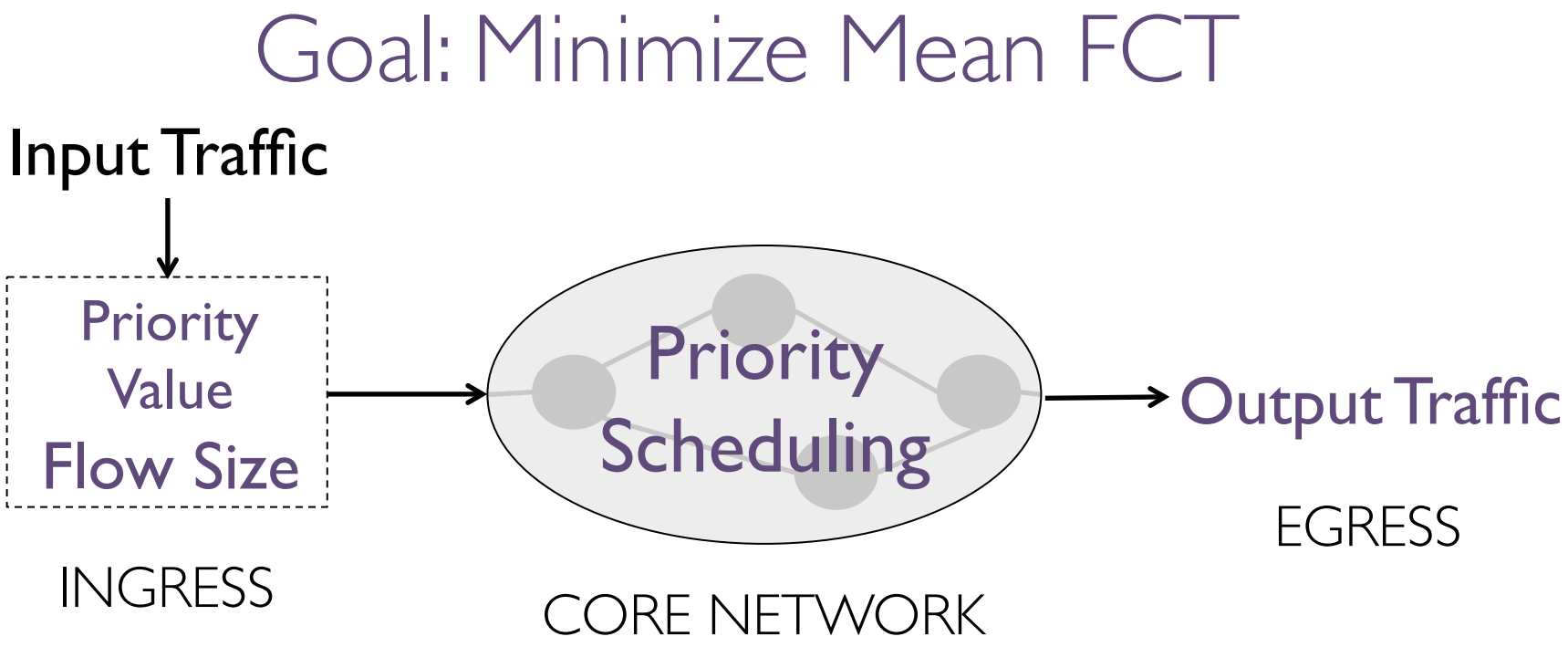
Network Model



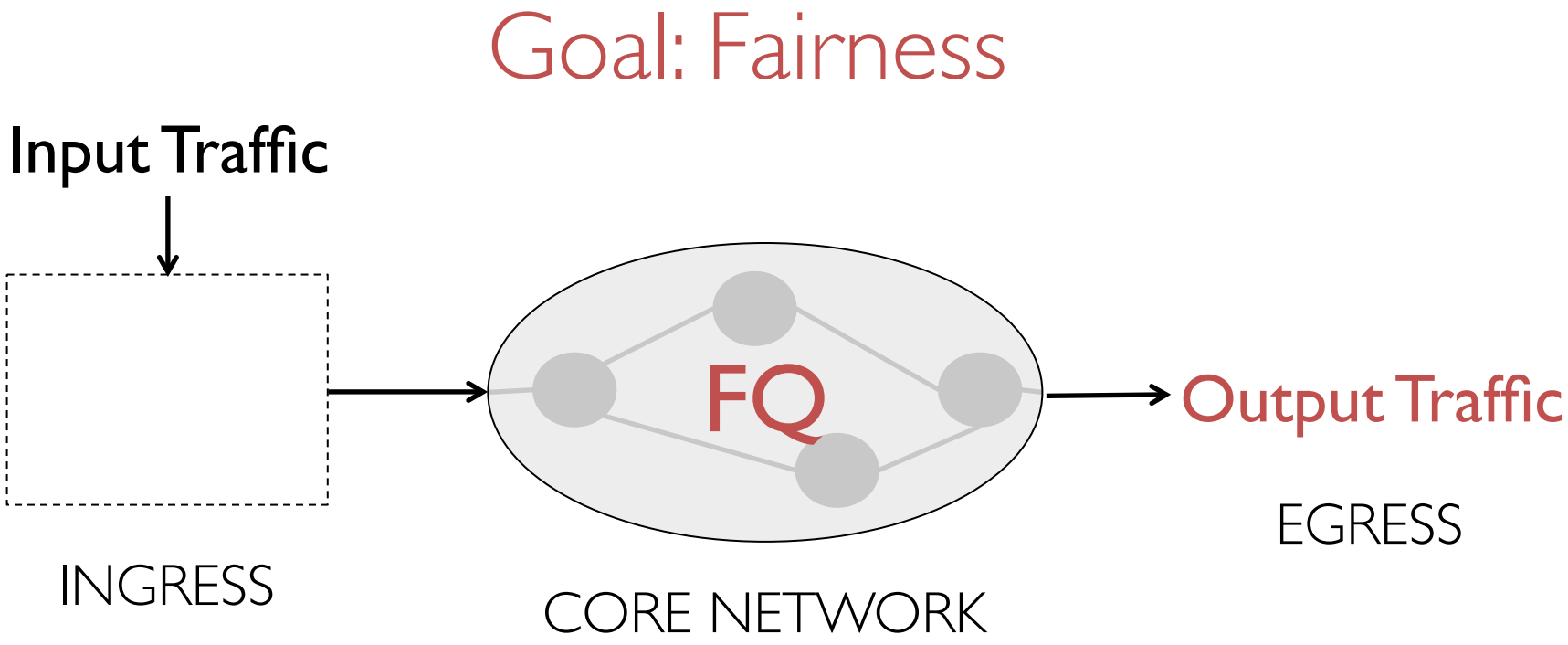
Network Model



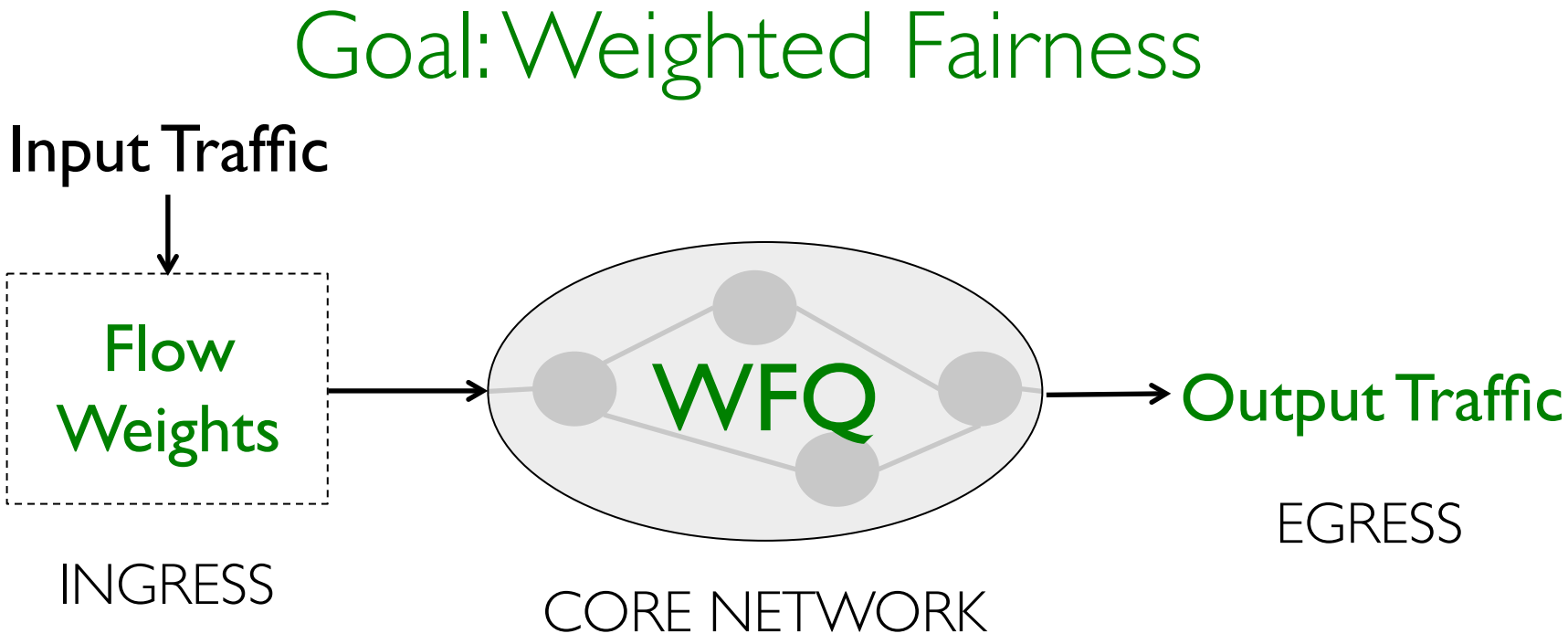
Network Model



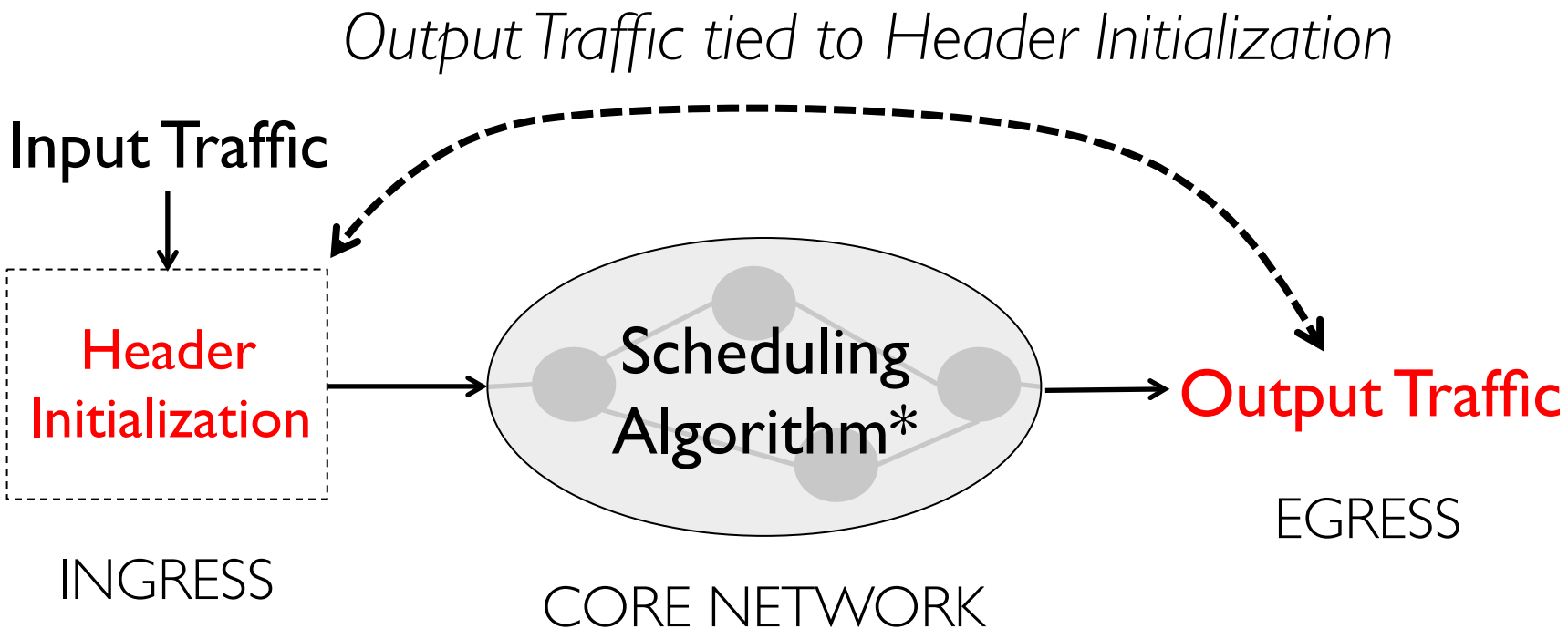
Network Model



Network Model

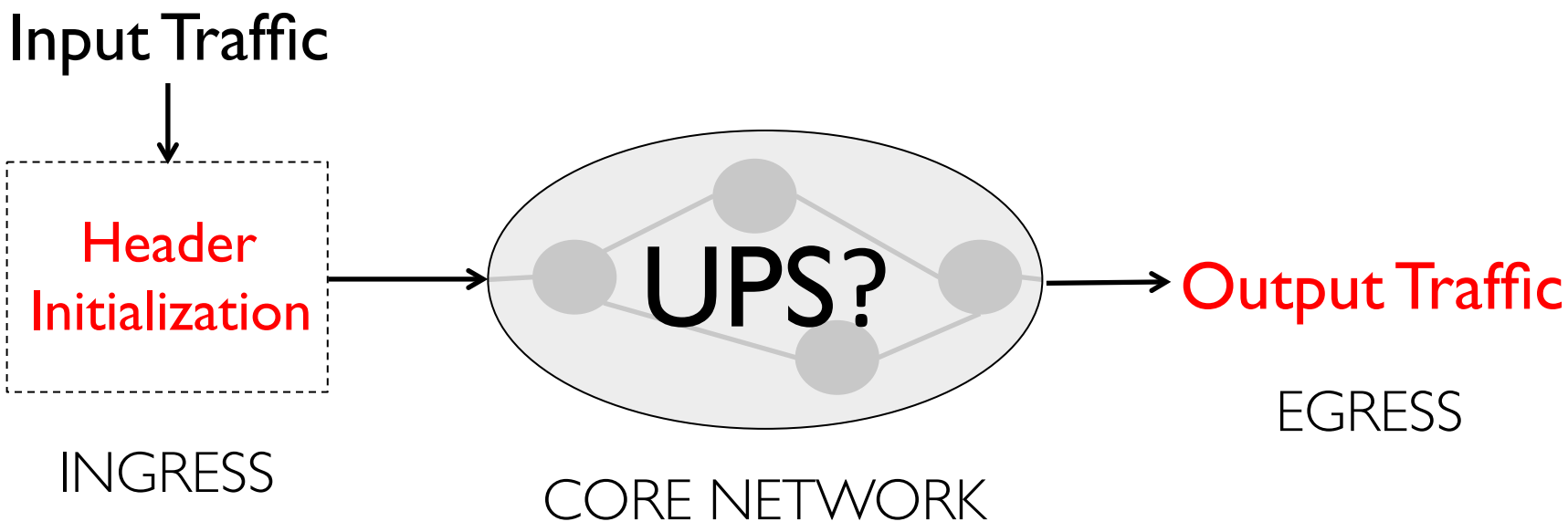


Network Model



* Uses packet header state to make scheduling decisions

Network Model



Network Model

$\{p \in P\}$ $arrive\ time\ i(p)$ $path(p)$

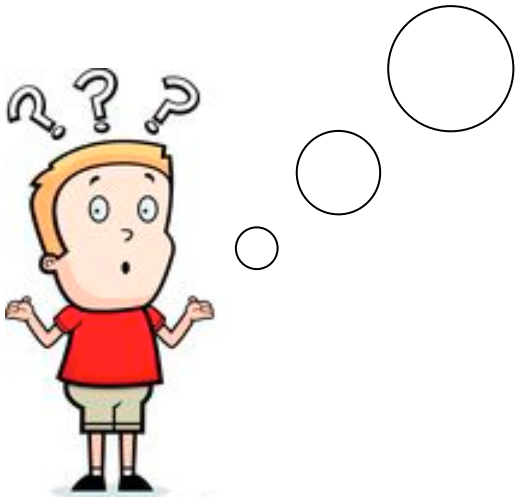
$incoming\ load\ \{(p, i(p), path(p))\}$

$a\ collection\ of\ scheduling\ algorithms\ \{A_\alpha\} (router\ \alpha\ with\ A_\alpha)$

$a\ set\ of\ packet\ output\ times\ \{o(p)\}$

$a\ schedule: \{(p, i(p), o(p))\}$

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?

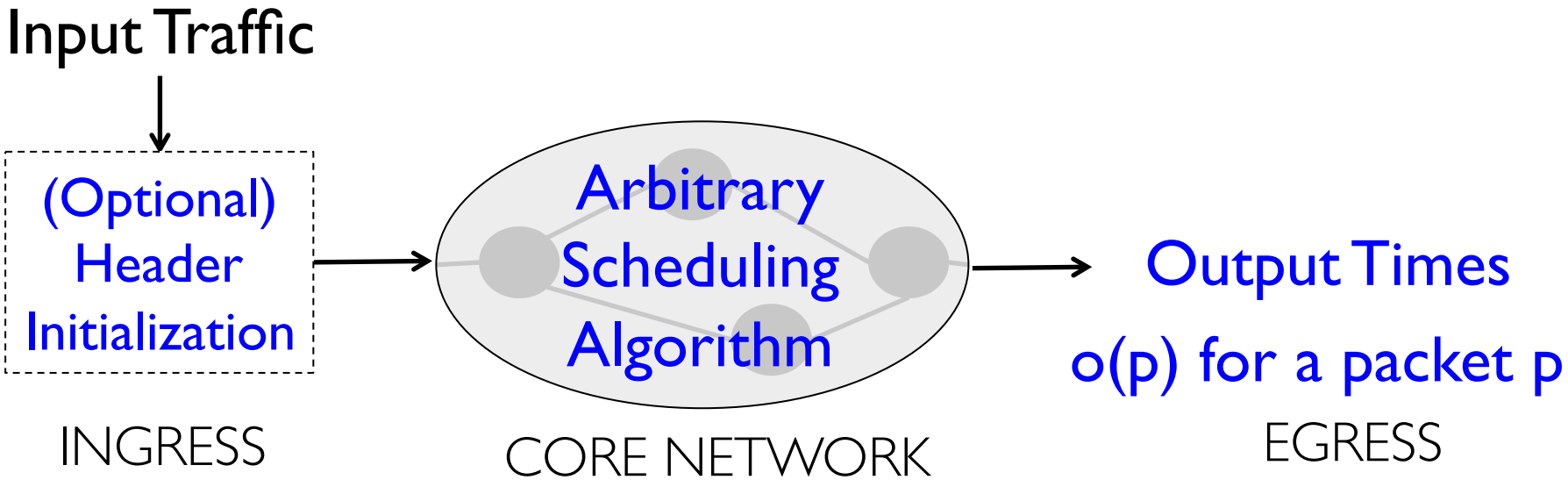
Theoretical Viewpoint

Can it replay a given schedule?

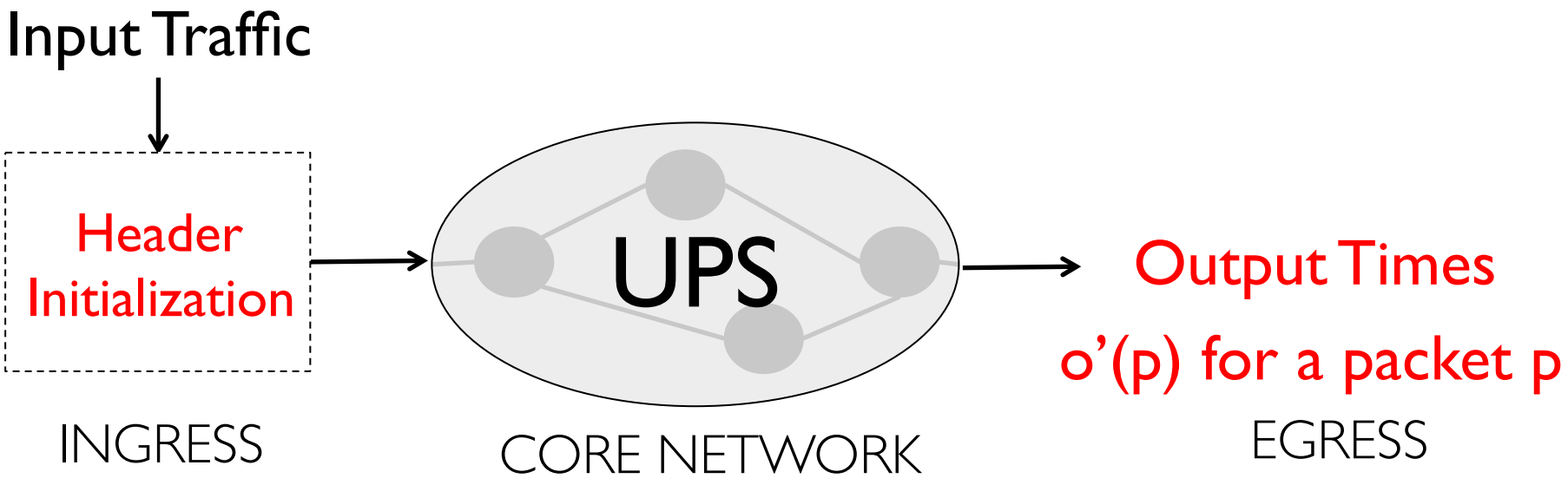
Original Schedule

Only requirement from original schedule:

Output Times are viable

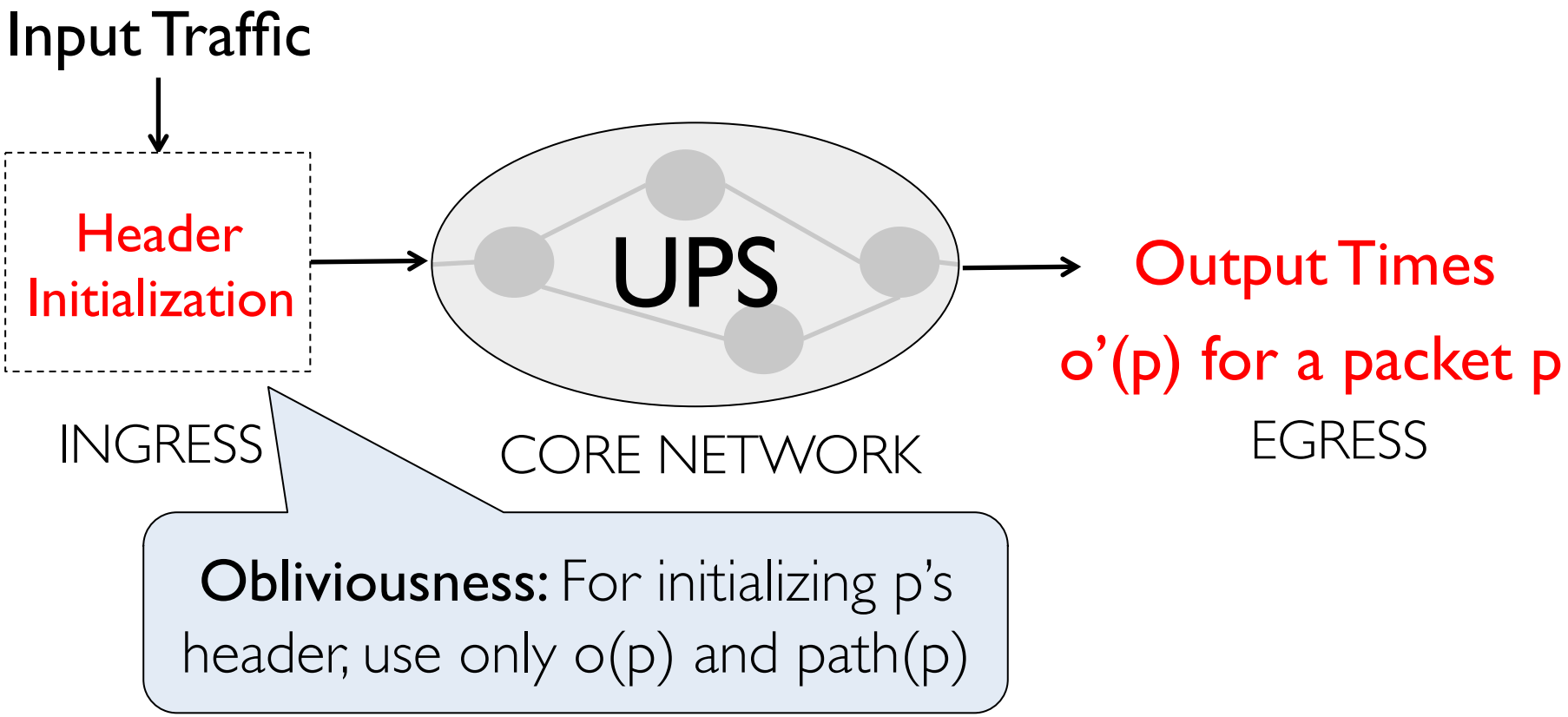


Replaying the Schedule, given o(p)

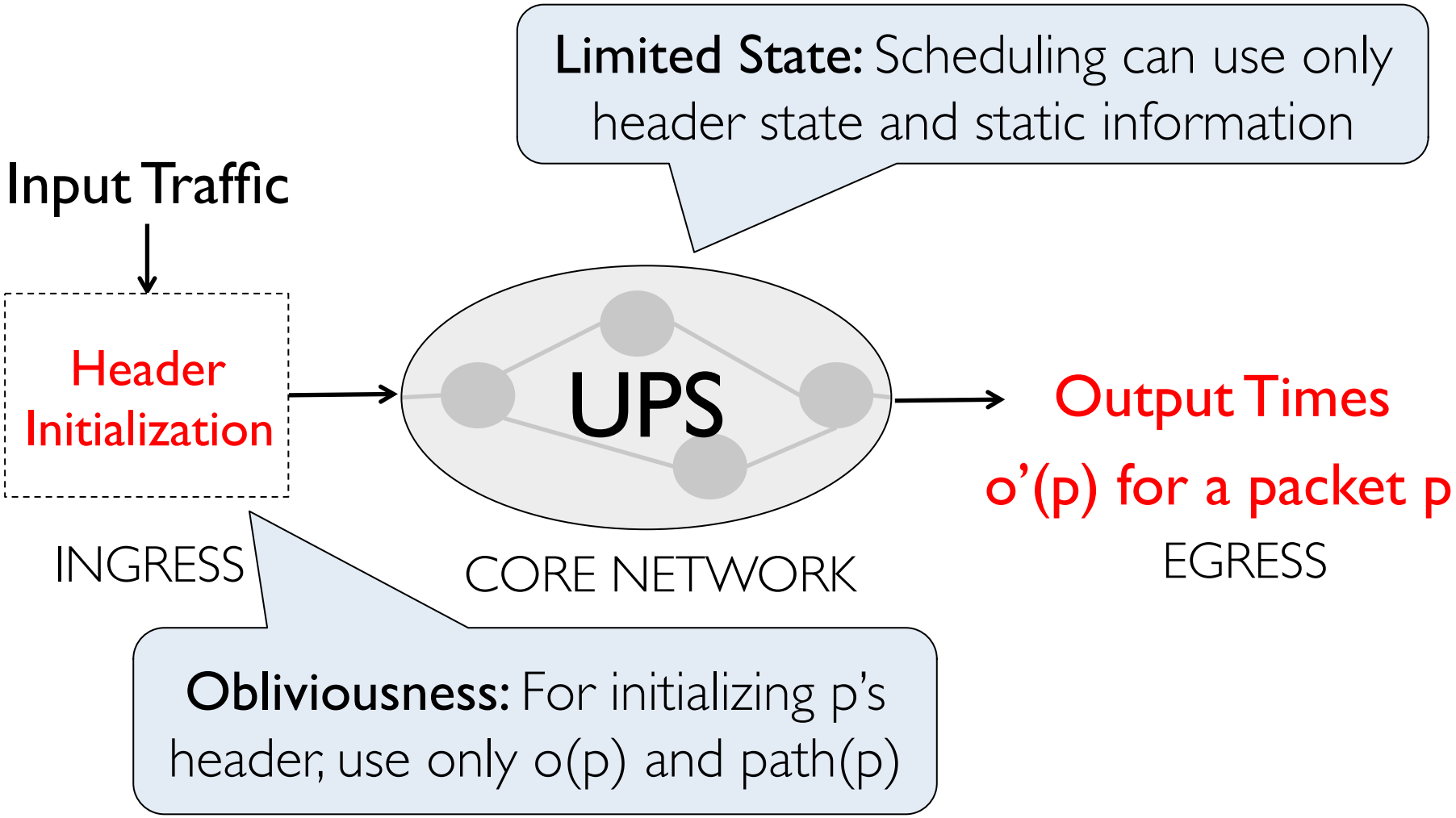


For every packet p, $o'(p) \leq o(p)$

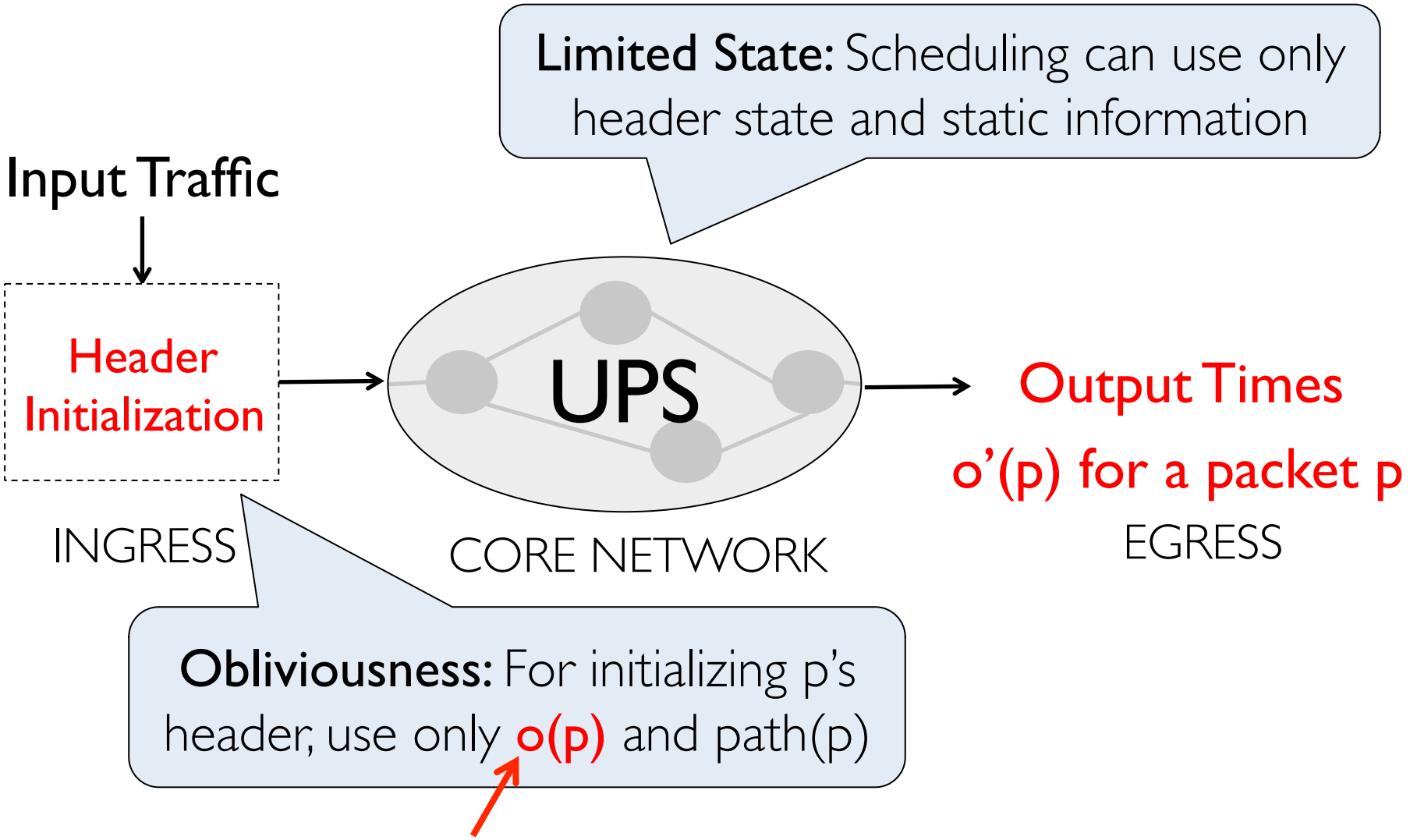
Pragmatic Constraints on a UPS



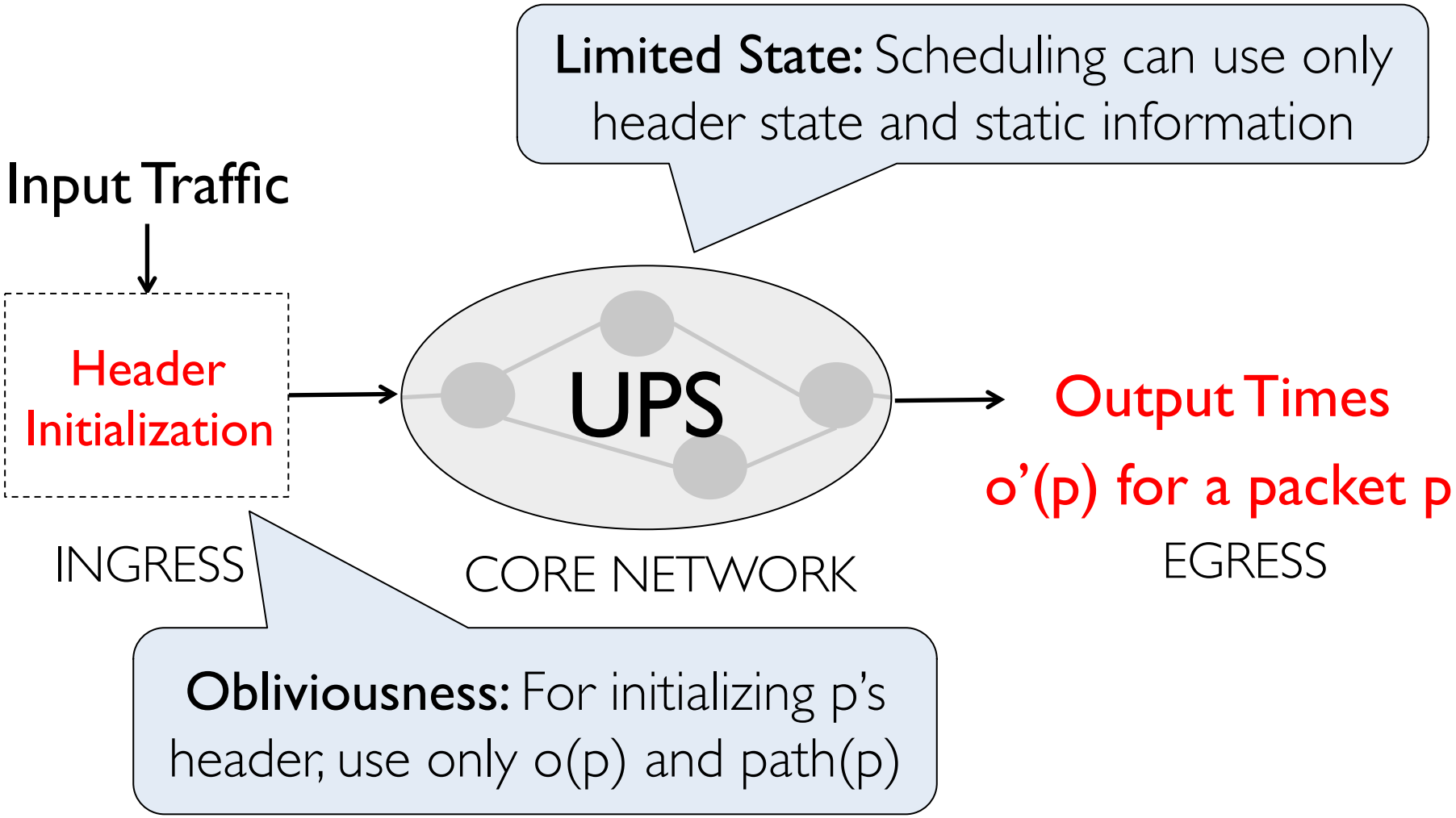
Pragmatic Constraints on a UPS



Pragmatic Constraints on a UPS



We call this Blackbox Initialization



UPS Model

- *Uniformity and Determinism*
  - A UPS must use the same deterministic scheduling logic at every router
- *Limited state used in scheduling decisions*
  - Packet headers
  - Static information about the network topology, link bandwidths, and propagation delays
  - It can modify the header of a packet before forwarding it at every router
- *Limited state used in header initialization*
  - $o(p)$ from the original schedule
  - *path*(p)

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

Topology	Avg. Link Utilization	Scheduling Algorithm	Fraction of packets overdue	
			Total	$> T$
I2 1Gbps-10Gbps	70%	Random	0.0021	0.0002
I2 1Gbps-10Gbps	10% 30% 50% 90%	Random	0.0007 0.0281 0.0221 0.0008	0.0 0.0017 0.0002 4×10^{-6}
I2 1Gbps-1Gbps I2 10Gbps-10Gbps I2 / 10	70%	Random	0.0204 0.0631 0.0127	8×10^{-6} 0.0448 0.00001
Rocketfuel Datacenter	70%	Random	0.0246 0.0164	0.0063 0.0154
I2 1Gbps-10Gbps	70%	FIFO FQ SJF LIFO FQ/FIFO+ FQ: SJF/FIFO	0.0143 0.0271 0.1833 0.1477 0.0152 0.0297	0.0006 0.0002 0.0019 0.0067 0.0004 0.0003

Table 1: LSTF replay performance across various scenarios. T represents the transmission time at the bottleneck link.

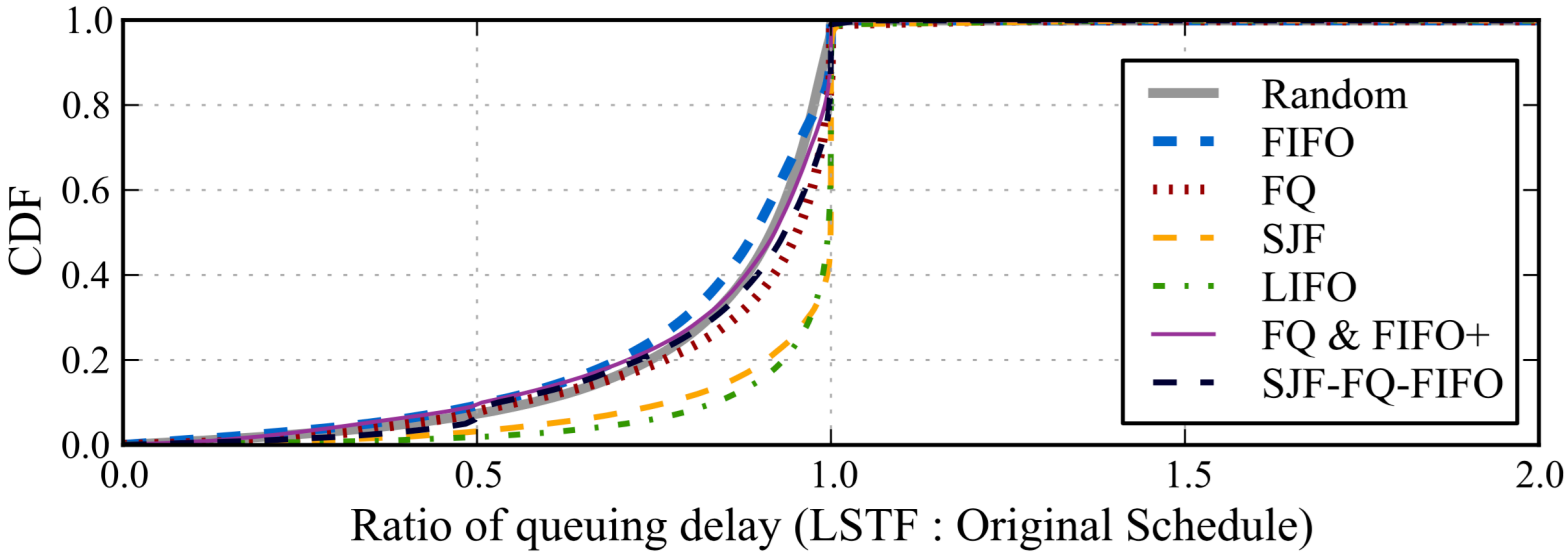


Figure 1: Ratio of queuing delay with varying packet scheduling algorithms, on I2 1Gbps-10Gbps topology at 70% utilization.

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

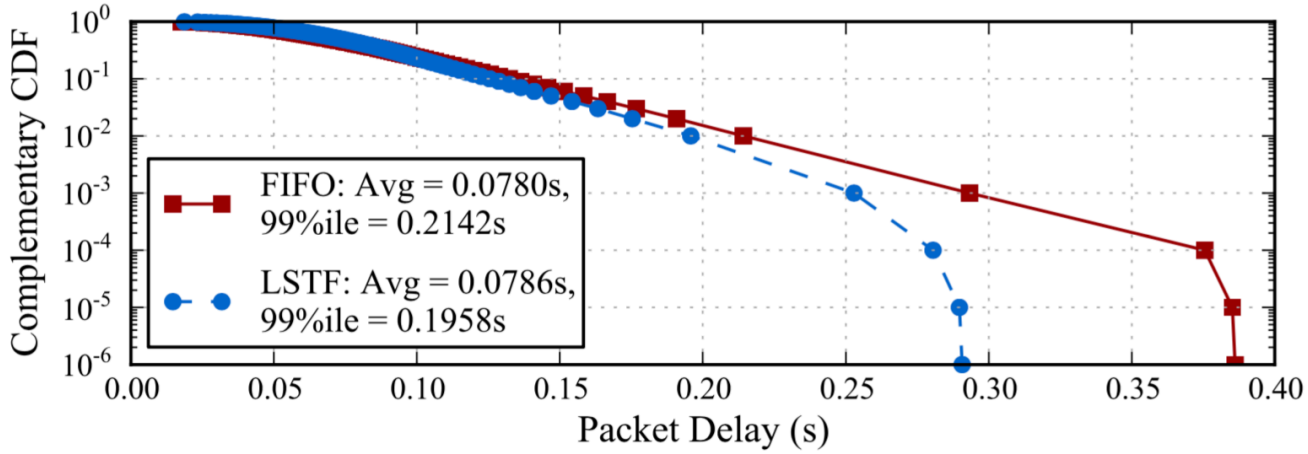
Tail Packet Delays

Slack Assignment: Same slack for all packets

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

Results:

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



Expt. Setup	Avg Delay (s)		99%ile Delay (s)	
	FIFO	LSTF	FIFO	LSTF
I2 1Gbps-10Gbps at 30% util.	0.0411	0.0411	0.0911	0.0868
I2 1Gbps-10Gbps at 50% util.	0.0516	0.0517	0.1288	0.1195
I2 1Gbps-10Gbps at 70% util.	0.0780	0.0786	0.2142	0.1958
I2 1Gbps-1Gbps at 70% util.	0.0771	0.0771	0.2163	0.216
I2 / 10 at 70% util.	0.5762	0.5765	1.9393	1.9367
Rocketfuel at 70% util.	0.1891	0.1883	3.8139	3.7199
Datacenter at 70% util.	0.0250	0.0240	0.1352	0.1100

Figure 3: Tail packet delays for LSTF compared to FIFO. The graph shows the complementary CDF of packet delays for the I2 1Gbps-10Gbps topology at 70% utilization with the average and 99%ile packet delay values indicated in the legend. The table shows the corresponding results for varying settings.

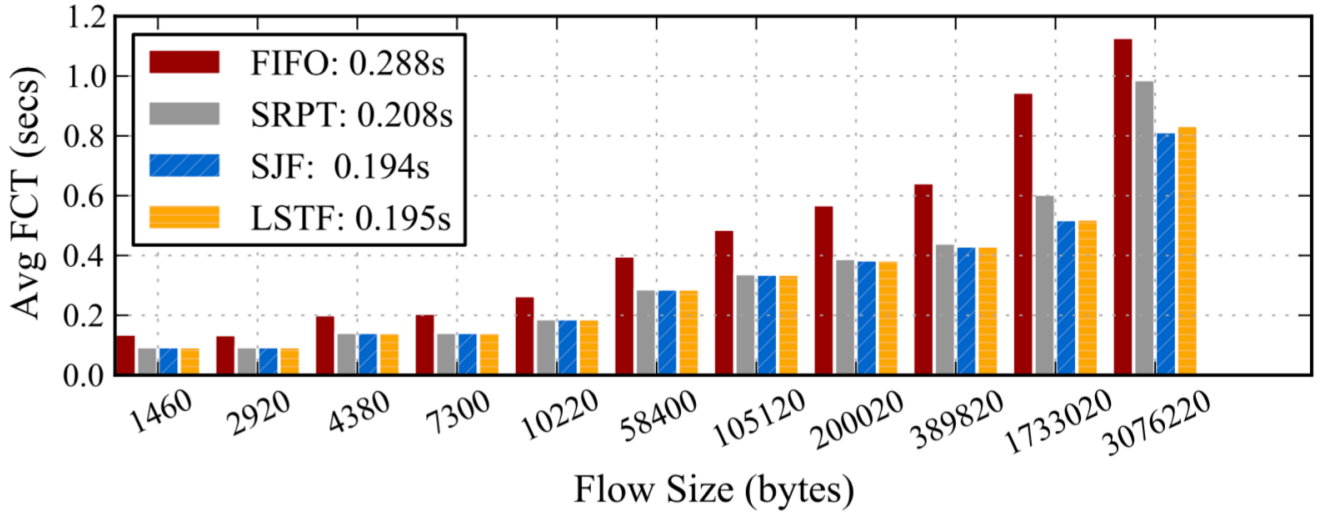
Mean Flow Completion Time

Slack Assignment: Proportional to flow size

State-of-the-art: SJF, SRPT

Results:

- Mean FCTs comparable to both SJF and SRPT.



Expt. Setup	Avg FCT (s)			
	FIFO	SRPT	SJF	LSTF
I2 1Gbps-10Gbps at 30% util.	0.189	0.183	0.182	0.182
I2 1Gbps-10Gbps at 50% util.	0.212	0.189	0.185	0.185
I2 1Gbps-10Gbps at 70% util.	0.288	0.208	0.194	0.195
I2 1Gbps-1Gbps at 70% util.	0.252	0.209	0.202	0.202
I2 / 10 at 70% util.	0.899	0.658	0.620	0.621
Rocketfuel at 70% util.	0.305	0.240	0.228	0.228
Datacenter at 70% util.	0.058	0.018	0.016	0.015

Figure 2: The graph shows the average FCT bucketed by flow size obtained with FIFO, SRPT and SJF (using priorities and LSTF) for I2 1Gbps-10Gbps at 70% utilization. The legend indicates the average FCT across all flows. The table indicates the average FCTs for varying settings.

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})

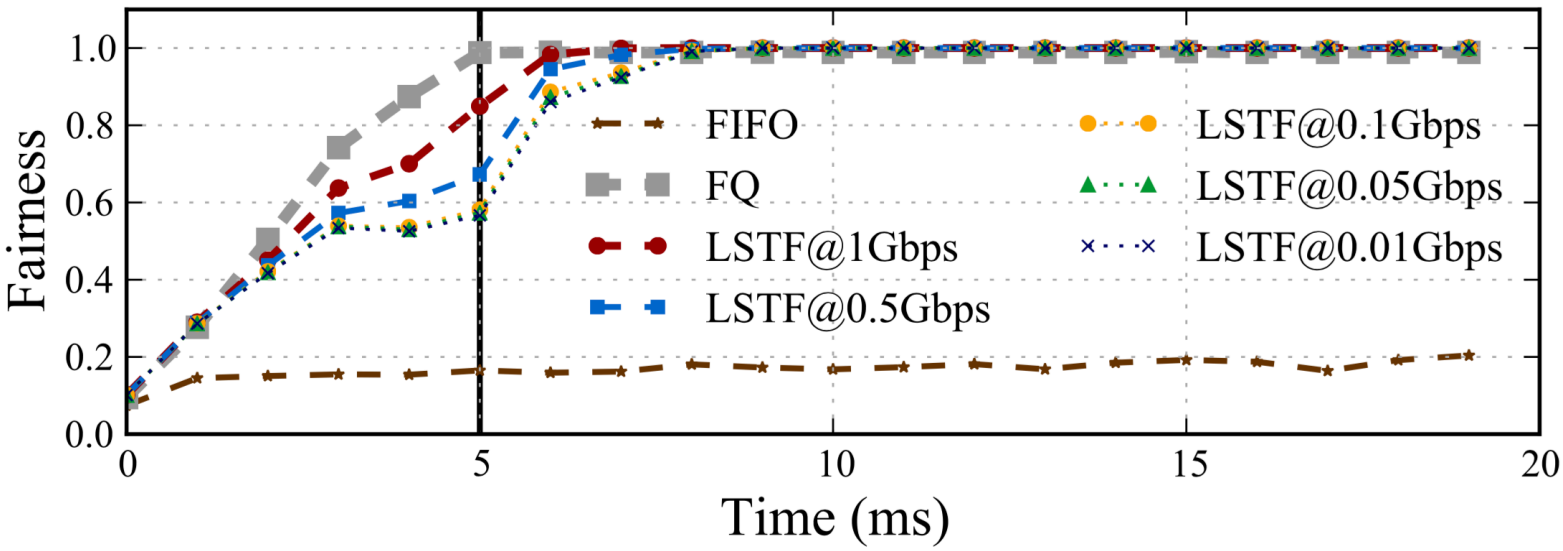


Figure 4: Fairness for long-lived flows on Internet2 topology. The legend indicates the value of r_{est} used for LSTF slack initialization.

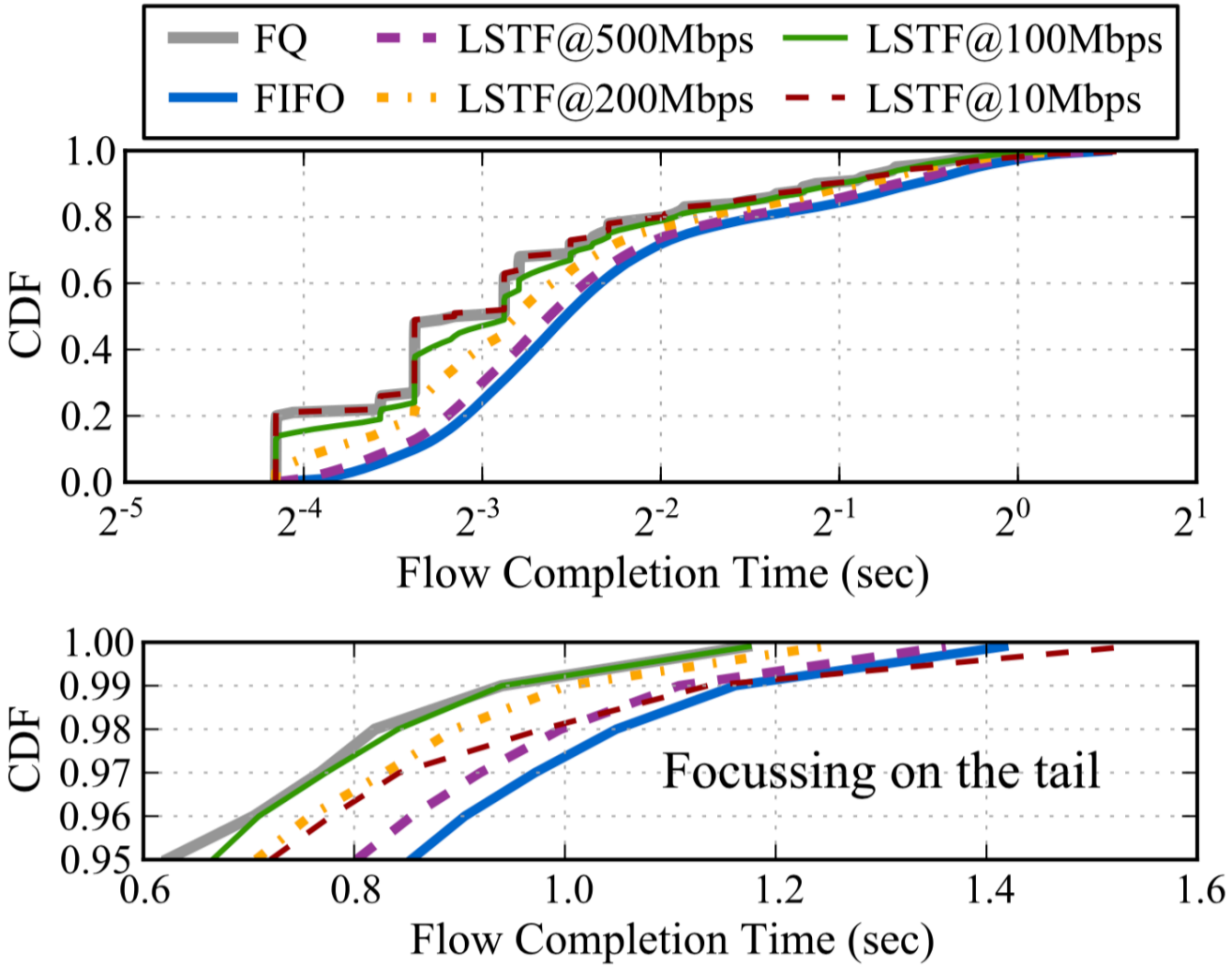
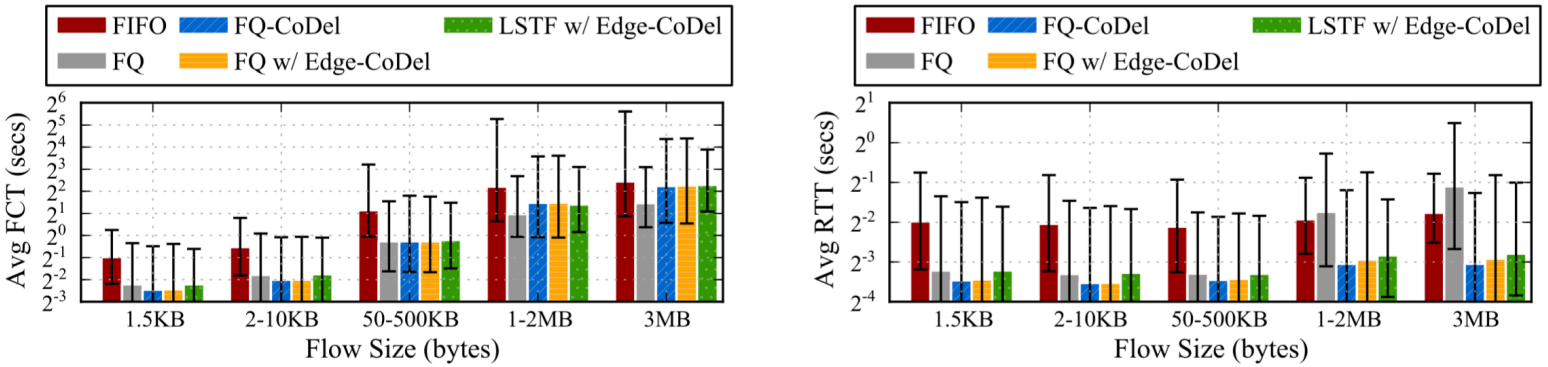


Figure 5: CDF of FCTs for the I2 1Gbps-10Gbps topology at 70% utilization.

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).



Expt. Setup	r_{est} (Mbps)	Avg FCT across bytes (s)					Avg RTT across bytes (s)				
		FIFO	FQ	FQ-CoDel	FQ w/ Edge-CoDel	LSTF w/ Edge-CoDel	FIFO	FQ	FQ-CoDel	FQ w/ Edge-CoDel	LSTF w/ Edge-CoDel
I2 1Gbps-10Gbps at 70% util.	100	0.811	0.622	0.642	0.633	0.641	0.0756	0.0733	0.0642	0.0646	0.0661
I2 1Gbps-1Gbps at 70% util.	100	0.766	0.630	0.642	0.637	0.658	0.0716	0.0702	0.0639	0.0643	0.0666
I2 / 10 at 30% util.	40	0.918	0.836	0.897	0.887	0.907	0.0998	0.1085	0.0792	0.0798	0.0826
I2 / 10 at 50% util.	30	1.706	1.214	1.430	1.369	1.427	0.1384	0.1752	0.0901	0.0918	0.1001
I2 / 10 at 70% util.	10	4.837	2.295	3.687	3.738	3.739	0.2779	0.3752	0.1182	0.1281	0.1388
I2 / 10, half RTTs at 70% util.	10	4.569	2.023	3.196	3.245	3.405	0.2555	0.3607	0.0995	0.1131	0.1165
I2 / 10, double RTTs at 70% util.	10	5.098	2.769	4.243	4.125	4.389	0.325	0.4172	0.1591	0.1640	0.1843
Rocketfuel at 70% util.	100	0.964	0.796	0.840	0.813	0.835	0.0922	0.0991	0.0794	0.0788	0.0836

Figure 6: The figures show the average FCT and RTT values for I2 / 10 at 70% utilization (LSTF uses fairness slack assignment with $r_{est} = 10Mbps$). The error bars indicate the 10^{th} and the 99^{th} percentile values and the y-axis is in log-scale. The table indicates the average FCT and RTTs (across bytes) for varying settings.

Util.	Avg FCT (s)			Avg RTT (ms)		
	FIFO w/ No ECN	FIFO w/ ECN	LSTF w/ Edge-ECN	FIFO w/ No ECN	FIFO w/ ECN	LSTF w/ Edge-ECN
30%	0.0020	0.0011	0.0011	0.2069	0.1123	0.1077
50%	0.0219	0.0086	0.0079	0.3425	0.1601	0.1477
70%	0.0501	0.0241	0.0240	0.4497	0.2616	0.2494

Table 3: DCTCP performance with no ECN, ECN (in-switch) and Edge-ECN for the datacenter topology at varying utilizations.

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!