Cebinae:

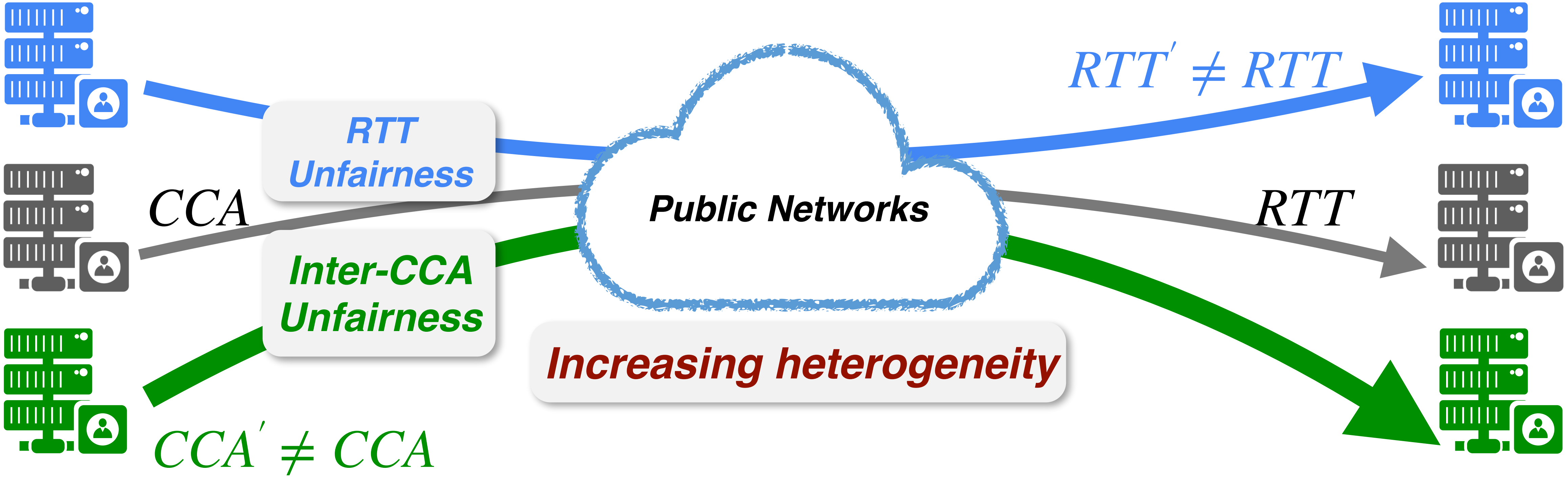
Scalable In-network Fairness Augmentation

Liangcheng Yu, *John Sonchack, Vincent Liu*

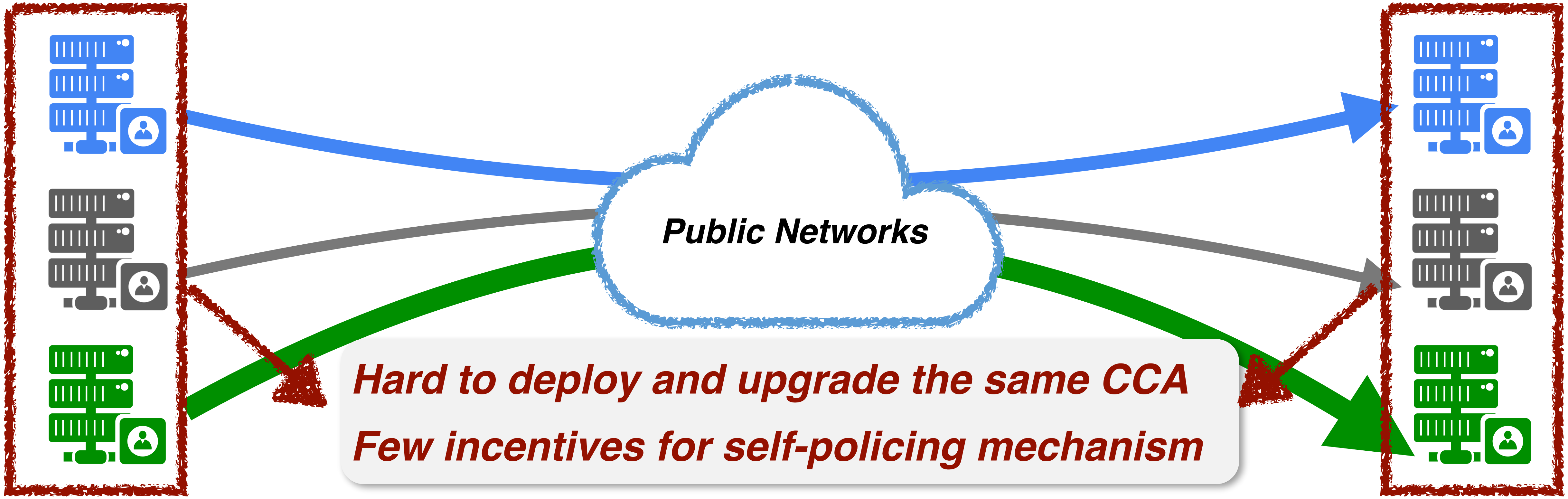
Penn
University of Pennsylvania

Princeton University

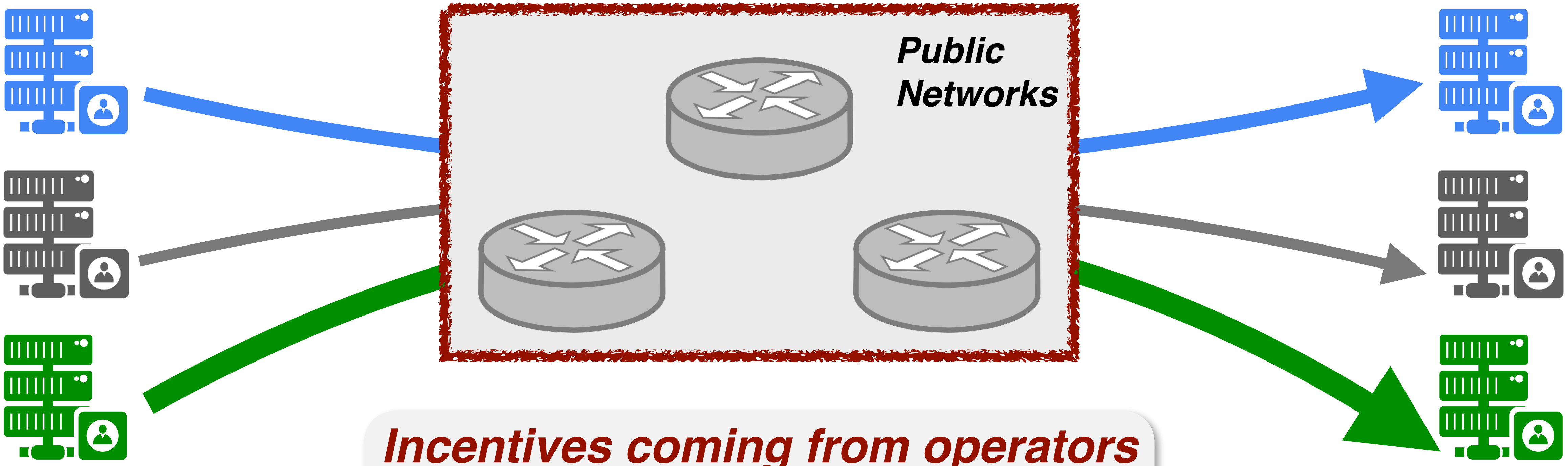
Public networks care about fairness



Fairness enforcement at the end hosts?



In-network fairness enforcement



Incentives coming from operators of the in-network devices

In-network fairness enforcement

- Existing approaches suffer from limited practicalities
  - ***Assumption***: *specialized hardware for per-flow queues, end-host cooperation…*

In-network fairness enforcement

- Existing approaches suffer from limited practicalities
  - ***Assumption***: *specialized hardware for per-flow queues, end-host cooperation…*

- AFQ [NSDI ’18]: practical emulation of ideal FQ on COTS hardware
  - ***Constraints***: *e.g., # priorities, queues, buffers*

In-network fairness enforcement

- Existing approaches suffer from limited practicalities
  - ***Assumption***: *specialized hardware for per-flow queues, end-host cooperation…*

- AFQ [NSDI ’18]: practical emulation of ideal FQ on COTS hardware
  - ***Constraints***: *e.g., # priorities, queues, buffers*

*Challenging to **strictly** enforce FQ on **each individual flow***

Cebinae: a simpler approach

- Relaxation of fairness **at every instance in time**
  - *Penalize/redistribute BW from flows exceeding fair share to others*

Cebinae: a simpler approach

- Relaxation of fairness **at every instance in time**
    - *Penalize/redistribute BW from flows exceeding fair share to others*
- **Binary classification** of flows
    - *Efficiently implement various subroutines (e.g., leaky-bucket filter)*

Cebinae: a simpler approach

- Relaxation of fairness **at every instance in time**
  - *Penalize/redistribute BW from flows exceeding fair share to others*
- **Binary classification** of flows
  - *Efficiently implement various subroutines (e.g., leaky-bucket filter)*

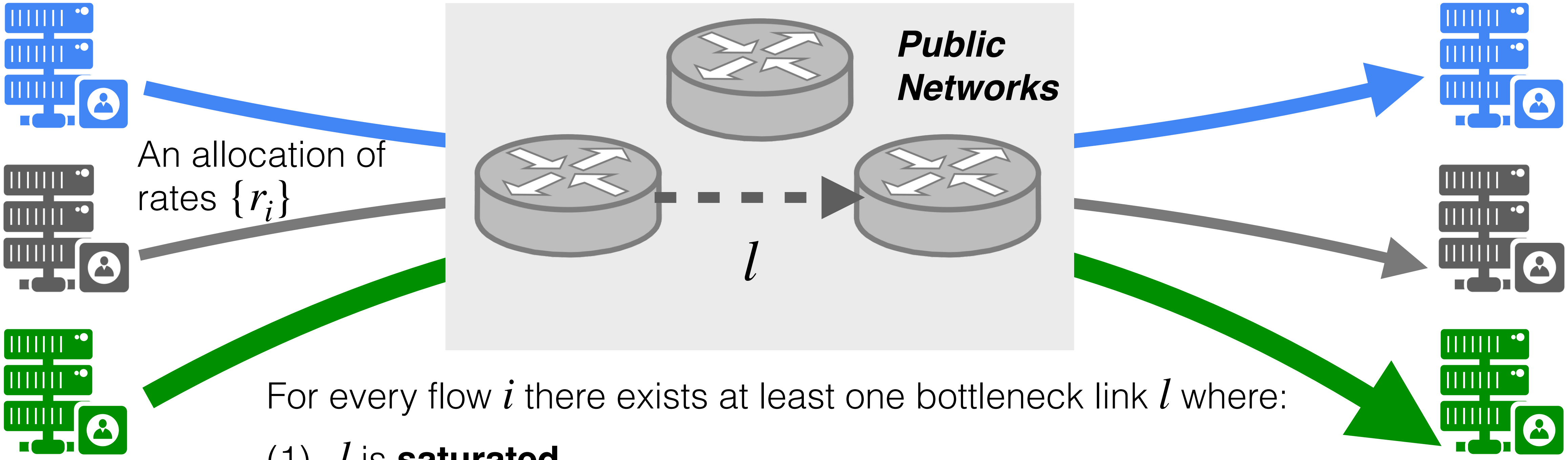
Cebinae router architecture for binary taxation

- **Zero modifications and coordinations** to/with legacy host CCAs
- Requirement of only **two queues/priorities**
- Compatibility with CCAs operating on **both loss and delay** signals

Outline

1. Conceptual foundation for binary classification
2. Cebinae's taxation mechanism
3. Evaluation

Max-min fairness

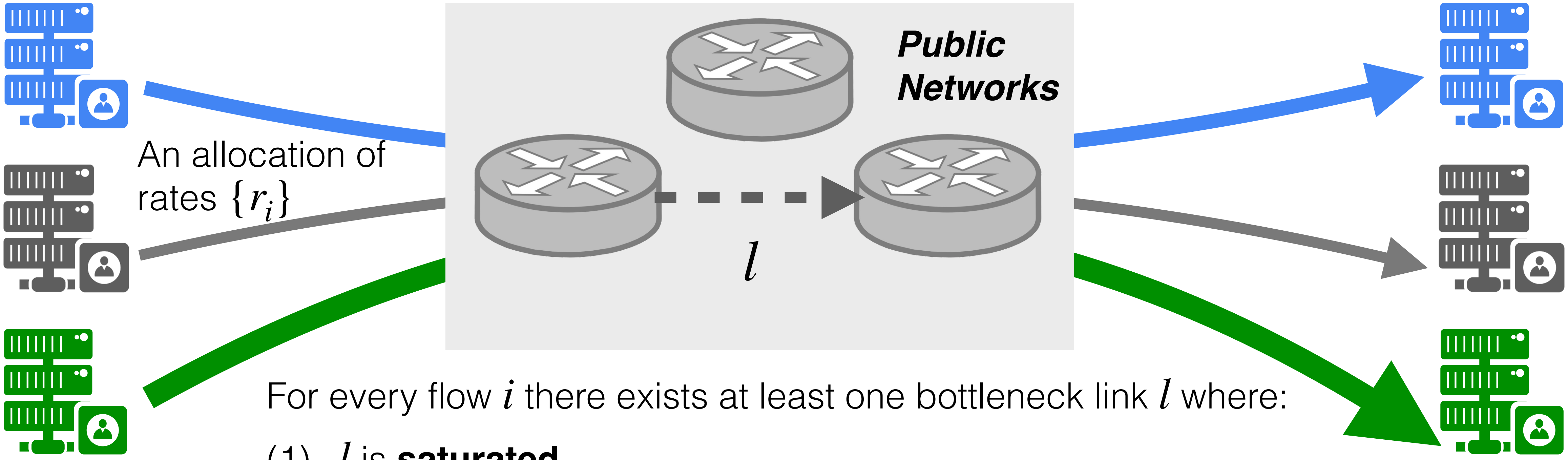


For every flow i there exists at least one bottleneck link l where:

(1) l is **saturated**

(2) r_i is among **the largest** flows sharing the link l

Max-min fairness



For every flow i there exists at least one bottleneck link l where:

(1) l is **saturated**

(2) r_i is among **the largest** flows sharing the link l

<u>Implication:</u> distributed *verification* of max-min fairness

Local verification

Each link l can determine the set of bottlenecked flows:

If l non-saturated:

  All flows not bottlenecked

Else, for each flow i:

  If i is among l's largest rate(s)

    i *is* ***bottlenecked*** *at* l

  Else

    i is **not bottlenecked** at l

Local verification

Each link l can determine the set of bottlenecked flows:

If l non-saturated:

All flows not bottlenecked

Else, for each flow i:

If i is among l's largest rate(s)

i *is **bottlenecked** at* l

Else

i is **not bottlenecked** at l

Observation:

1. Each conditional can be determined using ***only local information***
2. ***Binary classification***: bottlenecked ($\top$), not bottlenecked ($\perp$)

Naive enforcement

<u>Each link l can determine the set of bottlenecked flows:</u>

If l non-saturated:

NOP

Else, for each flow i:

If i is among l's largest rate(s)

*Drop packets of **all is** per their current rate*

Else

NOP

Naive enforcement

Each link l can determine the set of bottlenecked flows:

If l non-saturated:

NOP

Else, for each flow i:

If i is among l's largest rate(s)

*Drop packets of **all** $\boldsymbol{i}$**s** per their current rate*

Else

NOP

Drawbacks:

1. Can not push an already-unfair allocation fair
2. CCAs may not be responsive to loss signals

Cebinae taxation

Each link l can determine the set of bottlenecked flows:

If l non-saturated:

NOP

Else, for each flow i:

If i is among l's largest rate(s)

Penalize *is with their* ***taxed rate***

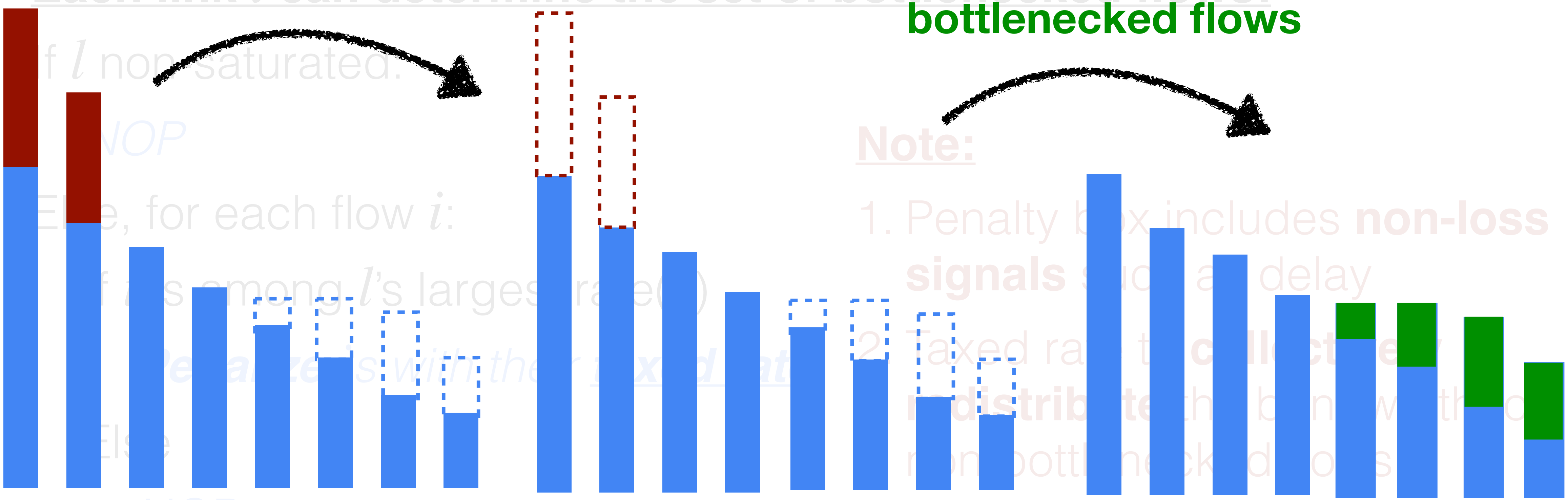
Else

NOP

Note:

1. Penalty box includes **non-loss signals** such as delay
2. Taxed rate to **collectively redistribute** the bandwidth to non-bottlenecked flows

Cebinae taxation



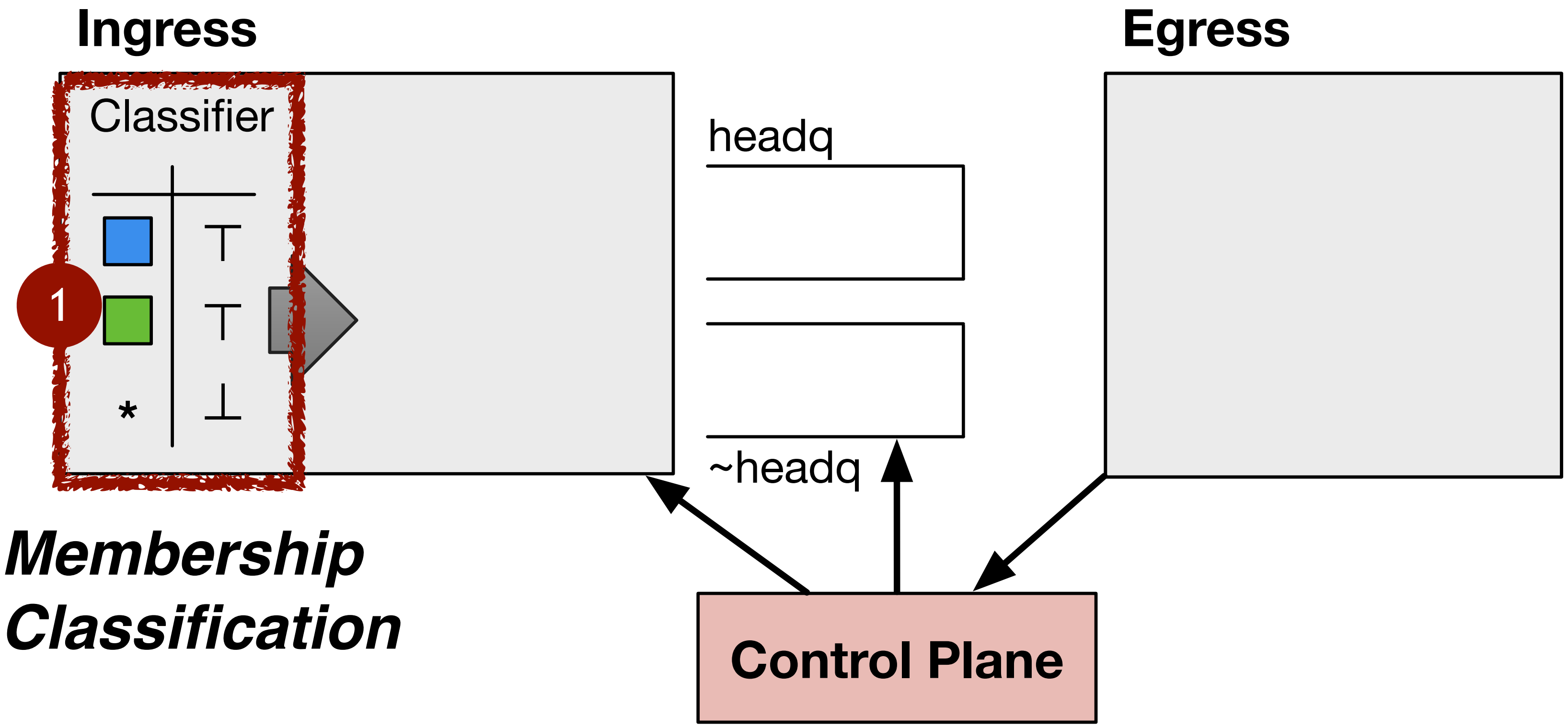
Instantiation: Cebinae router architecture

1. **Egress-pipeline cache**: port saturation and $\top$ flow status tracking

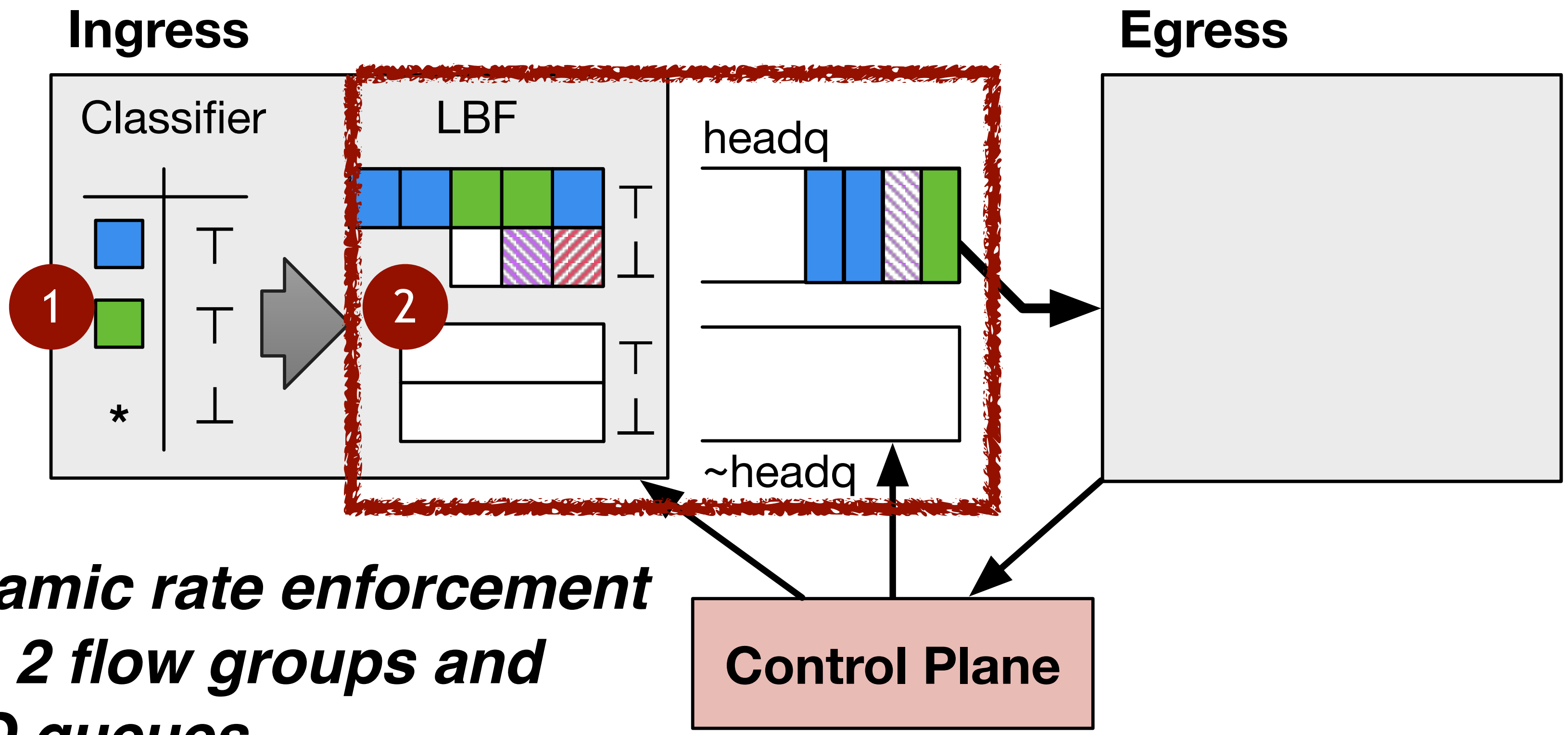
2. **Ingress-pipeline leaky-bucket filter**: $\top$ flow taxation

3. **Local CPU dynamic shuffling agent**: egress state polling and reconfiguration of $\top$ flow membership, redistributed rates, and queues

Normal operation

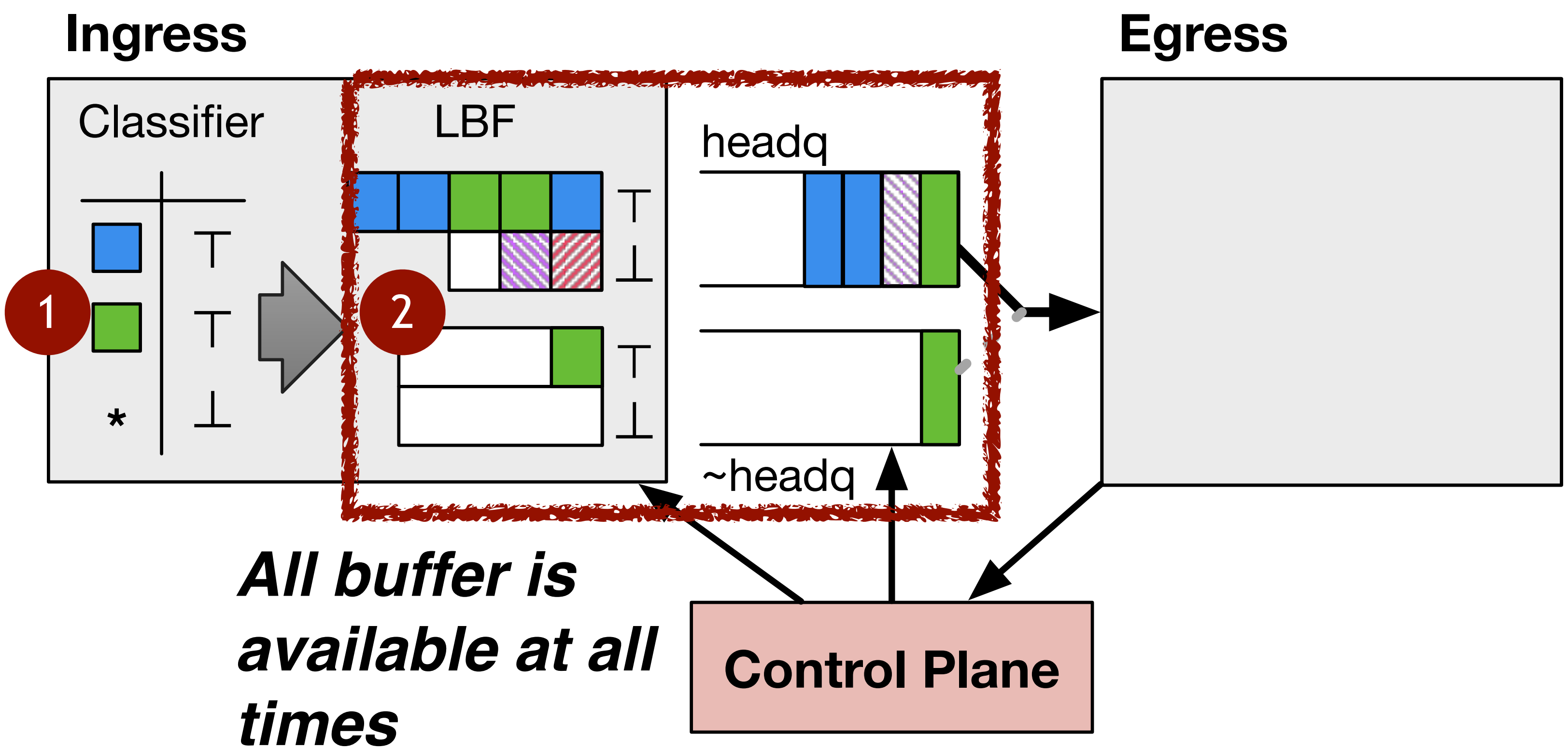


Normal operation



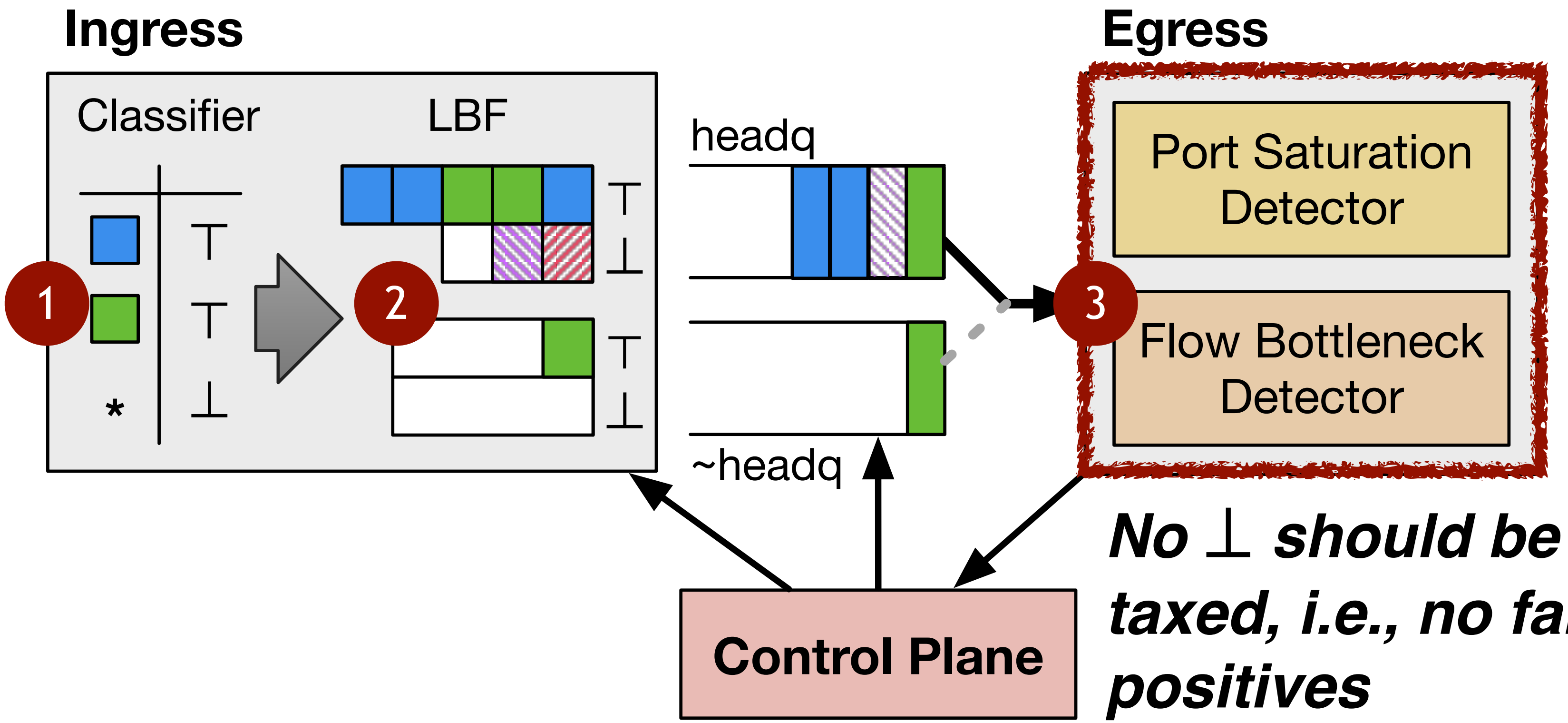
Dynamic rate enforcement with 2 flow groups and FIFO queues

Normal operation



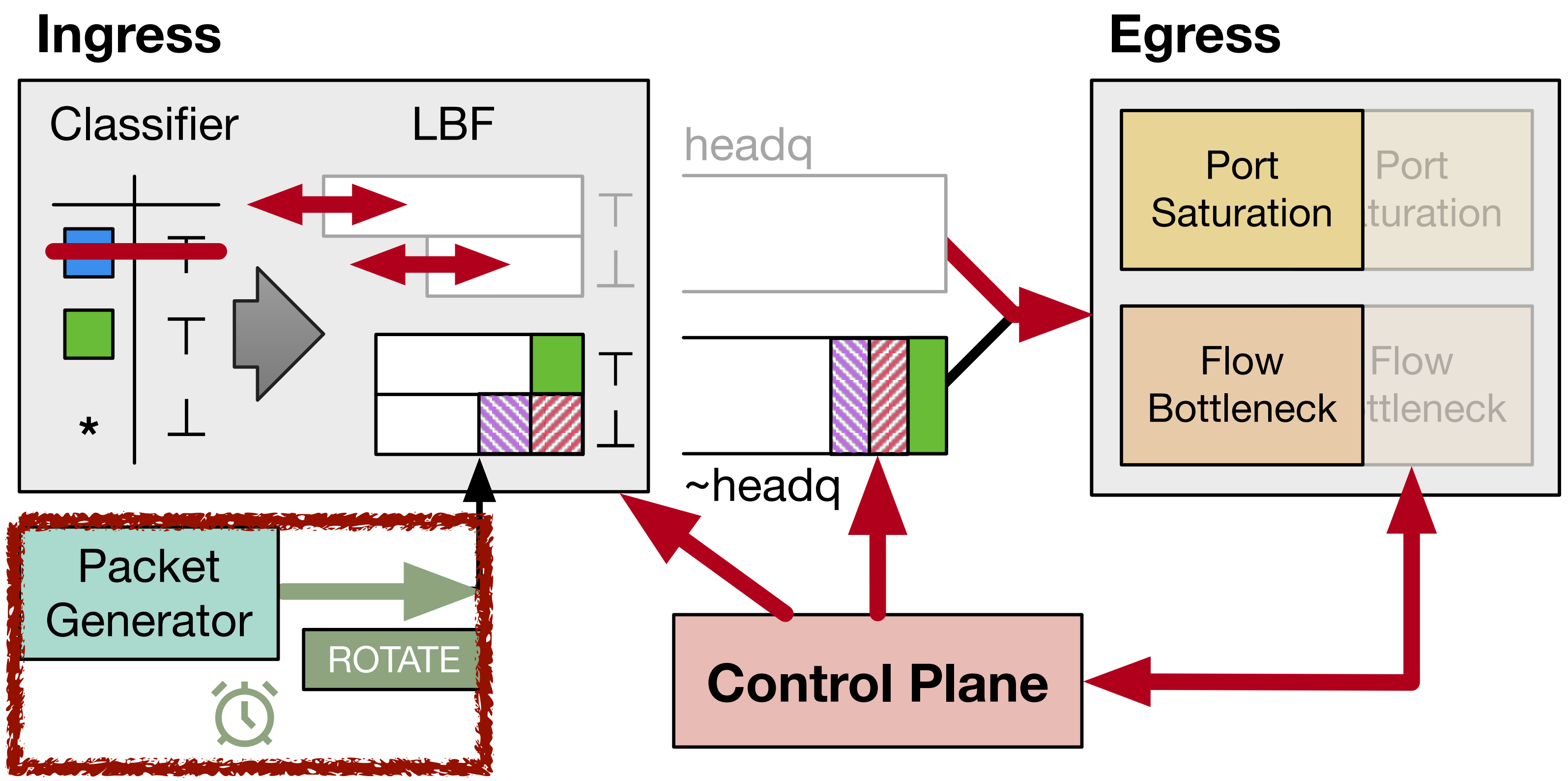
All buffer is available at all times

Normal operation

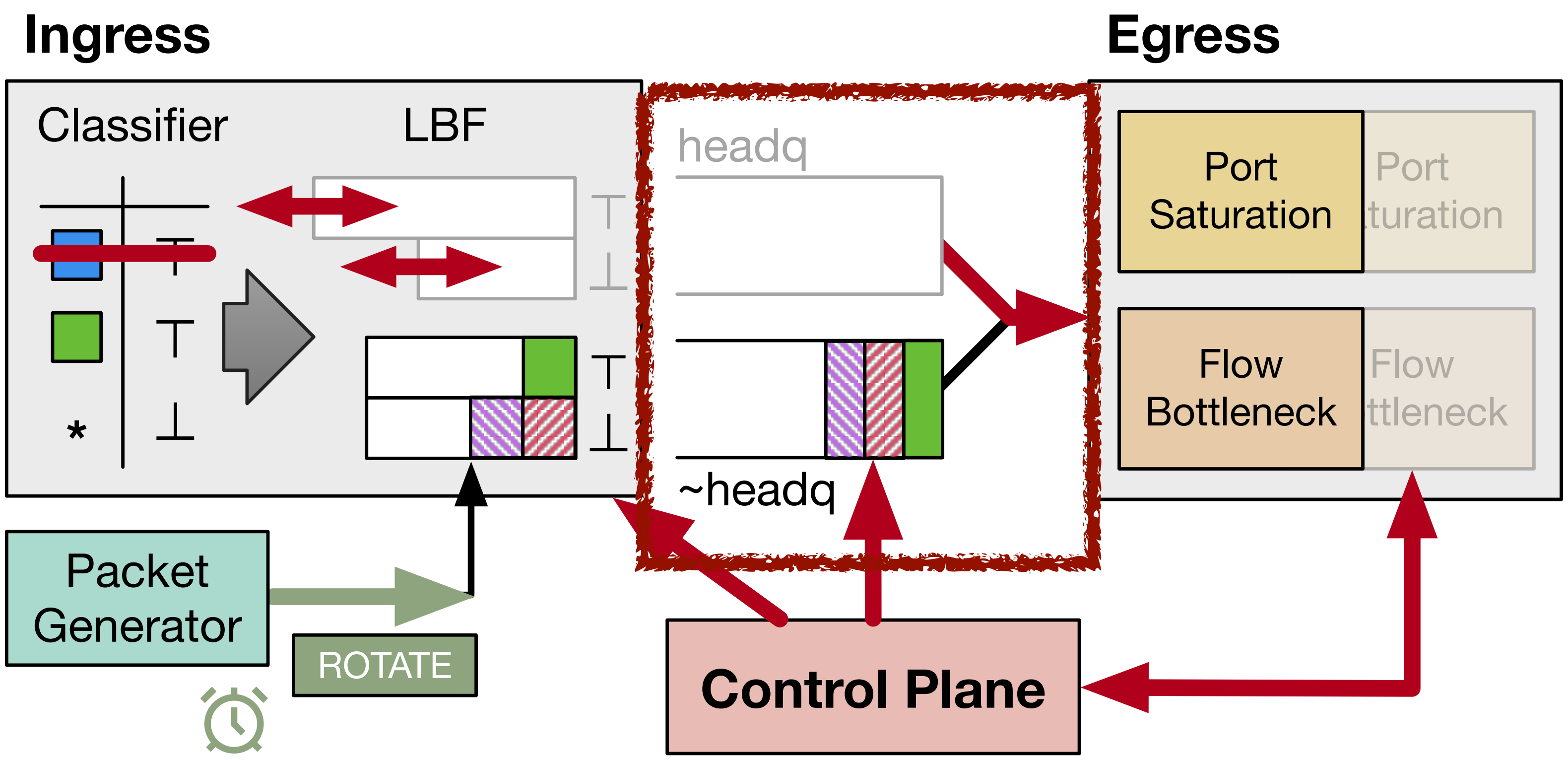


No ⊥ should be taxed, i.e., no false positives

Per-round reconfiguration

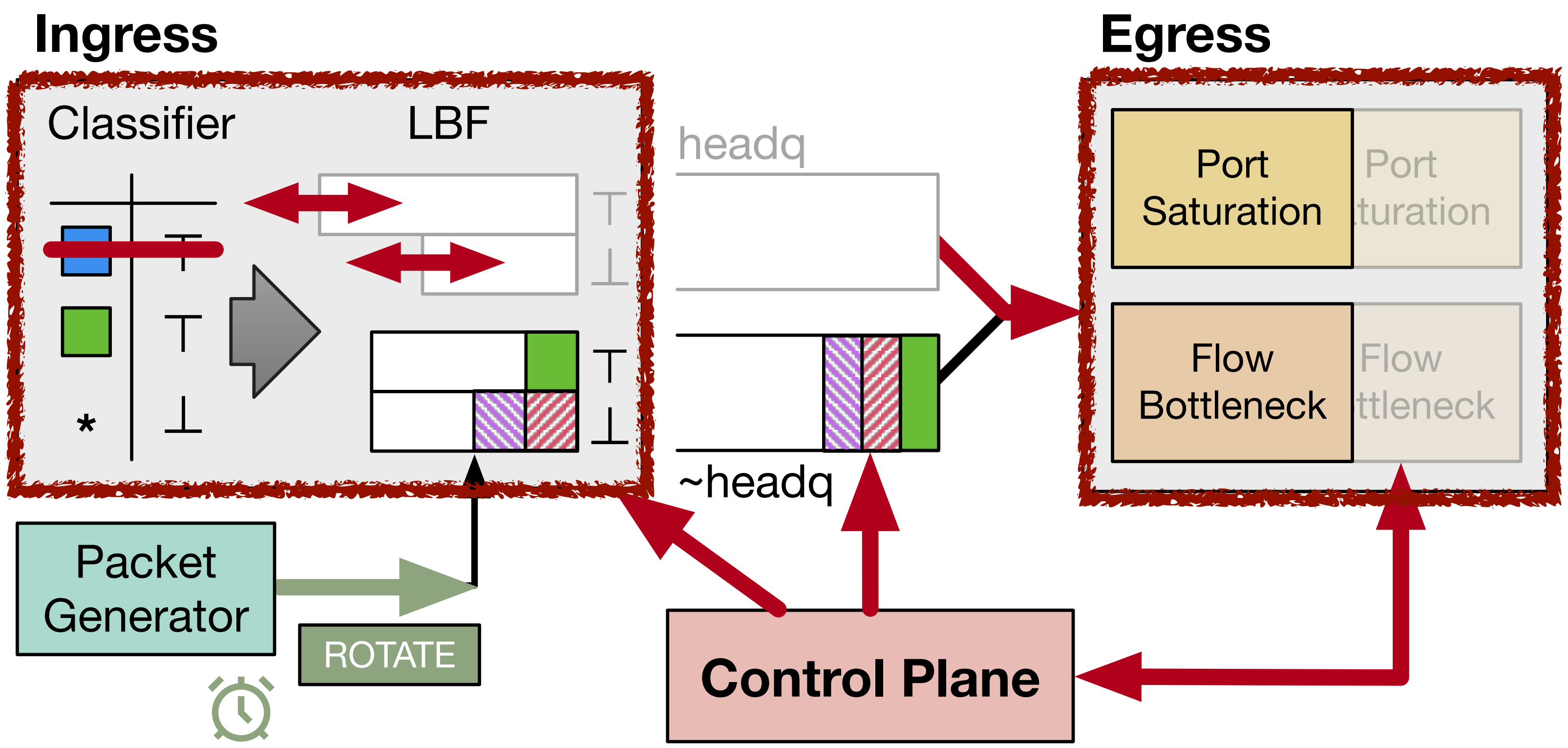


Per-round reconfiguration



Virtual pacing : *guarantee* ***no reordering*** *and avoid violation of draining deadline in the worst case*

Per-round reconfiguration



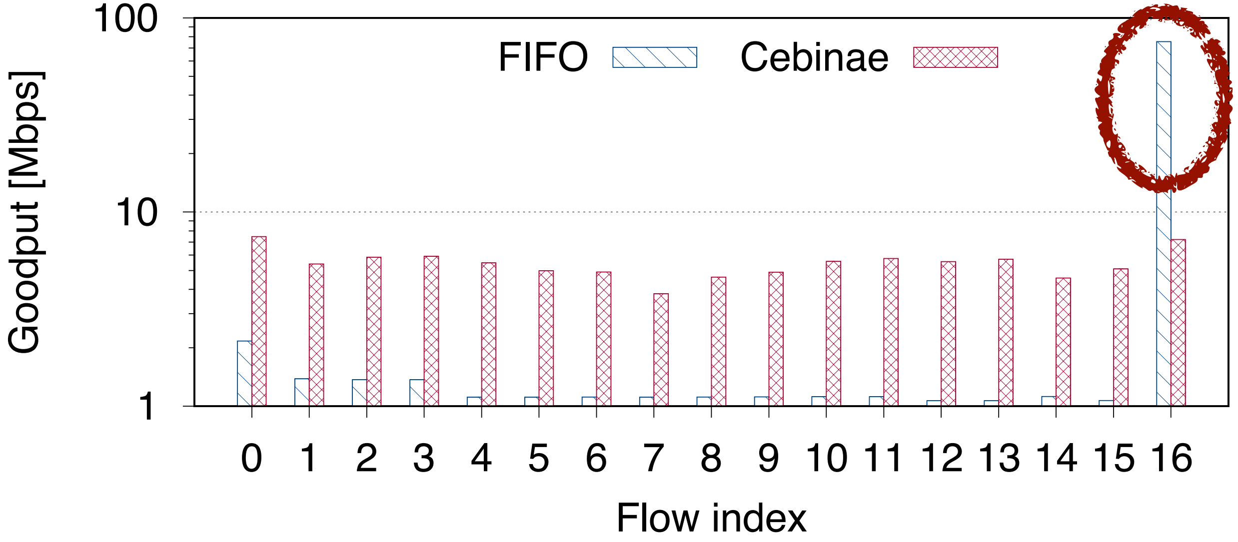
***Atomic transactions**: LBF states and egress caches*

Implementation and evaluation

Hardware prototype on a Wedge100BF Tofino switch testbed and NS-3 module

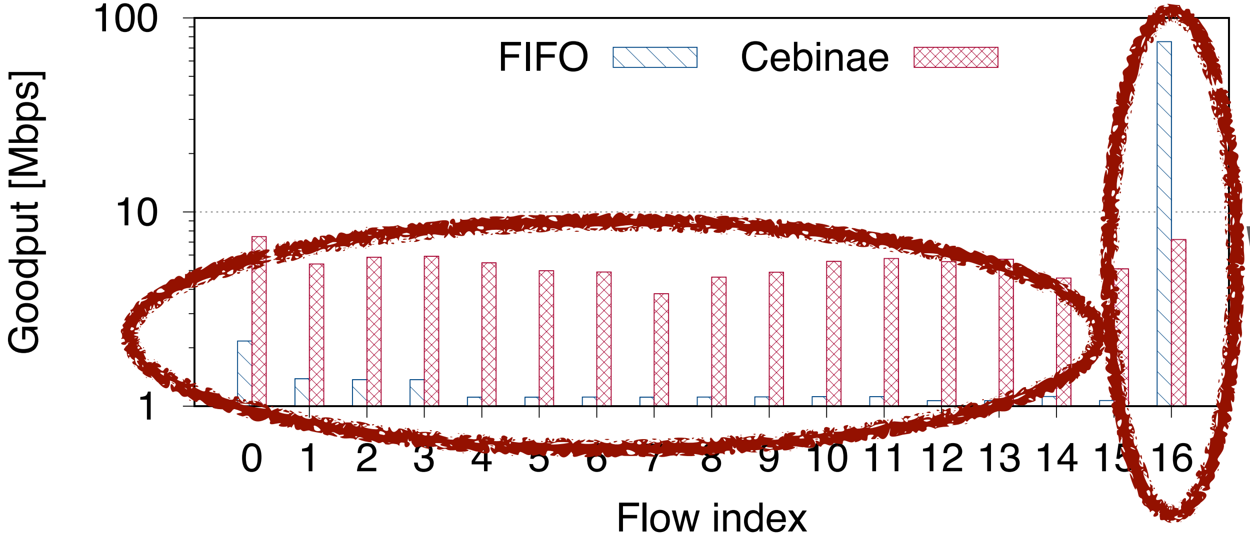
- Is Cebinae agnostic to CCAs?
- Can Cebinae mitigates unfairness (RTT, inter-CCA)?
- Can Cebinae move towards max-min fairness?
- Is Cebinae easy to configure?
- Does Cebinae resource usage scale?
- …

Cebinae mitigates unfairness



16 TCP Vegas (0–15)
v.s. 1 NewReno (16)

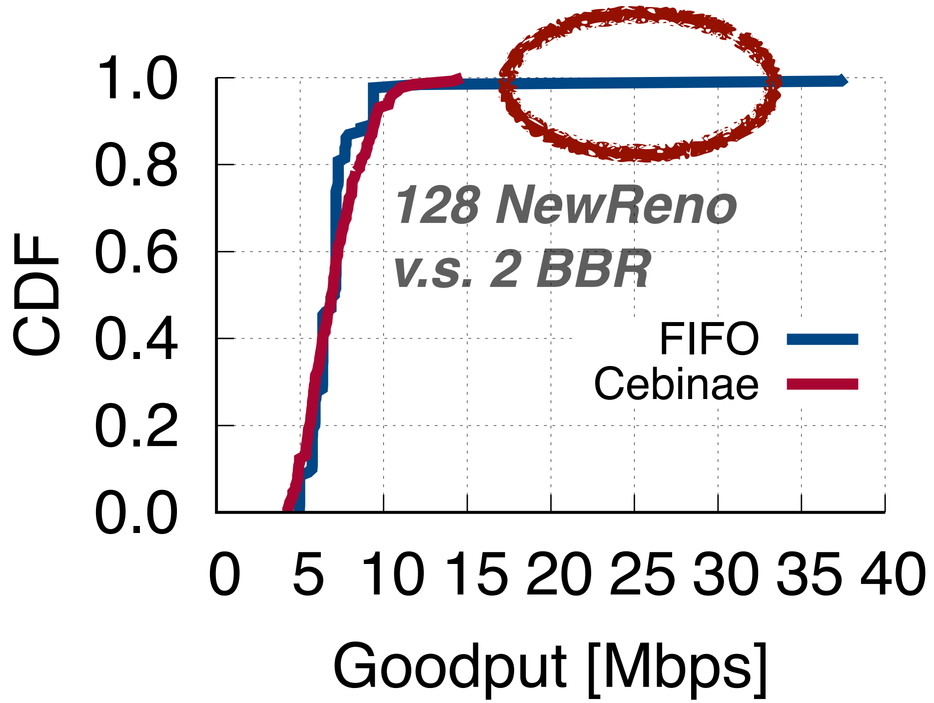
Cebinae mitigates unfairness



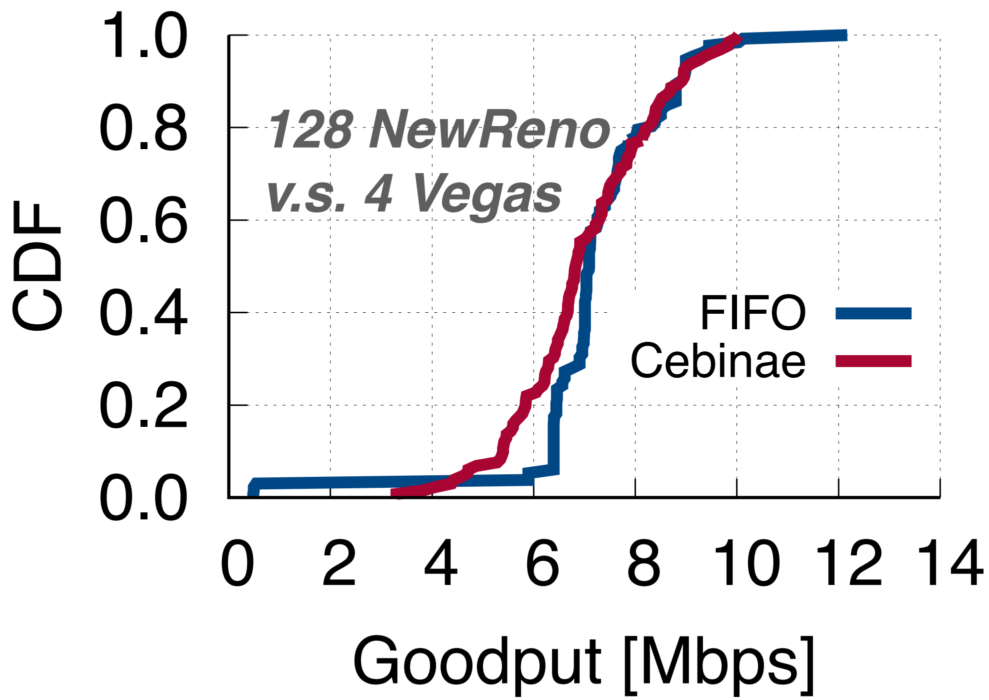
16 TCP Vegas (0–15)
v.s. 1 NewReno (16)

Mitigates the ***skewed and persistent unfairness*** *with little efficiency impact:* ***JFI from 0.093 to 0.984***

Cebinae mitigates unfairness

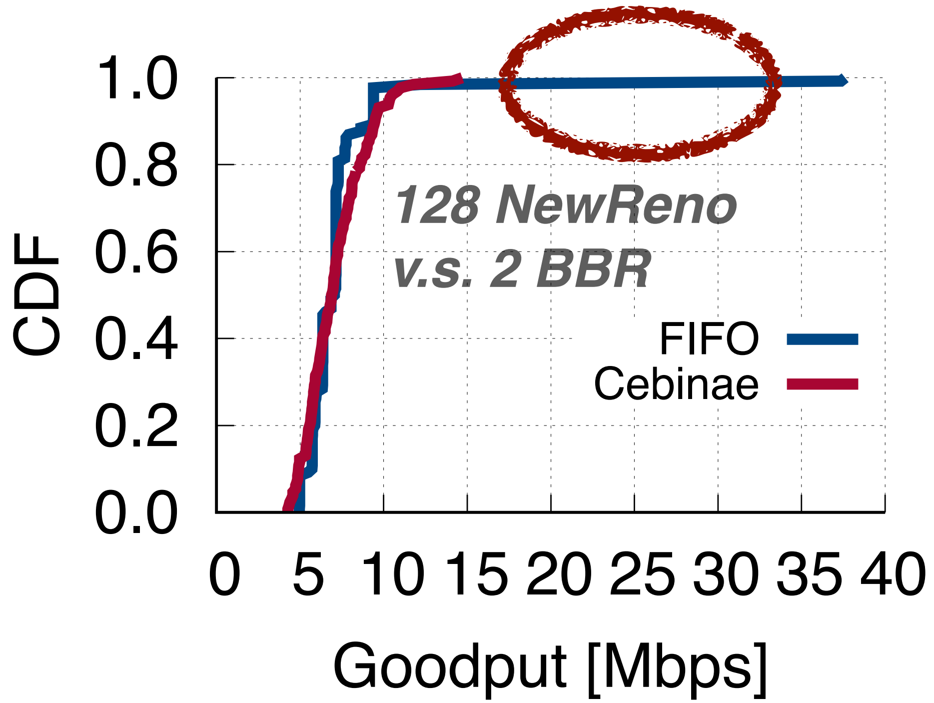


Preventing aggressiveness

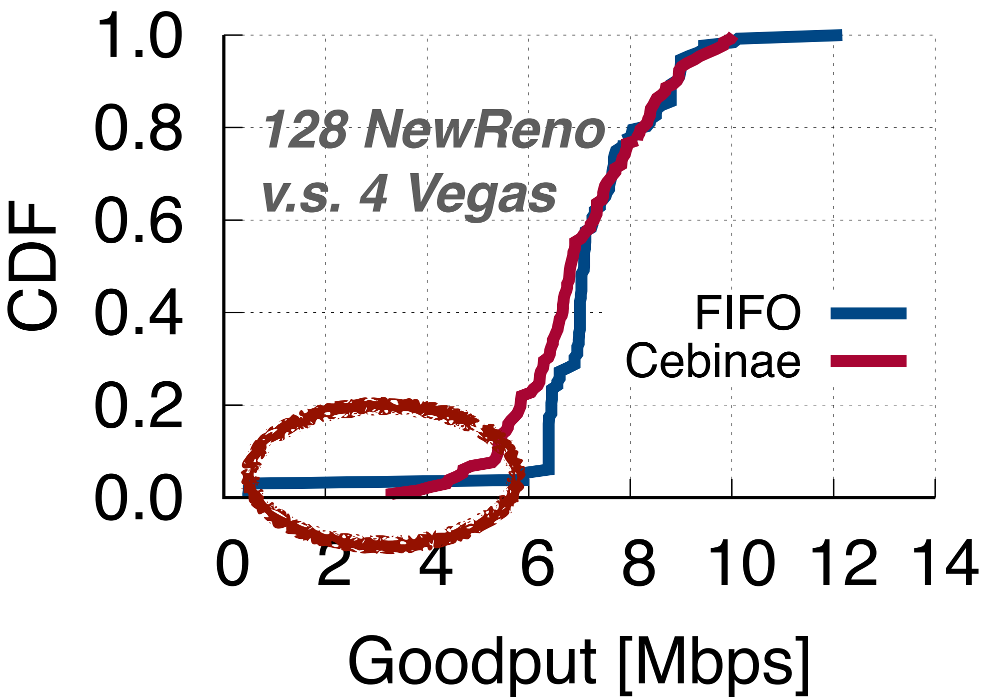


Mitigating starvation

Cebinae mitigates unfairness



Preventing aggressiveness



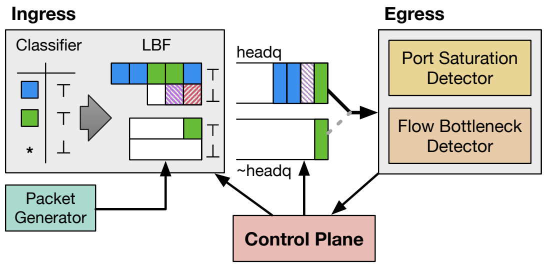
Mitigating starvation

Cebinae mitigates unfairness

Btl. BW	RTTs [ms]	Buf. [MTU]	CCAs	Throughput [Mbps] FIFO	Throughput [Mbps] FQ	Throughput [Mbps] Cebinae	Goodput [Mbps] FIFO	Goodput [Mbps] FQ	Goodput [Mbps] Cebinae	JFI FIFO	JFI FQ	JFI Cebinae
100 Mbps	{20.8, 28}	250	{NewReno:2, NewReno:8}	98.95	95.62	95.92	95.35	92.16	92.44	0.740	0.982	0.999
100 Mbps	{20.4, 40}	350	{Cubic:8, Cubic:2}	98.96	98.95	98.00	95.37	95.37	94.45	0.539	1.000	0.980
100 Mbps	{20.4, 60}	500	{Vegas:2, Vegas:8}	98.88	98.83	98.88	95.29	95.24	95.29	0.873	1.000	0.993
100 Mbps	{200}	1700	{NewReno:16, Cubic:1}	98.28	90.99	94.53	94.38	87.61	91.02	0.446	0.995	0.925
100 Mbps	{100}	850	{NewReno:16, Cubic:1}	98.72	91.45	95.58	95.11	88.10	92.08	0.857	0.998	0.960
100 Mbps	{50}	420	{NewReno:16, Cubic:1}	98.90	93.86	95.37	95.30	90.45	91.90	0.936	0.999	0.993
100 Mbps	{50}	420	{Vegas:16, Cubic:1}	98.90	98.90	95.47	95.30	95.30	91.99	0.096	1.000	0.988
100 Mbps	{100}	850	{Vegas:16, NewReno:1}	98.71	97.77	95.67	95.07	94.19	92.16	0.093	0.999	0.985
100 Mbps	{100}	850	{Vegas:128, NewReno:1}	98.88	98.74	97.45	95.26	95.10	93.88	0.189	0.966	0.976
100 Mbps	{60}	500	{Vegas:8, NewReno:8, Cubic: 2}	98.87	98.02	96.52	95.27	94.45	93.00	0.510	0.991	0.973
1 Gbps	{5}	420	{NewReno:32, Cubic:8}	989.8	989.8	985.4	954.0	954.0	949.7	0.844	0.988	0.955
1 Gbps	{10}	850	{Vegas:128, Cubic:1}	989.8	989.8	968.0	954.0	954.0	932.9	0.048	0.966	0.953
1 Gbps	{10}	850	{Vegas:1024, Cubic:2}	989.8	989.8	949.2	953.6	953.6	914.1	0.275	0.833	0.846
1 Gbps	{50}	4200	{NewReno: 128, BBR: 1}	988.7	923.6	981.6	952.7	890.0	945.8	0.992	0.975	0.990
1 Gbps	{50}	4200	{NewReno: 128, BBR: 2}	988.9	953.9	979.9	952.8	919.2	944.2	0.951	0.963	0.981
1 Gbps	{50}	21000	{NewReno: 128, BBR: 2}	988.8	953.9	963.8	952.7	919.2	928.7	0.773	0.963	0.936
1 Gbps	{100}	8350	{NewReno: 128, BBR: 2}	986.9	938.2	956.3	950.7	903.9	921.1	0.884	0.968	0.967
1 Gbps	{10}	850	{Vegas:64, NewReno:1}	989.8	989.8	976.2	953.8	954.0	940.7	0.042	0.967	0.976
1 Gbps	{100}	8500	{Vegas:4, NewReno:128}	986.9	917.6	957.3	950.8	884.1	922.2	0.946	0.970	0.971
1 Gbps	{100, 64}	8500	{Vegas:4, NewReno:128}	988.4	941.1	959.8	952.4	906.8	924.7	0.956	0.970	0.964
1 Gbps	{100}	8500	{Vegas:8, NewReno:128}	987.0	936.1	964.4	950.8	901.8	929.0	0.921	0.968	0.969
1 Gbps	{10}	850	{Vegas:128, BBR:1}	989.8	989.8	987.3	954.0	954.0	951.5	0.886	0.965	0.985
1 Gbps	{100}	8500	{Bic:2, Cubic:32}	985.1	960.3	952.6	944.9	924.9	911.3	0.799	0.999	0.946
10 Gbps	{50, 44}	41667	{NewReno:128, Cubic:16}	9876	9705	9780	9514	9352	9420	0.917	0.969	0.968
10 Gbps	{28, 28}	25000	{NewReno:128, Cubic:128}	9891	9856	9787	9532	9498	9432	0.863	0.942	0.952

Cebinae is agnostic to CCAs

Btl. BW	RTTs [ms]	Buf. [MTU]	CCAs	Throughput [Mbps] FIFO	FQ	Cebinae	Goodput [Mbps] FIFO	FQ	Cebinae	JFI FIFO	FQ	Cebinae
100 Mbps	{20.8, 28}	250	{NewReno:2, NewReno:8}	98.95	95.62	95.92	95.35	92.16	92.44	0.740	0.982	0.999
100 Mbps	{20.4, 40}	350	{Cubic:8, Cubic:2}	98.96	98.95	98.00	95.37	95.37	94.45	0.539	1.000	0.980
100 Mbps	{20.4, 60}	500	{Vegas:2, Vegas:8}	98.88	98.83	98.88	95.29	95.24	95.29	0.873	1.000	0.993
100 Mbps	{200}	1700	{NewReno:16, Cubic:1}	98.28	90.99	94.53	94.38	87.61	91.02	0.446	0.995	0.925
100 Mbps	{100}	850	{NewReno:16, Cubic:1}	98.72	91.45	95.58	95.11	88.10	92.08	0.857	0.998	0.960
100 Mbps	{50}	420	{NewReno:16, Cubic:1}	98.90	93.86	95.37	95.30	90.45	91.90	0.936	0.999	0.993
100 Mbps	{50}	420	{Vegas:16, Cubic:1}	98.90	98.90	95.47	95.30	95.30	91.99	0.096	1.000	0.988
100 Mbps	{100}	850	{Vegas:16, NewReno:1}	98.71	97.77	95.67	95.07	94.19	92.16	0.093	0.999	0.985
100 Mbps	{100}	850	{Vegas:128, NewReno:1}	98.88	98.74	97.45	95.26	95.10	93.88	0.189	0.966	0.976
100 Mbps	{60}	500	{Vegas:8, NewReno:8, Cubic: 2}	98.87	98.02	96.52	95.27	94.45	93.00	0.510	0.991	0.973
1 Gbps	{5}	420	{NewReno:32, Cubic:8}	989.8	989.8	985.4	954.0	954.0	949.7	0.844	0.988	0.955
1 Gbps	{10}	850	{Vegas:128, Cubic:1}	989.8	989.8	968.0	954.0	954.0	932.9	0.048	0.966	0.953
1 Gbps	{10}	850	{Vegas:1024, Cubic:2}	989.8	989.8	949.2	953.6	953.6	914.1	0.275	0.833	0.846
1 Gbps	{50}	4200	{NewReno: 128, BBR: 1}	988.7	923.6	981.6	952.7	890.0	945.8	0.992	0.975	0.990
1 Gbps	{50}	4200	{NewReno: 128, BBR: 2}	988.9	953.9	979.9	952.8	919.2	944.2	0.951	0.963	0.981
1 Gbps	{50}	21000	{NewReno: 128, BBR: 2}	988.8	953.9	963.8	952.7	919.2	928.7	0.773	0.963	0.936
1 Gbps	{100}	8350	{NewReno: 128, BBR: 2}	986.9	938.2	956.3	950.7	903.9	921.1	0.884	0.968	0.967
1 Gbps	{10}	850	{Vegas:64, NewReno:1}	989.8	989.8	976.2	953.8	954.0	940.7	0.042	0.967	0.976
1 Gbps	{100}	8500	{Vegas:4, NewReno:128}	986.9	917.6	957.3	950.8	884.1	922.2	0.946	0.970	0.971
1 Gbps	{100, 64}	8500	{Vegas:4, NewReno:128}	988.4	941.1	959.8	952.4	906.8	924.7	0.956	0.970	0.964
1 Gbps	{100}	8500	{Vegas:8, NewReno:128}	987.0	936.1	964.4	950.8	901.8	929.0	0.921	0.968	0.969
1 Gbps	{10}	850	{Vegas:128, BBR:1}	989.8	989.8	987.3	954.0	954.0	951.5	0.886	0.965	0.985
1 Gbps	{100}	8500	{Bic:2, Cubic:32}	985.1	960.3	952.6	944.9	924.9	911.3	0.799	0.999	0.946
10 Gbps	{50, 44}	41667	{NewReno:128, Cubic:16}	9876	9705	9780	9514	9352	9420	0.917	0.969	0.968
10 Gbps	{28, 28}	25000	{NewReno:128, Cubic:128}	9891	9856	9787	9532	9498	9432	0.863	0.942	0.952



Summary

- **No modifications nor coordinations** to/with legacy host CCAs
    - *Real-time switch architecture serializing in-network compute modules*
- COTS hardware and **minimal resource requirements**
    - *Two queues/priorities are sufficient*
- Compatible with CCAs using **both loss and non-loss signals**
    - *Generic support of a wide range of Internet CCAs and environments*

https://github.com/eniac/Cebinae

Thank you for your attention!