Beyond Jain's Fairness Index: Setting The Bar For the Deployment of Congestion Control Algorithms

Ranysha Ware
Carnegie Mellon University

Matthew K. Mukerjee
Nefeli Networks

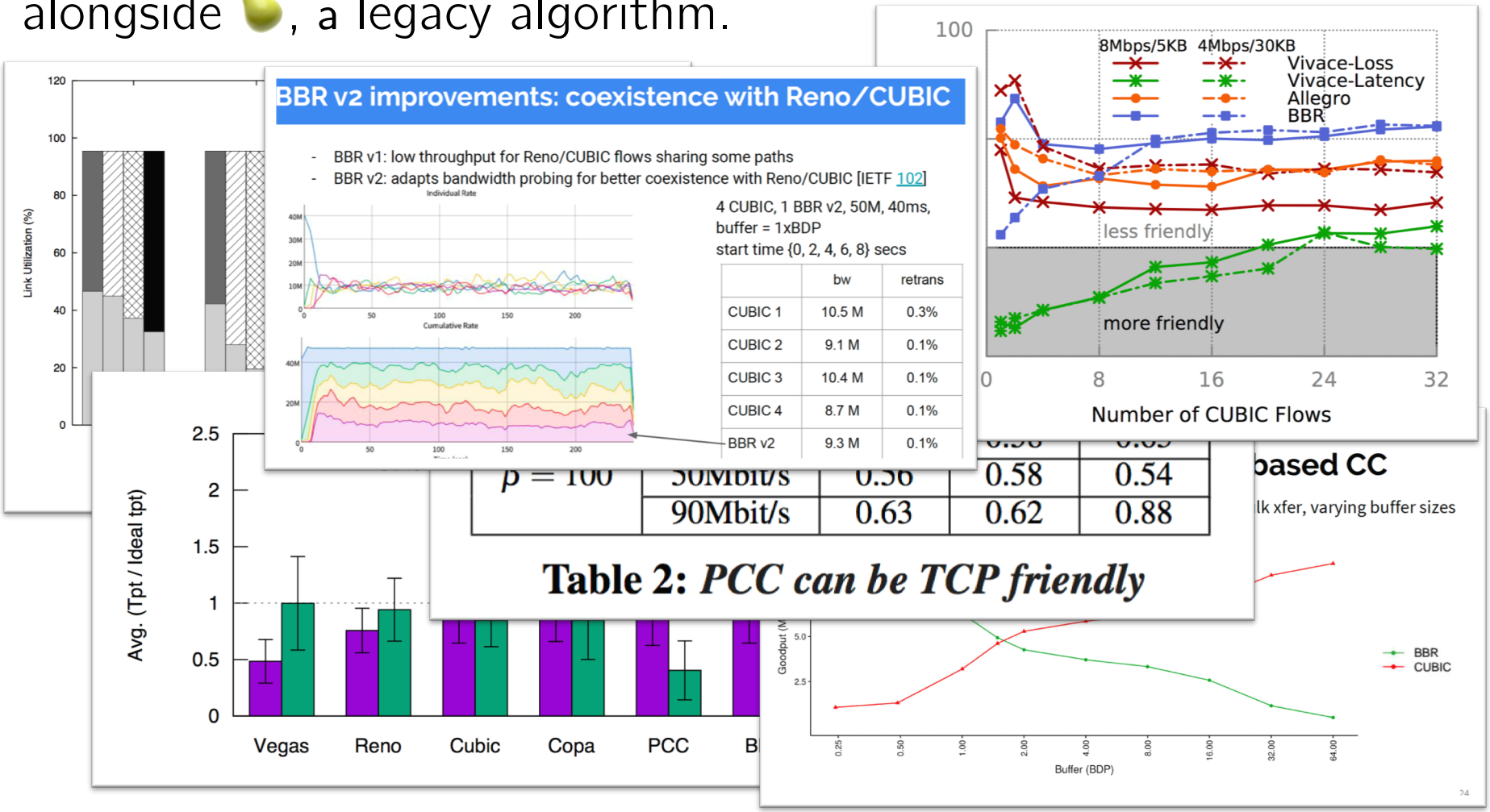
Justine Sherry
Carnegie Mellon University

Srinivasan Seshan
Carnegie Mellon University

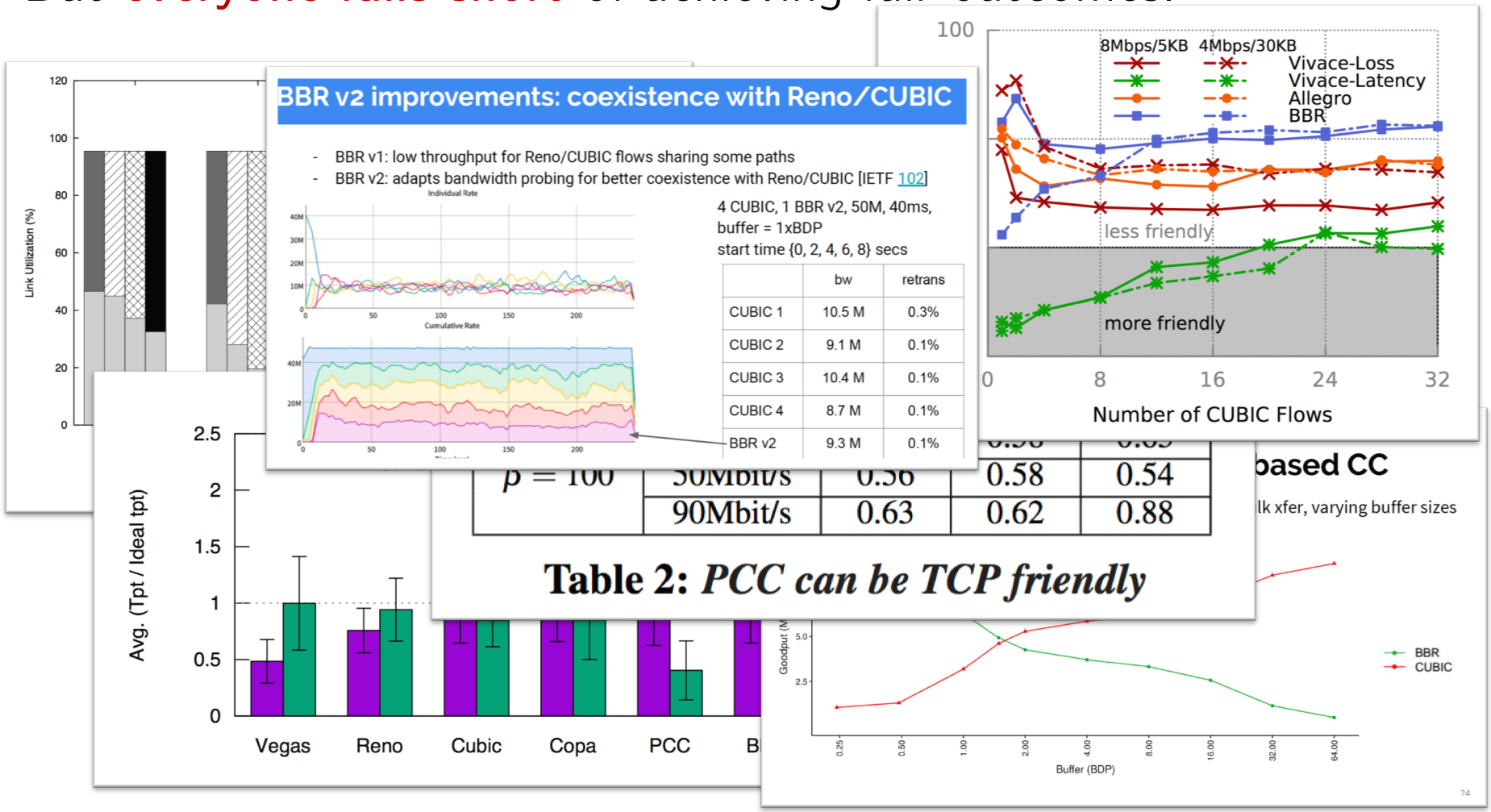
I have designed a new CCA: 🌮

How do we show 🌮 is reasonable to deploy in the Internet?

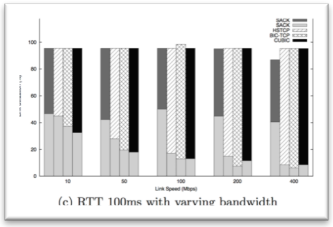
We typically use fairness to show that 🌮 is reasonably deployable alongside 🍐, a legacy algorithm.



But everyone falls short of achieving fair outcomes.

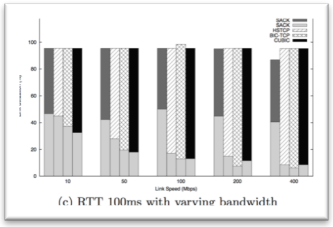


But everyone falls short of achieving fair outcomes.

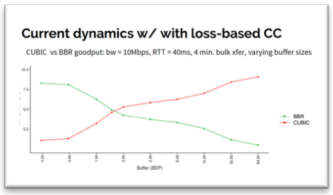


Cubic can be unfair to Reno, but "outside of TCP-friendly region" and "this doesn't highly impact Reno's performance."

But **everyone falls short** of achieving fair outcomes.

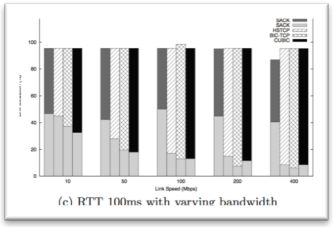


CUBIC can be unfair to Reno, but "outside of TCP-friendly region" and "this doesn't highly impact Reno's performance."

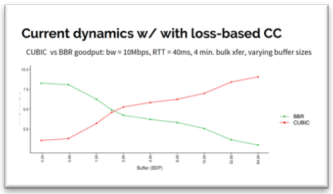


BBRv1 can be unfair to Cubic, but "we are looking at modeling shallow buffer situations".

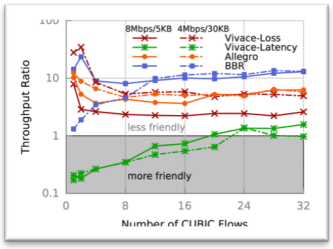
But **everyone falls short** of achieving fair outcomes.



CUBIC can be unfair to Reno, but "outside of TCP-friendly region" and "this doesn't highly impact Reno's performance."

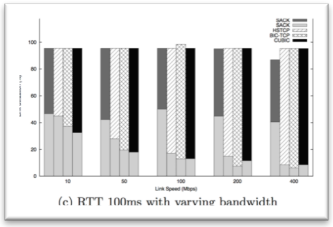


BBRv1 can be unfair to Cubic, but "we are looking at modeling shallow buffer situations".

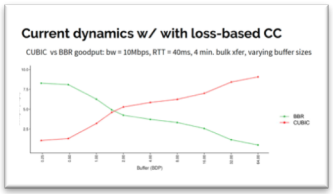


PCC Vivace can be unfair to Cubic, but "as the number of CUBIC senders increases, it achieves the best fairness among new generation protocols."

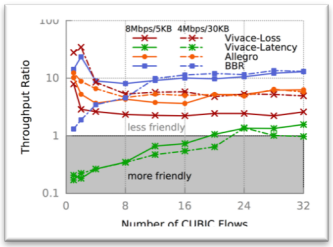
But **everyone falls short** of achieving fair outcomes.



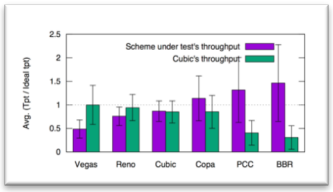
CUBIC can be unfair to Reno, but "outside of TCP-friendly region" and "this doesn't highly impact Reno's performance."



BBRv1 can be unfair to Cubic, but "we are looking at modeling shallow buffer situations".



PCC Vivace can be unfair to Cubic, but "as the number of CUBIC senders increases, it achieves the best fairness among new generation protocols."



Copa can be unfair to Cubic, but "is much fairer than BBR and PCC" and "uses bandwidth Cubic does not utilize."

Everyone makes excuses why their algorithm is still reasonable to deploy despite unfair outcomes.

This talk:
We need a practical deployment threshold: a bound on how aggressive 🌮, a new CCA, can be to 🍐, the status quo.

Outline:
1. What are desirable properties of a deployment threshold?

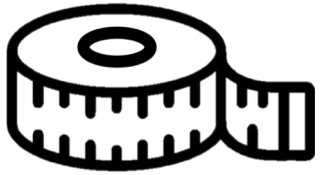
2. We define a new deployment threshold: harm.

Outline:

1. What are desirable properties of a deployment threshold?

2. We define a new deployment threshold: harm.

We identify 5 desirable properties for a deployment threshold.



MULTI-
METRIC

DEMAND-
AWARE

PRACTICAL

STATUS-QUO
BIASED

FUTURE-
PROOF

We identify 5 desirable properties for a deployment threshold.

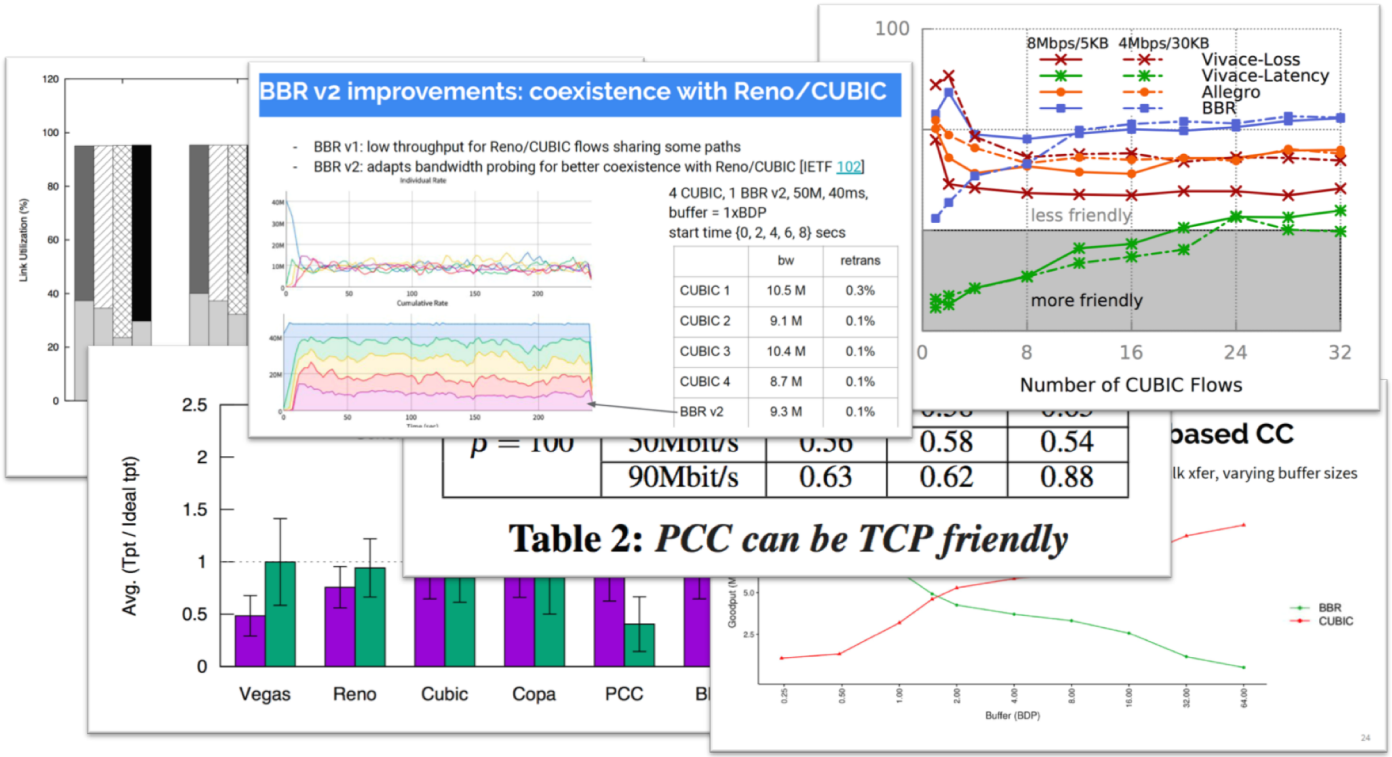
MULTI-METRIC

DEMAND-AWARE

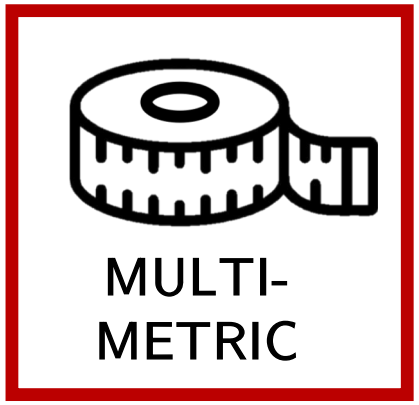
STATUS-QUO BIASED

FUTURE-PROOF

A deployment threshold needs to be practical: should be feasible for new CCA to meet threshold.



We identify 5 desirable properties for a deployment threshold.



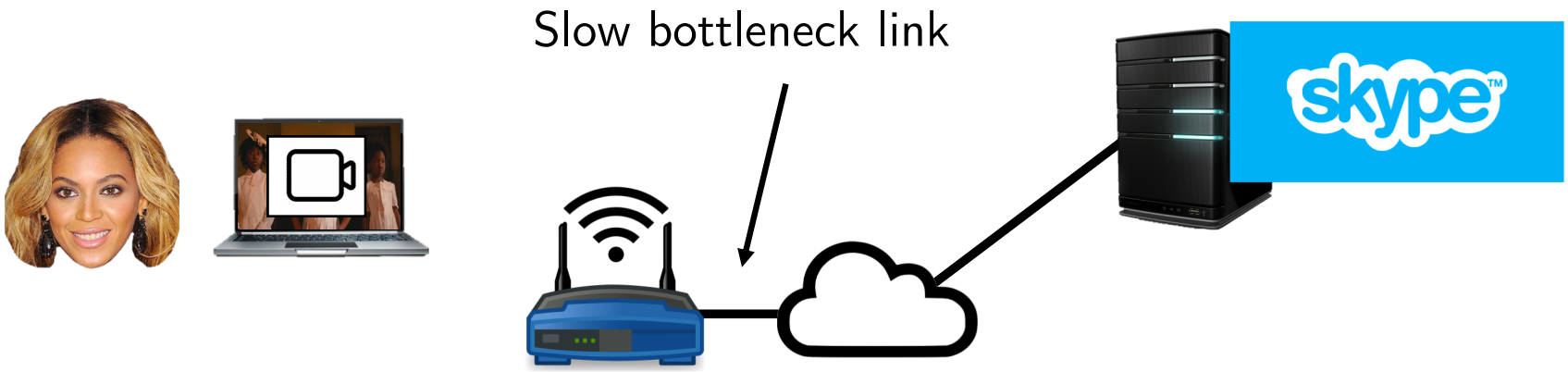
MULTI-
METRIC

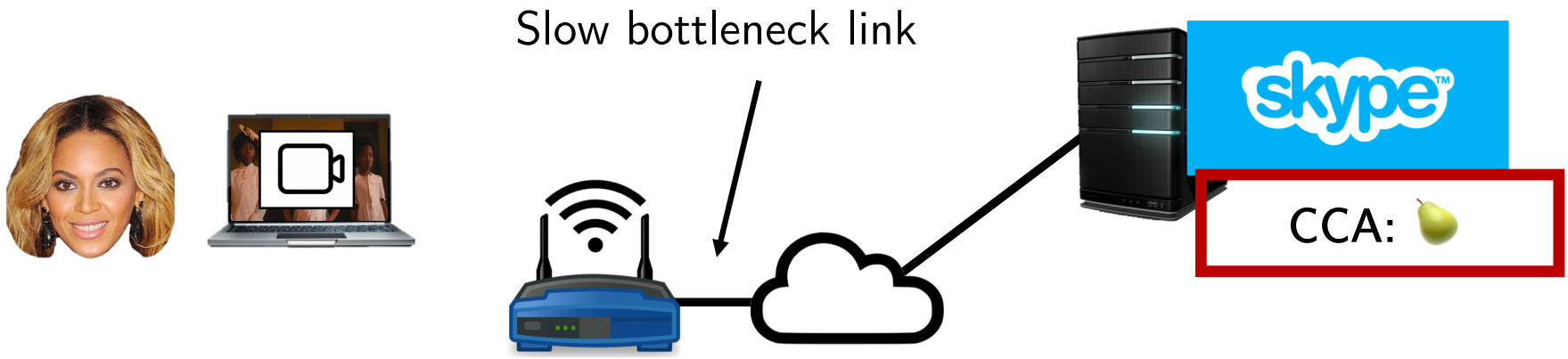
DEMAND-
AWARE

PRACTICAL

STATUS-QUO
BIASED

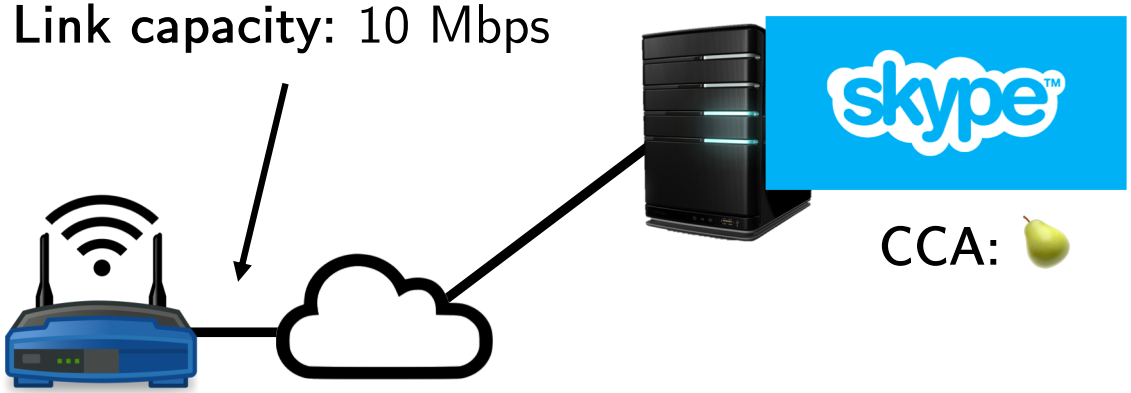
FUTURE-
PROOF


Slow bottleneck link
skype

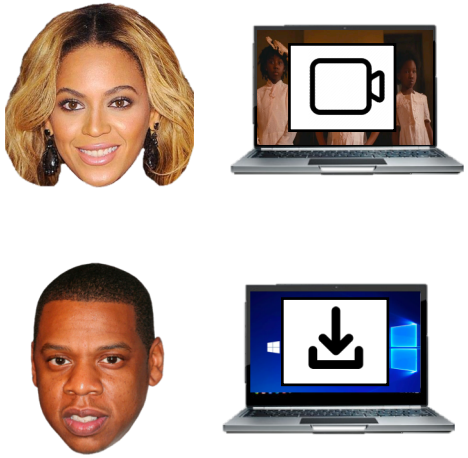

Slow bottleneck link
skype
CCA:

Latency: 5 ms
Download speed: 5 Mbps

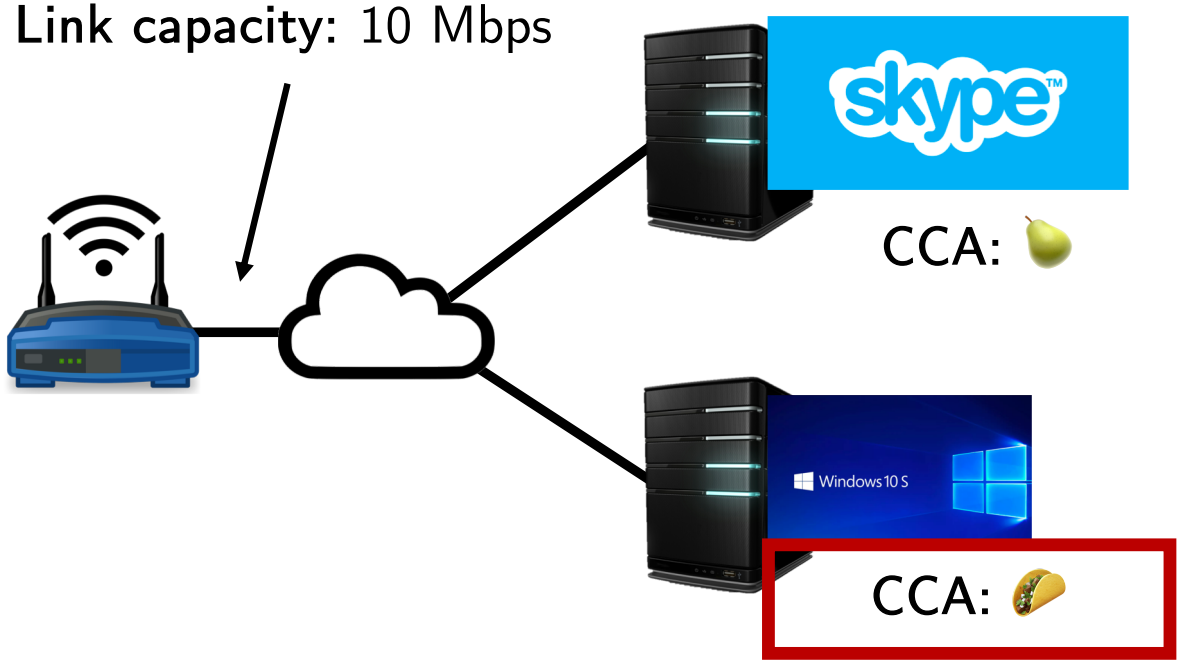
Link capacity: 10 Mbps



Latency: 5 ms
Download speed: 5 Mbps



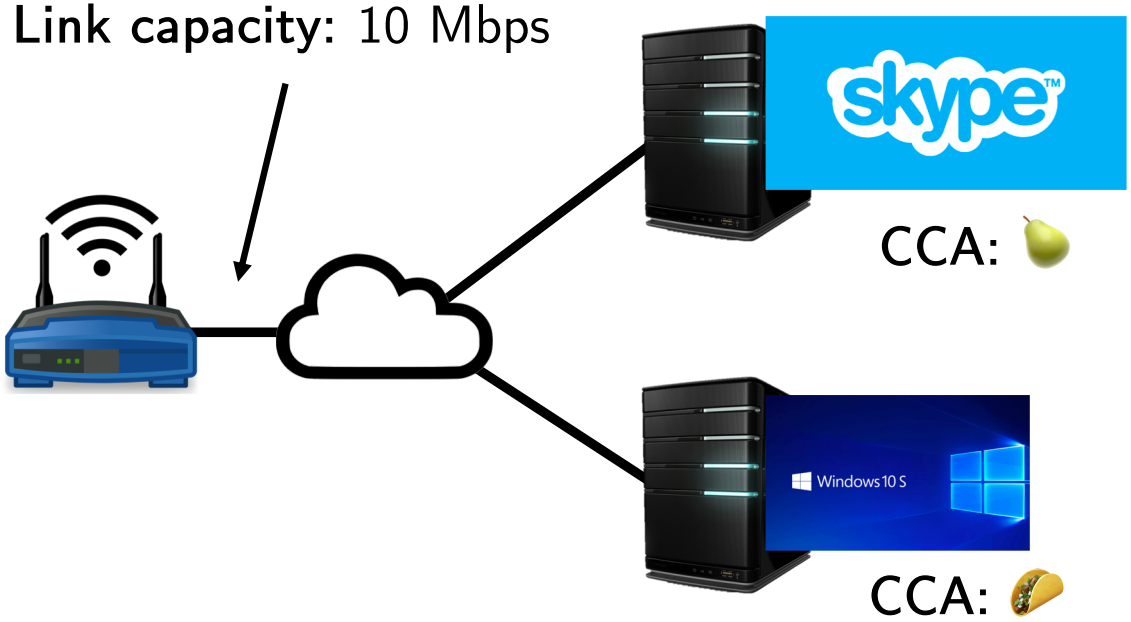
Link capacity: 10 Mbps



Latency: 5 ms
Download speed: 5 Mbps

Download speed: 5 Mbps

Link capacity: 10 Mbps



CCA:

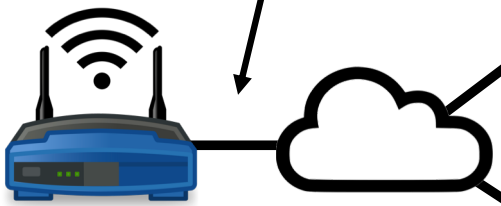
CCA:

Latency: ~~5 ms~~ 100 ms

Download speed: 5 Mbps

Download speed: 5 Mbps

Link capacity: 10 Mbps

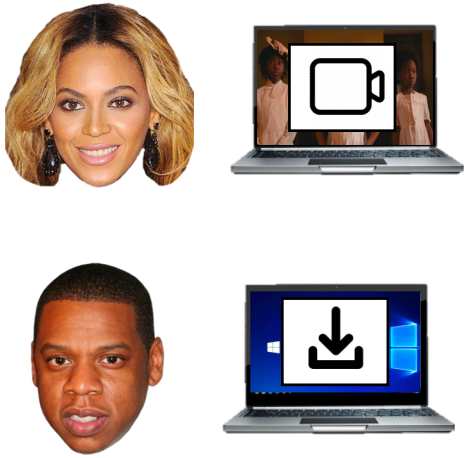


CCA:

CCA:

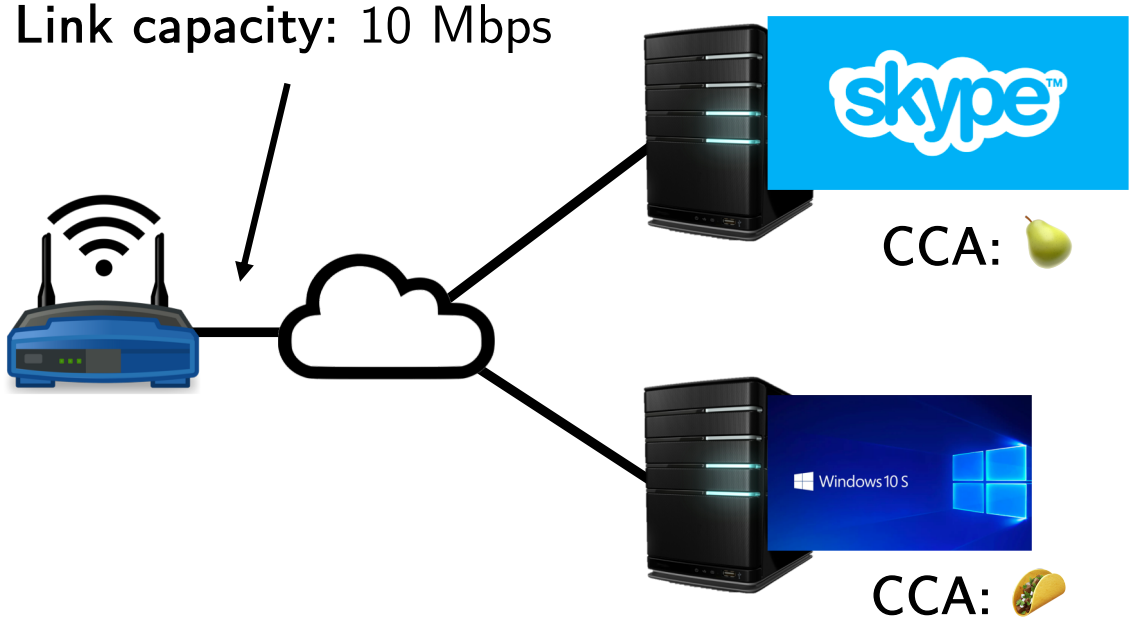
A deployment threshold needs to be **multi-metric**: can account for performance metrics beyond just throughput.

Latency: ~~5 ms~~ **100 ms**
Download speed: 5 Mbps



Download speed: 5 Mbps

Link capacity: 10 Mbps



CCA:

CCA:

Metrics like latency cannot be “divided fairly”.

We identify 5 desirable properties for a deployment threshold.

MULTI-METRIC

DEMAND-AWARE

PRACTICAL

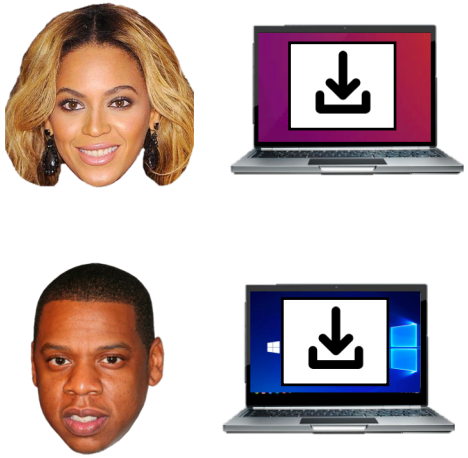
STATUS-QUO BIASED

FUTURE-PROOF

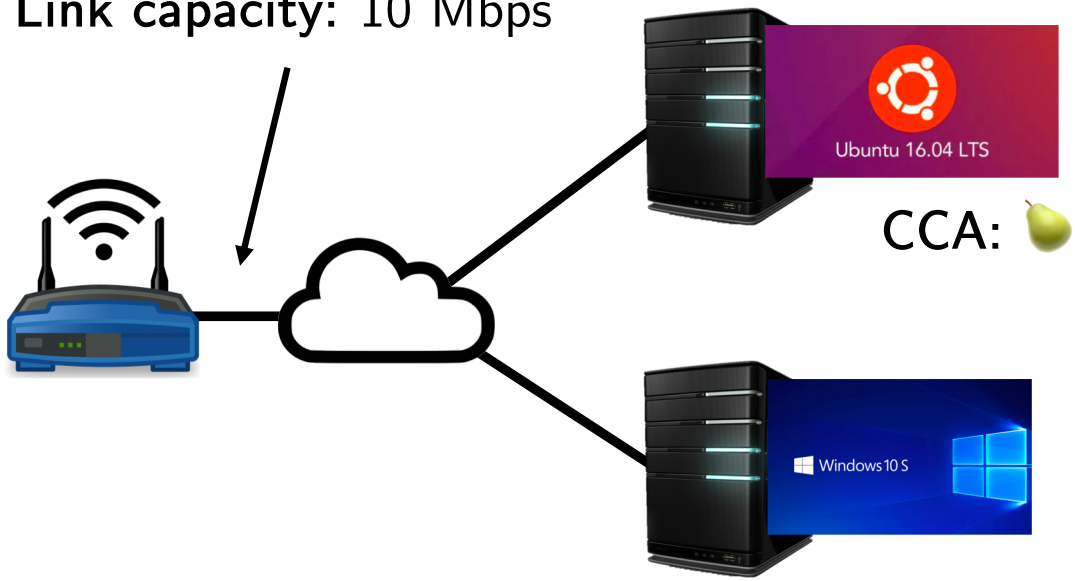
Download speed: 10 Mbps

Link capacity: 10 Mbps

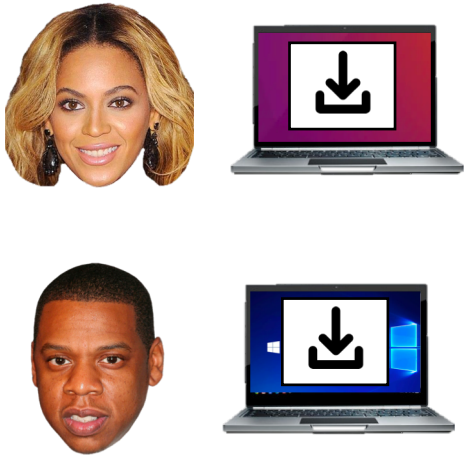
Download speed: 10 Mbps



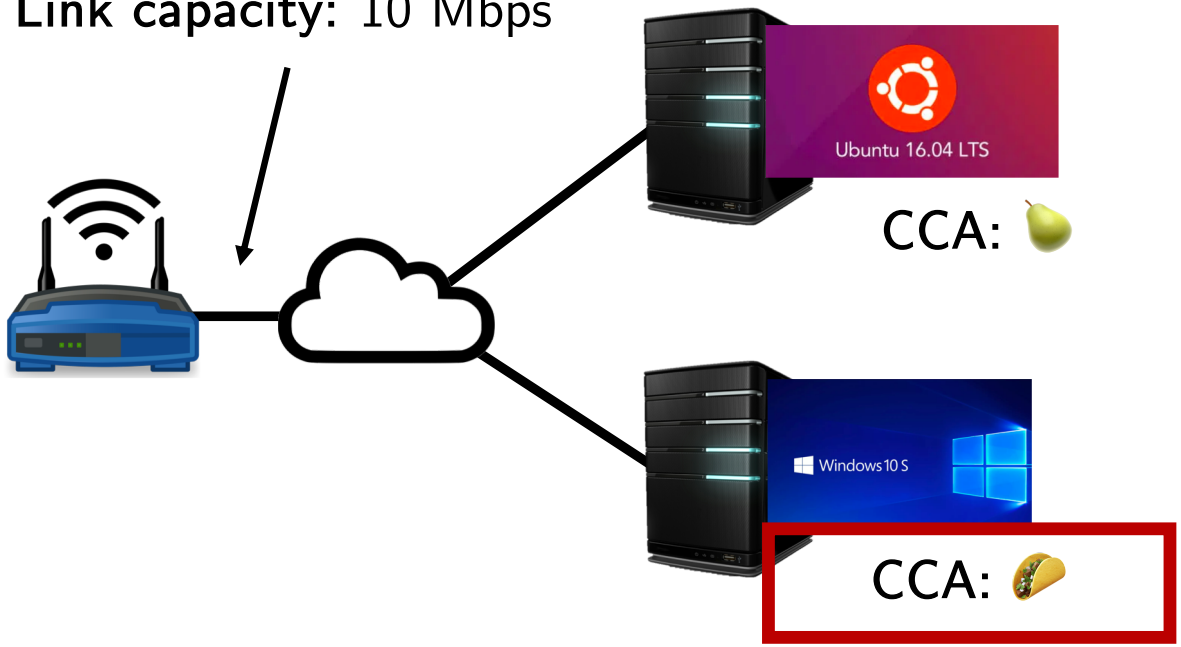
Link capacity: 10 Mbps

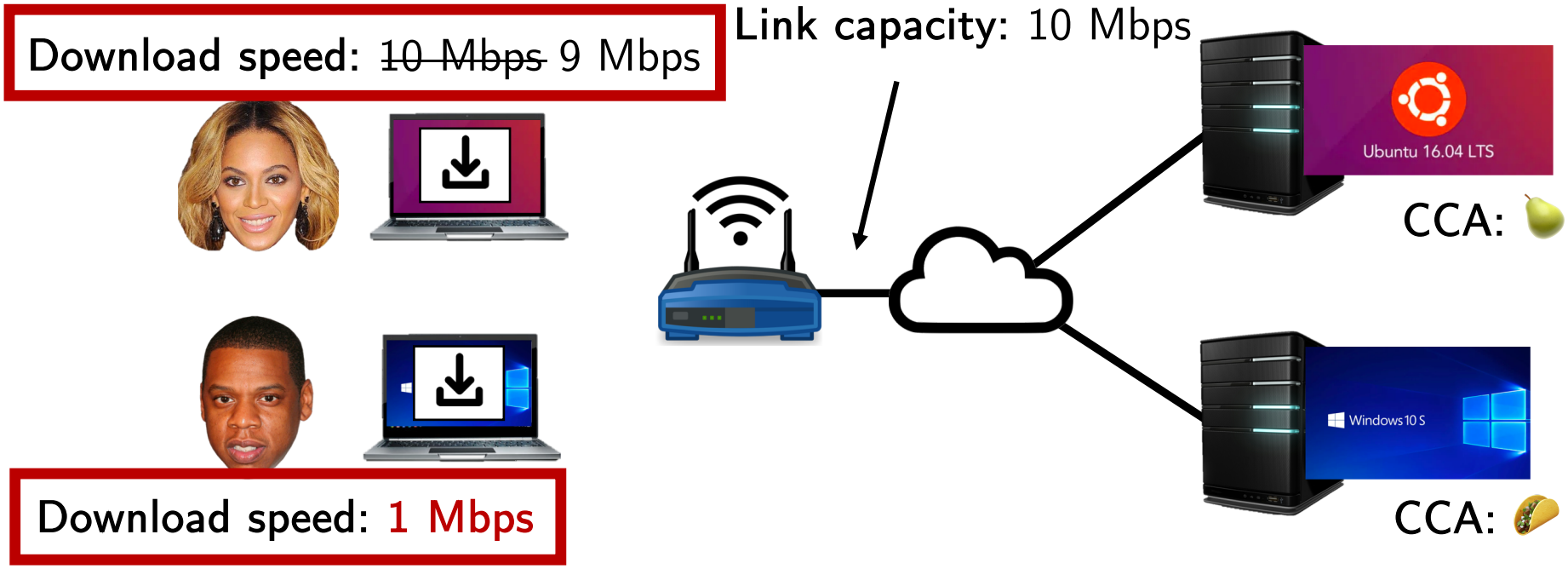


Download speed: 10 Mbps

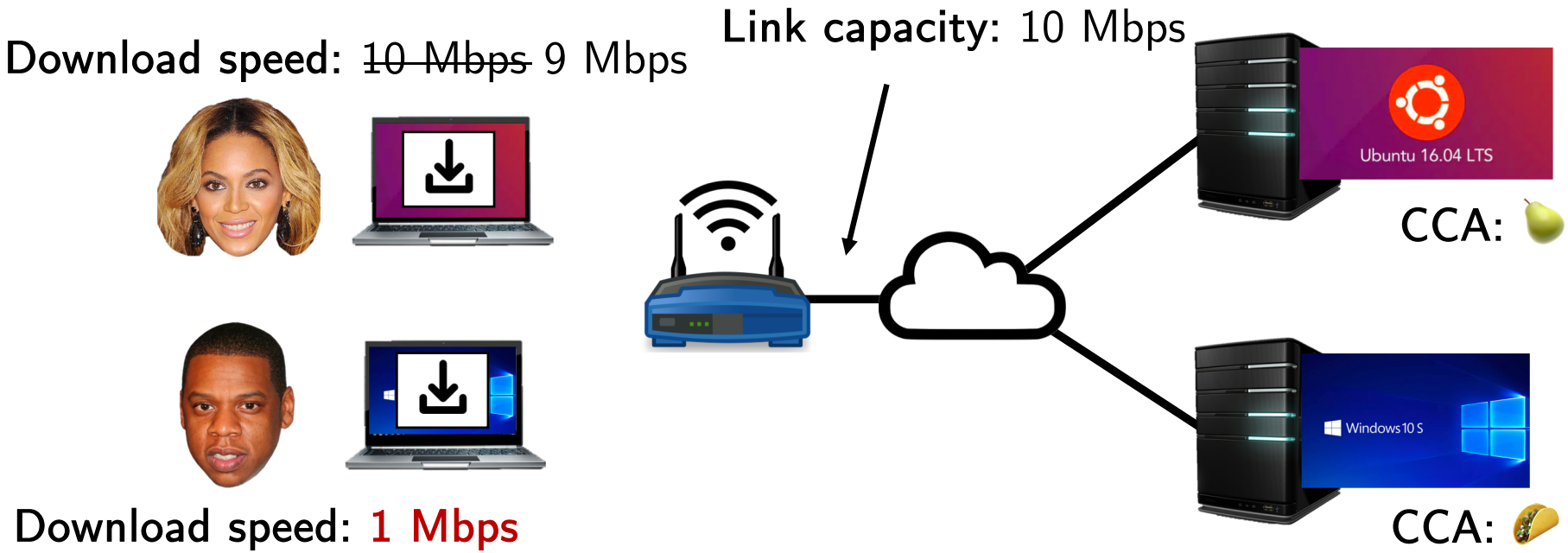


Link capacity: 10 Mbps



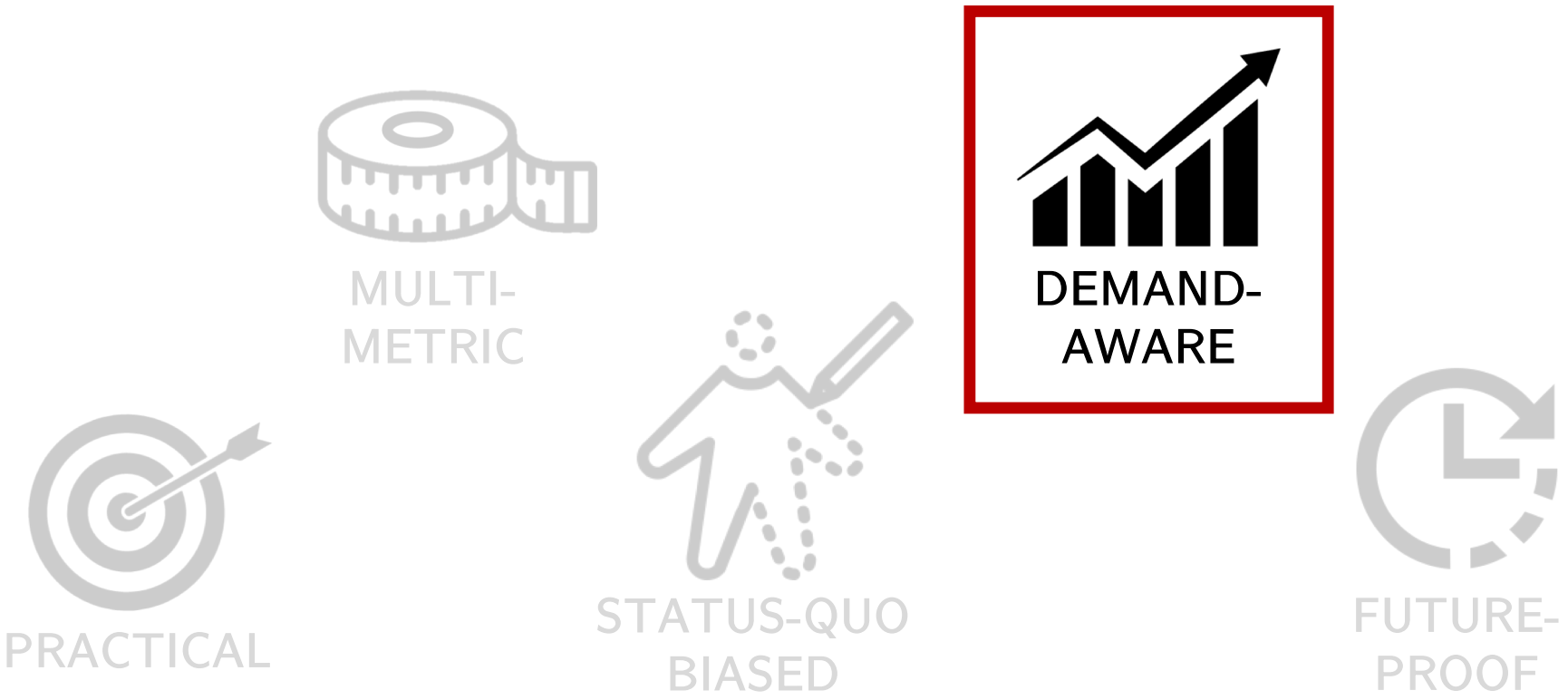

Download speed: ~~10 Mbps~~ 9 Mbps
Link capacity: 10 Mbps
Ubuntu 16.04 LTS
CCA:
Windows 10 S
CCA:
Download speed: 1 Mbps

A deployment threshold needs to be **status-quo biased**: based only on impact of 🌮 on 🍐, not vice-versa.



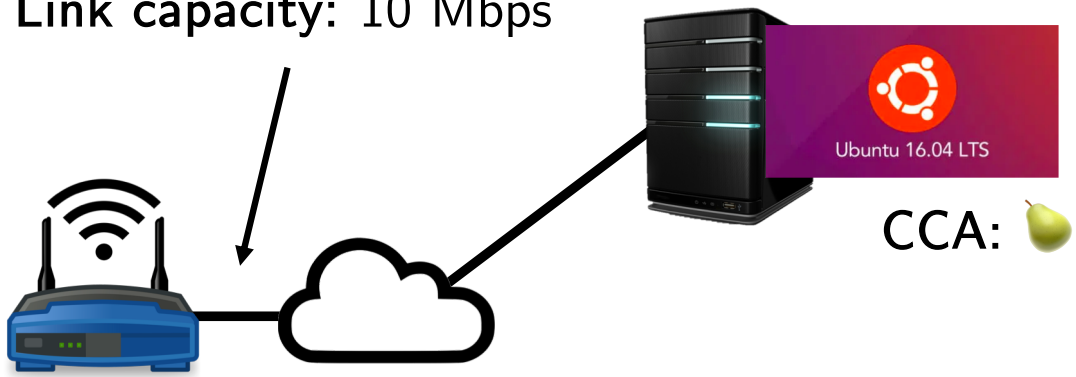
Jain's fairness index is not status-quo biased.

We identify 5 desirable properties for a deployment threshold.

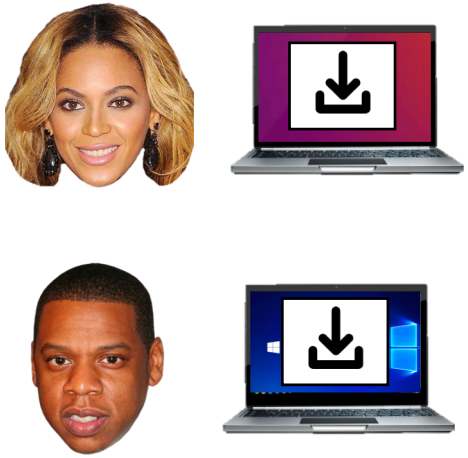


Download speed: 3 Mbps

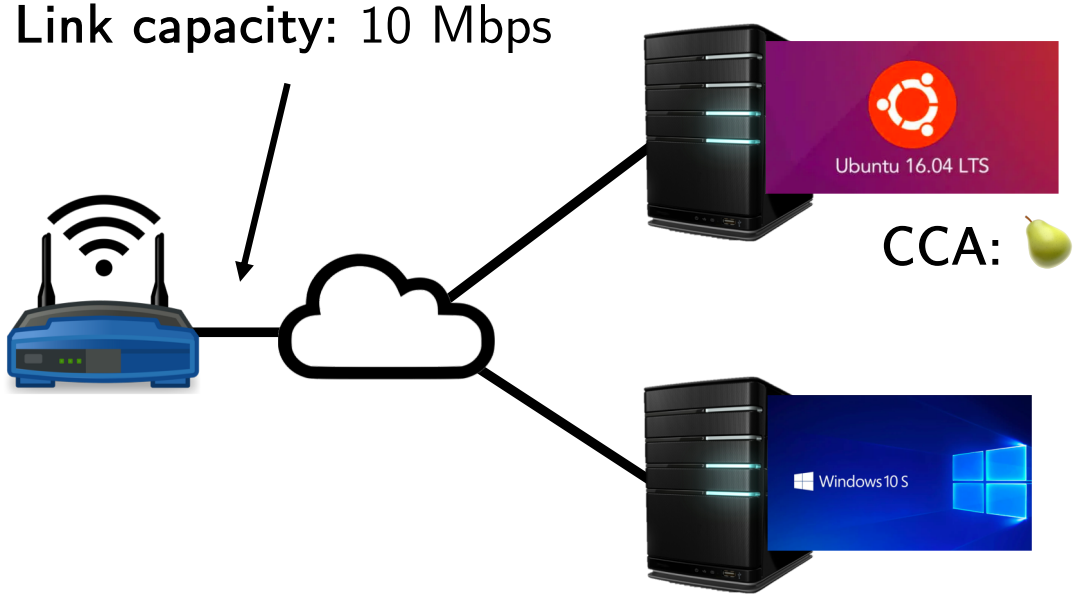
Link capacity: 10 Mbps



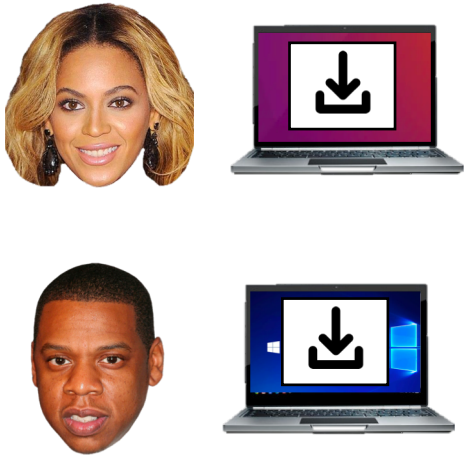
Download speed: 3 Mbps



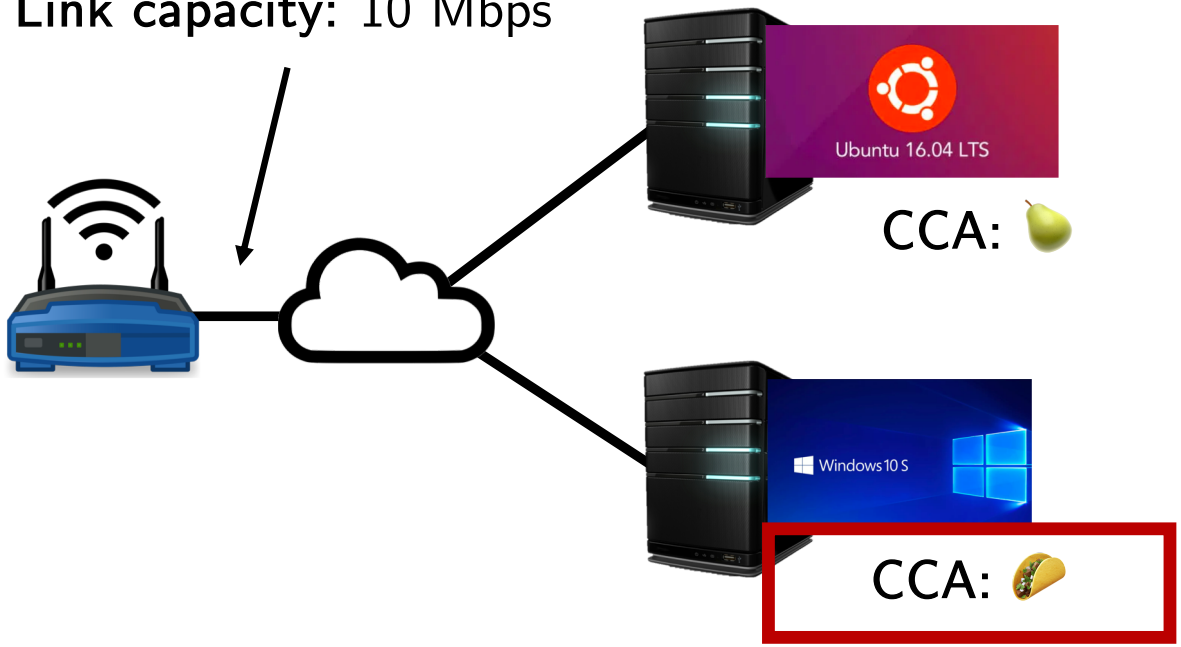
Link capacity: 10 Mbps



Download speed: 3 Mbps



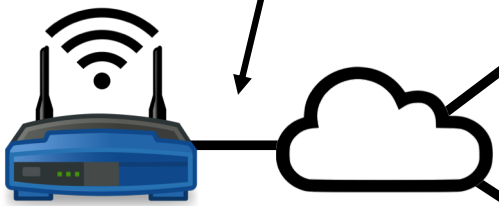
Link capacity: 10 Mbps



Download speed: 3 Mbps

Download speed: 7 Mbps

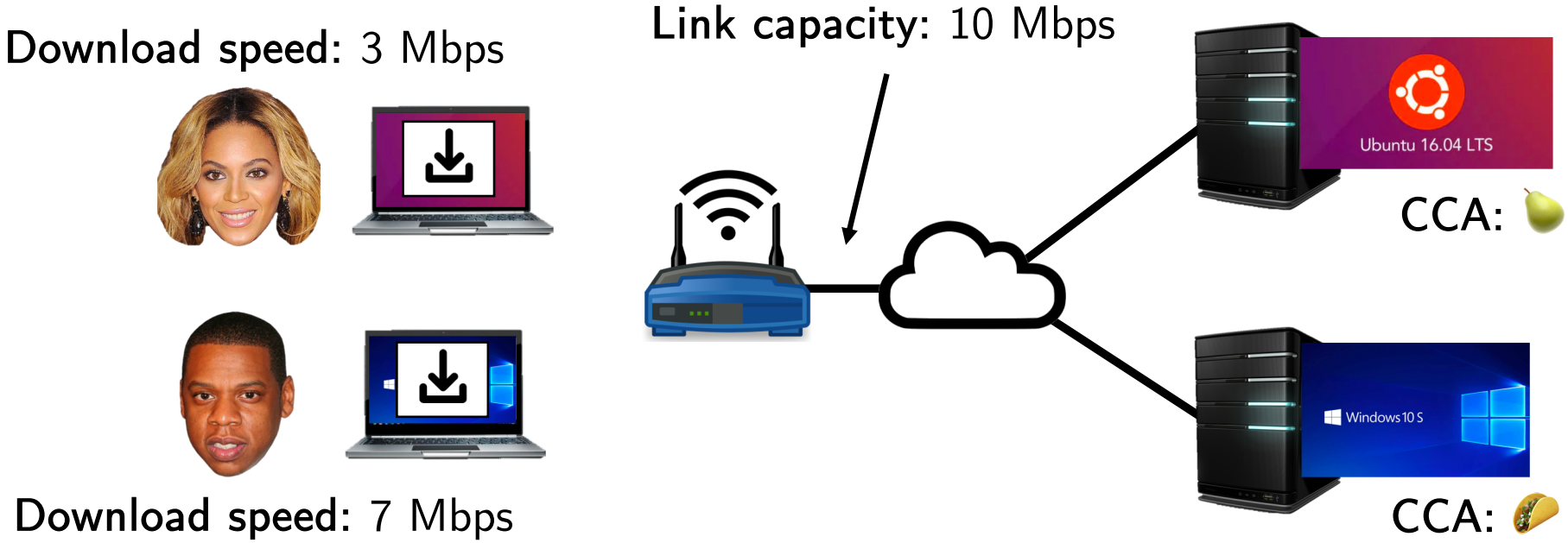
Link capacity: 10 Mbps



CCA: 🍐

CCA: 🌮

A deployment threshold needs to be demand-aware: do not penalize 🌮 when 🍐 has inherently poor performance.



Max-min fairness is demand aware, equal-rate fairness is not.

We identify 5 desirable properties for a deployment threshold.

MULTI-METRIC

DEMAND-AWARE

PRACTICAL

STATUS-QUO BIASED

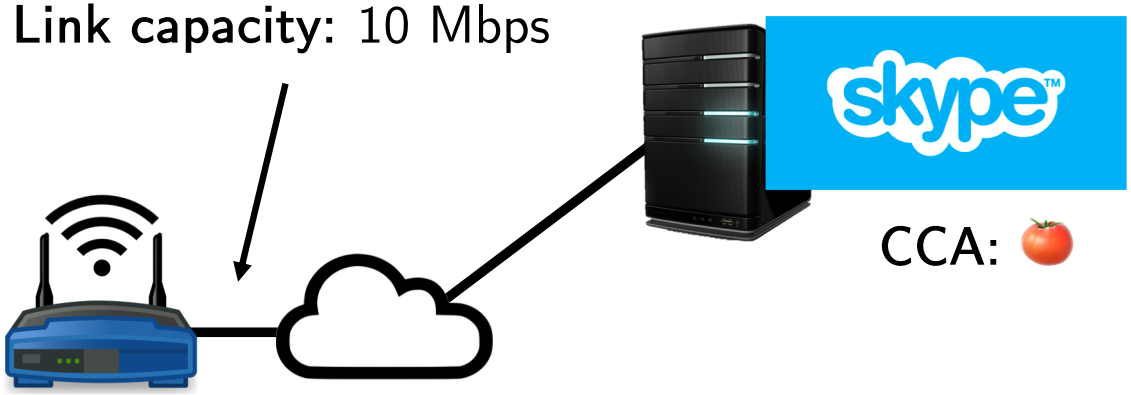
FUTURE-PROOF

A deployment threshold needs to be **future-proof**: useful on a future Internet where none of today's current CCAs are deployed.

A deployment threshold needs to be future-proof: useful on a future Internet where none of today's current CCAs are deployed.

Download speed: 1 Mbps

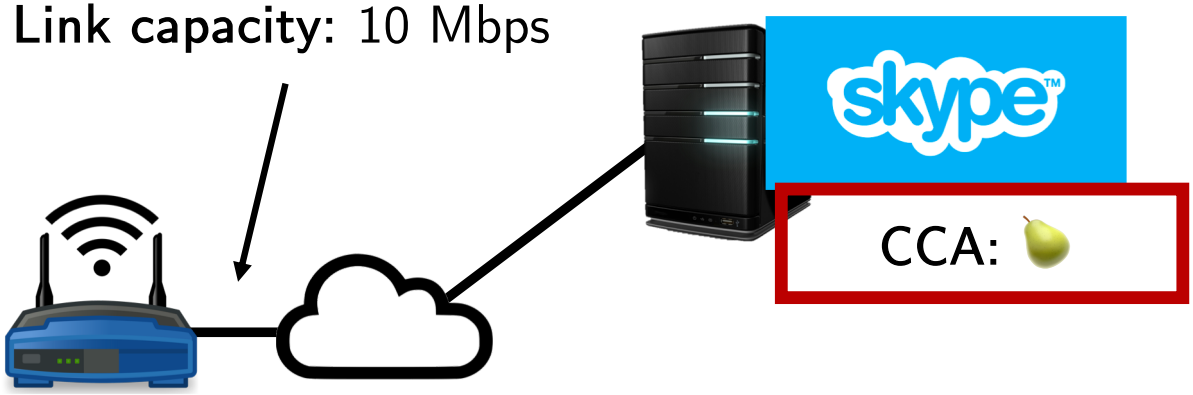
Link capacity: 10 Mbps



A deployment threshold needs to be future-proof: useful on a future Internet where none of today's current CCAs are deployed.

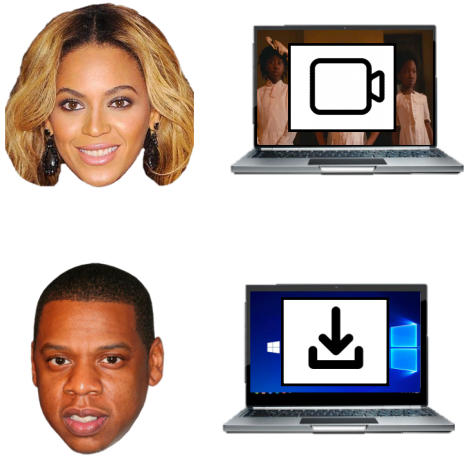
Download speed: 5 Mbps

Link capacity: 10 Mbps

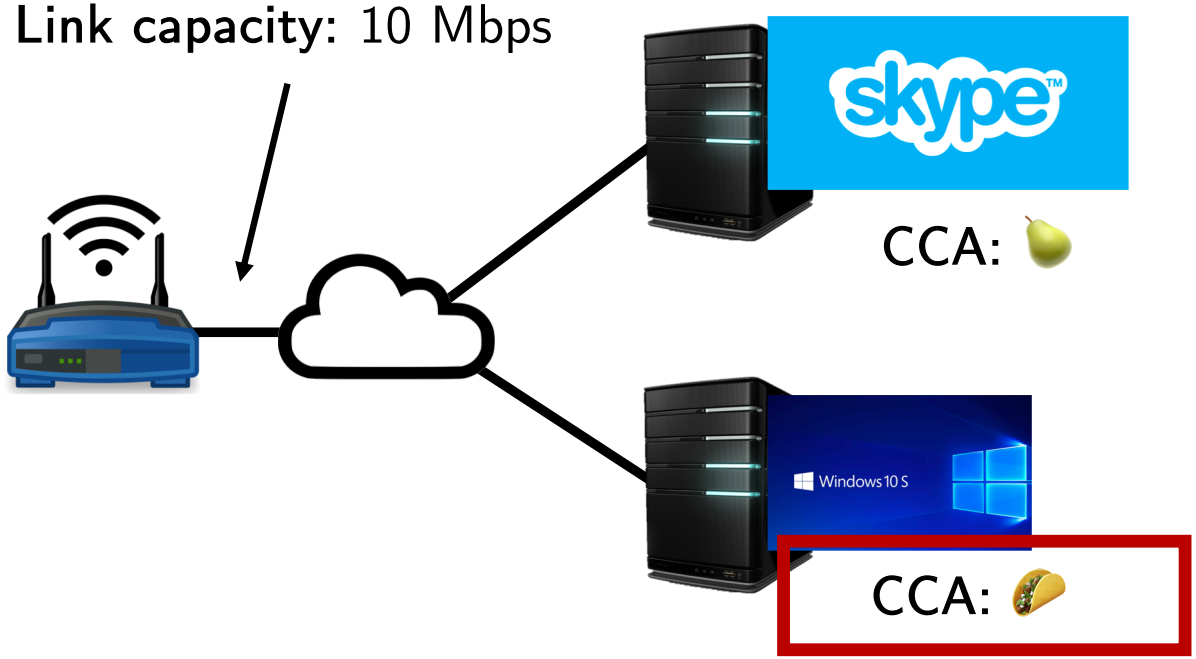


Does 🌮 need to be nice to 🍐 and 🍅 or just 🍐?

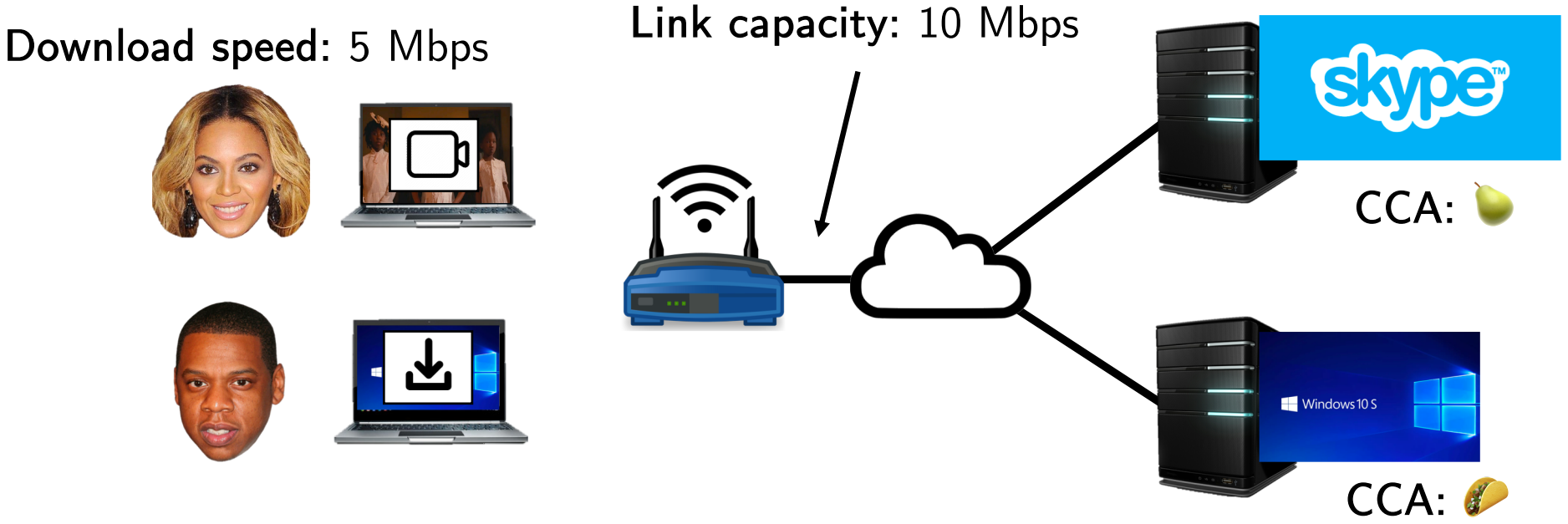
Download speed: 5 Mbps



Link capacity: 10 Mbps

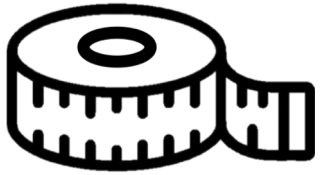


A future-proof threshold would only require 🌮 to be nice to 🍐



TCP-friendliness is not future-proof.

We identify 5 desirable properties for a deployment threshold.



MULTI-
METRIC

DEMAND-
AWARE

PRACTICAL

STATUS-QUO
BIASED

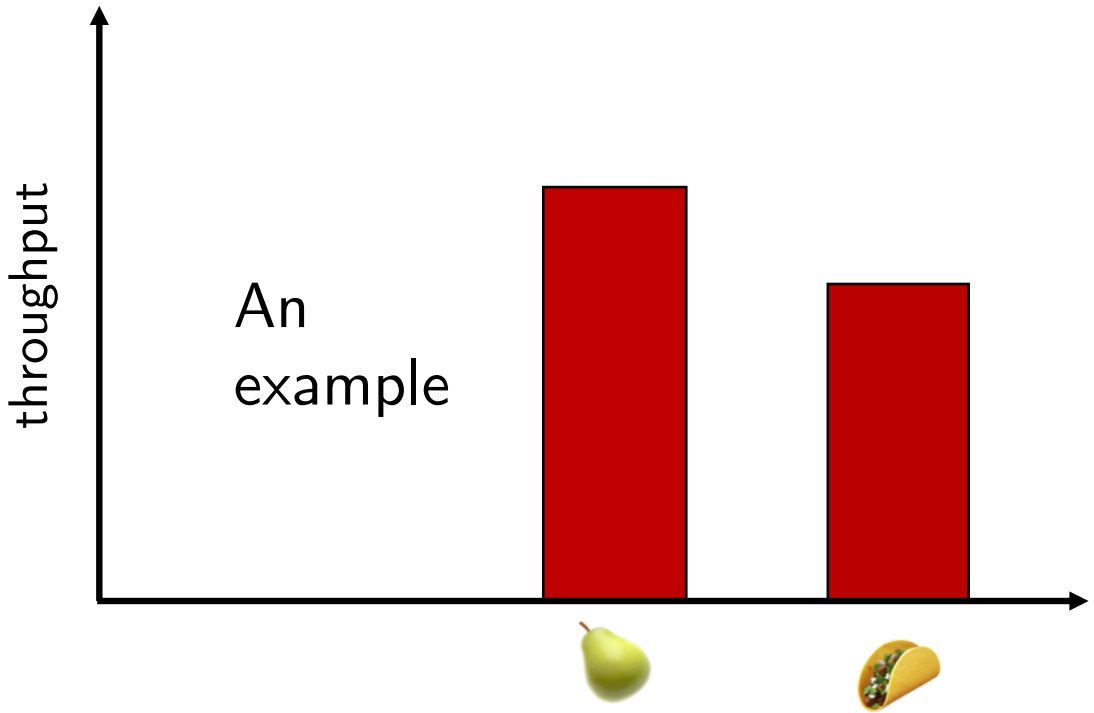
FUTURE-
PROOF

Outline:
1. What are desirable properties of a deployment threshold?

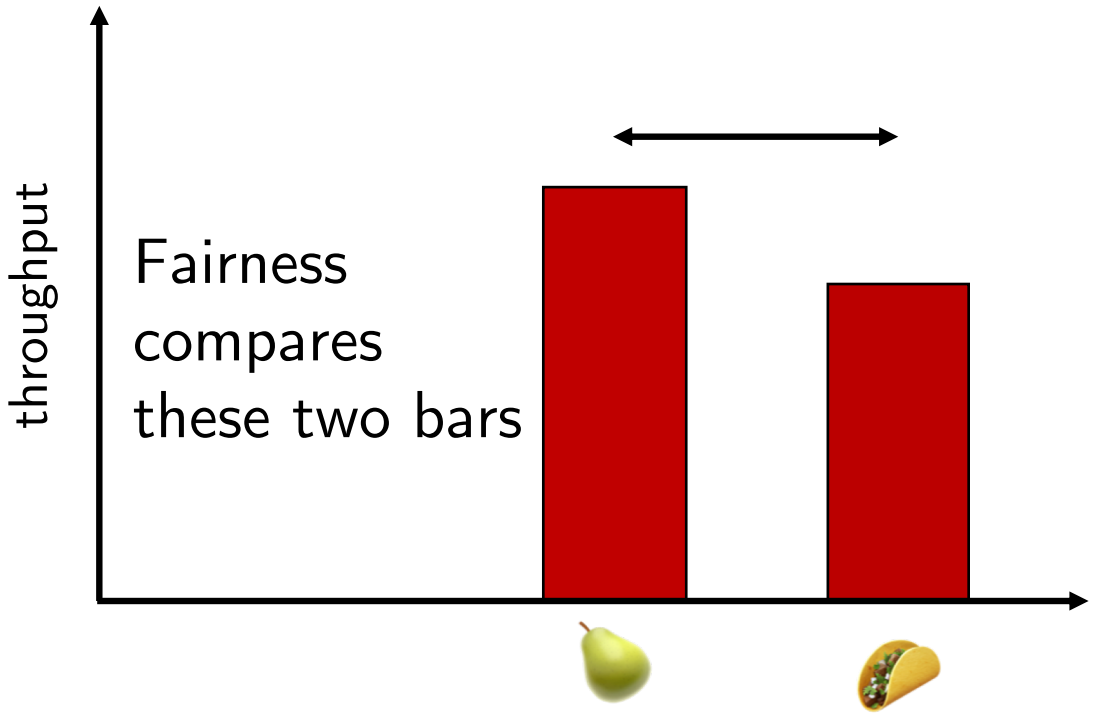
2. We define a new deployment threshold: harm.

When showing deployability: we run experiments of 🍐 vs. 🌮 and measure performance.

When showing deployability: we run experiments of 🍐 vs. 🌮 and measure performance.

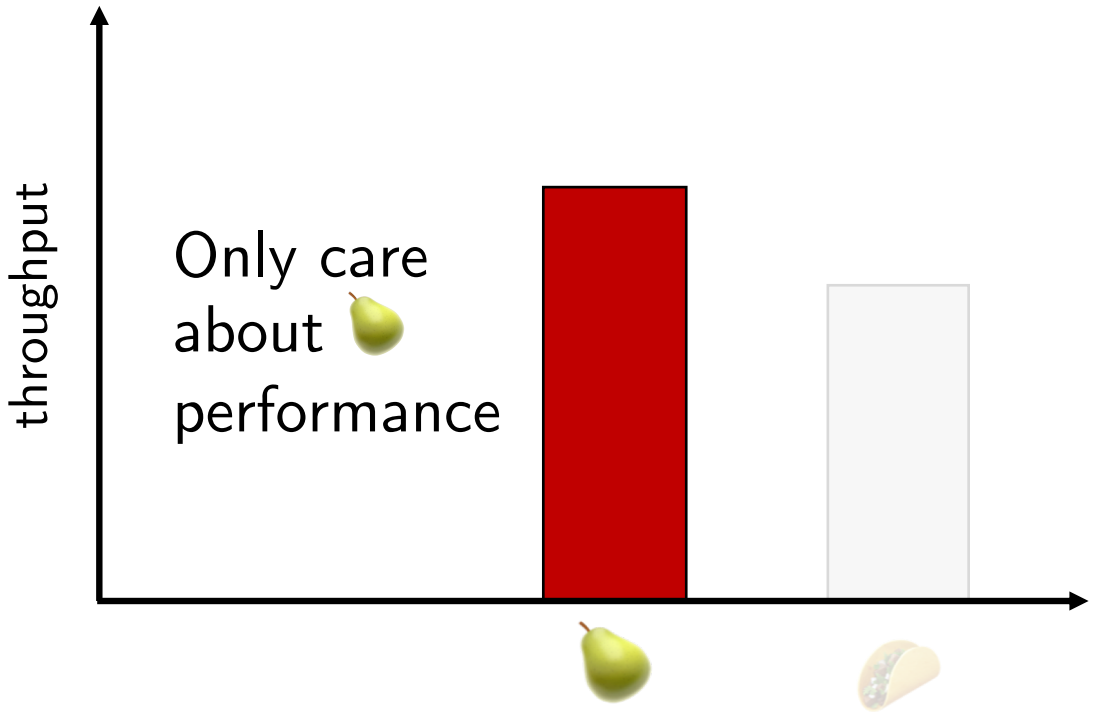


When showing deployability: we run experiments of 🍐 vs. 🌮 and measure performance.

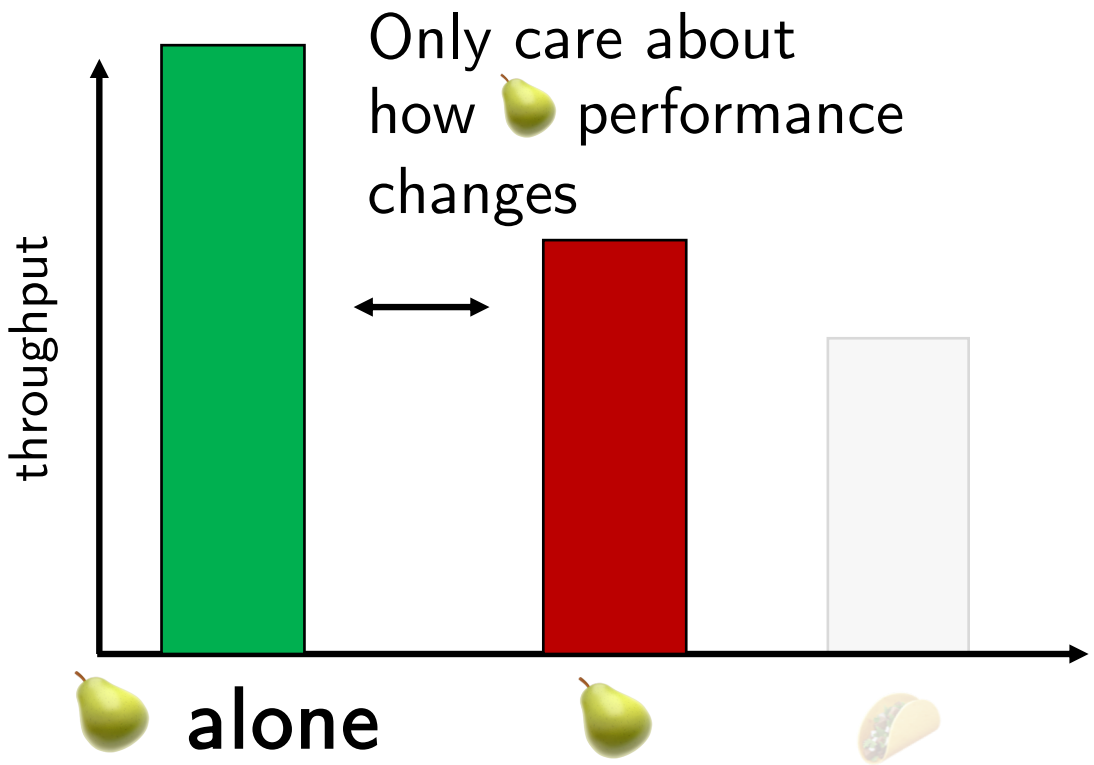


When showing deployability: we run experiments of 🍐 vs. 🌮 and measure performance.

When showing deployability: we run experiments of 🍐 vs. 🌮 and measure performance.



We want to **measure the impact** of 🌮 on 🍐 performance.



Our Proposal:

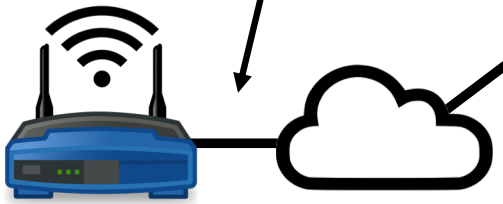
Deployment threshold should be based on how much harm 🌮 does to 🍐

This is 🍐 performance alone.

Latency: 5 ms
Download speed: 10 Mbps

🍐 alone

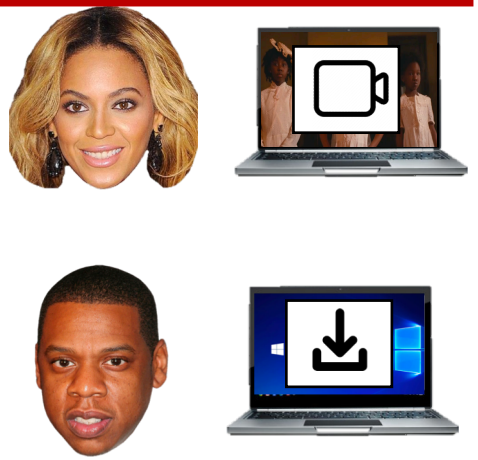
Link capacity: 10 Mbps



CCA: 🍐

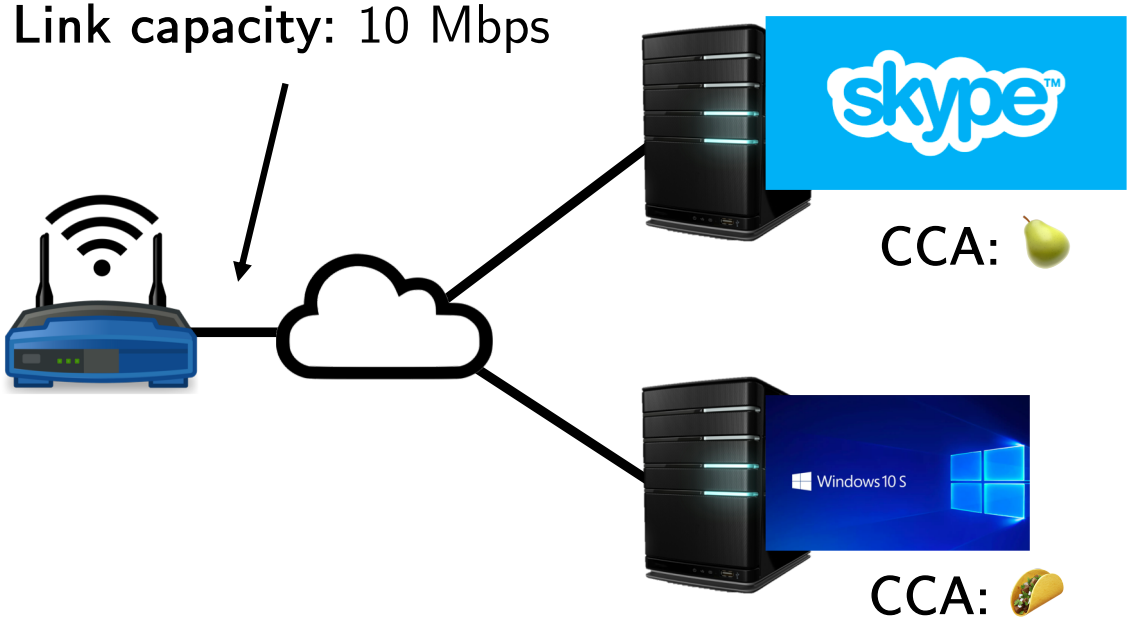
Harm measures the impact of 🌮 on 🍐 performance.

Latency: 100 ms
Download speed: 5 Mbps



Download speed: 5 Mbps

Link capacity: 10 Mbps



Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 alone: (x)

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

How to Compute Harm:
x = 🍐 solo performance (demand)

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone:** (x)

- **Latency**: 5 ms
  Download speed: 10 Mbps

🍐 **vs. 🌮:** (y)

- **Latency:** 100 ms
  Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone: (x)**

- **Latency**: 5 ms
  Download speed: 10 Mbps

🍐 **vs. 🌮: (y)**

- **Latency:** 100 ms
  Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x - y}{x}$

For "less is better" metrics (latency): $\frac{y - x}{y}$

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone: (x)**

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 **vs. 🌮: (y)**

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone: (x)**

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 **vs. 🌮: (y)**

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

Desirable threshold properties:

☐Practical ☐Demand-Aware ☐Status-Quo Biased **Multi-metric** ☐Future-Proof

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone: (x)**

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 **vs. 🌮: (y)**

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

Desirable threshold properties:

☐Practical ☐Demand-Aware **Status-Quo Biased** Multi-metric ☐Future-Proof

Harm is [0,1] where 0 is harmless and 1 is maximally harmful.

🍐 **alone: (x)**

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 **vs. 🌮: (y)**

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

How to Compute Harm:

$x =$ 🍐 solo performance (demand)

$y =$ 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

Desirable threshold properties:

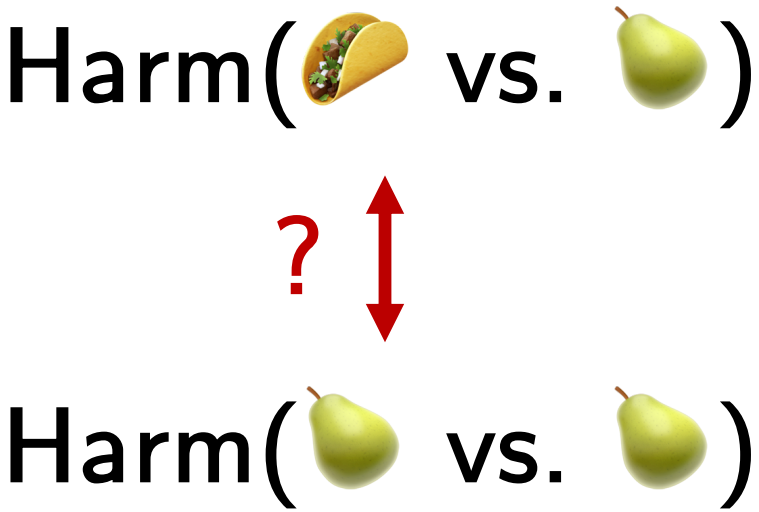
☐Practical **Demand-Aware** Status-Quo Biased Multi-metric ☐Future-Proof

But how much harm is OK?

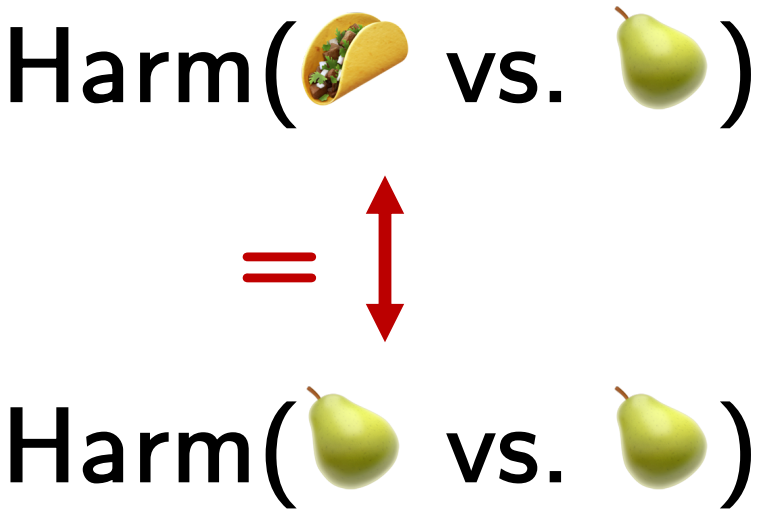
Key Insight:
A harm-based threshold:

🌮 should not harm 🍐 much more than 🍐 harms itself

Harm(🌮 vs. 🍐)

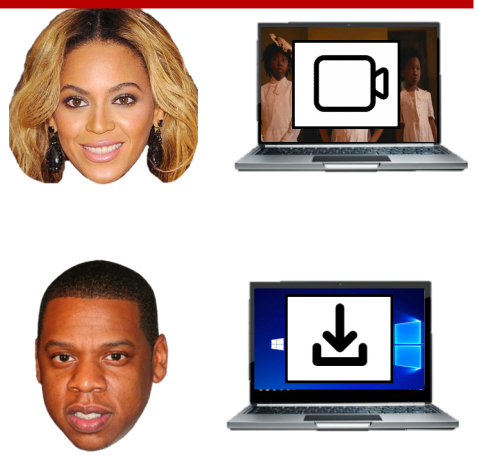

Harm(vs.)
?
Harm(vs.)

There are many possible thresholds based on harm (see paper!).
One possible harm-based threshold: equivalent-bounded harm.



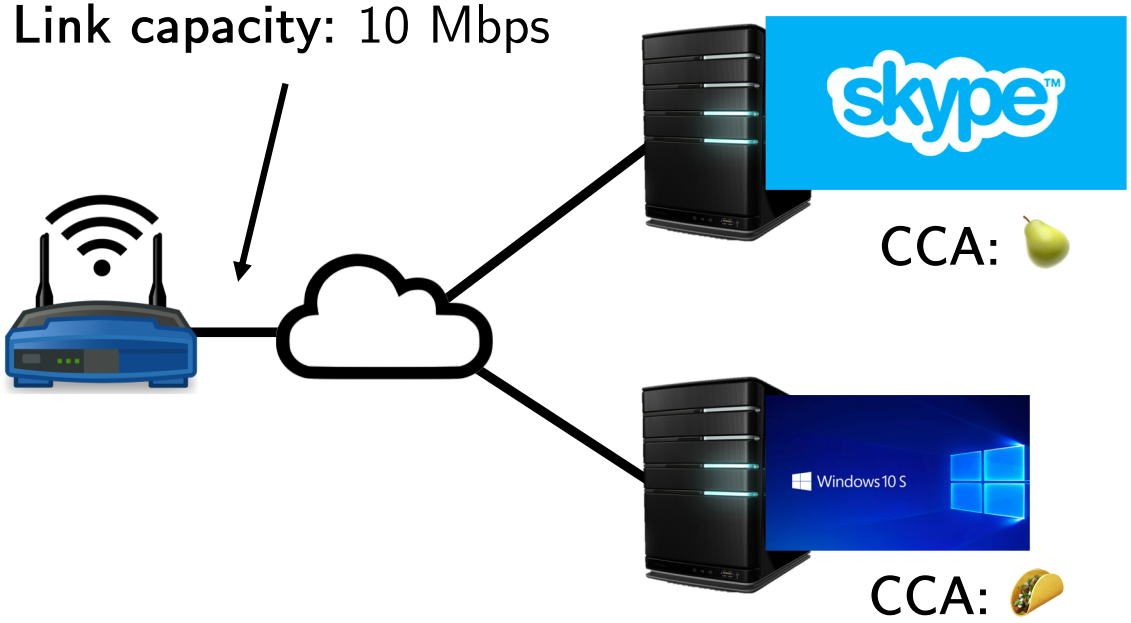
One possible harm-based threshold: equivalent-bounded harm.

Latency: 100 ms
Download speed: 5 Mbps



Download speed: 5 Mbps

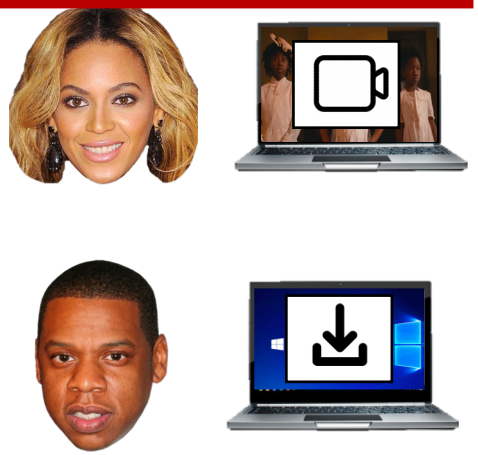
Link capacity: 10 Mbps



Harm(🌮 vs. 🍐)

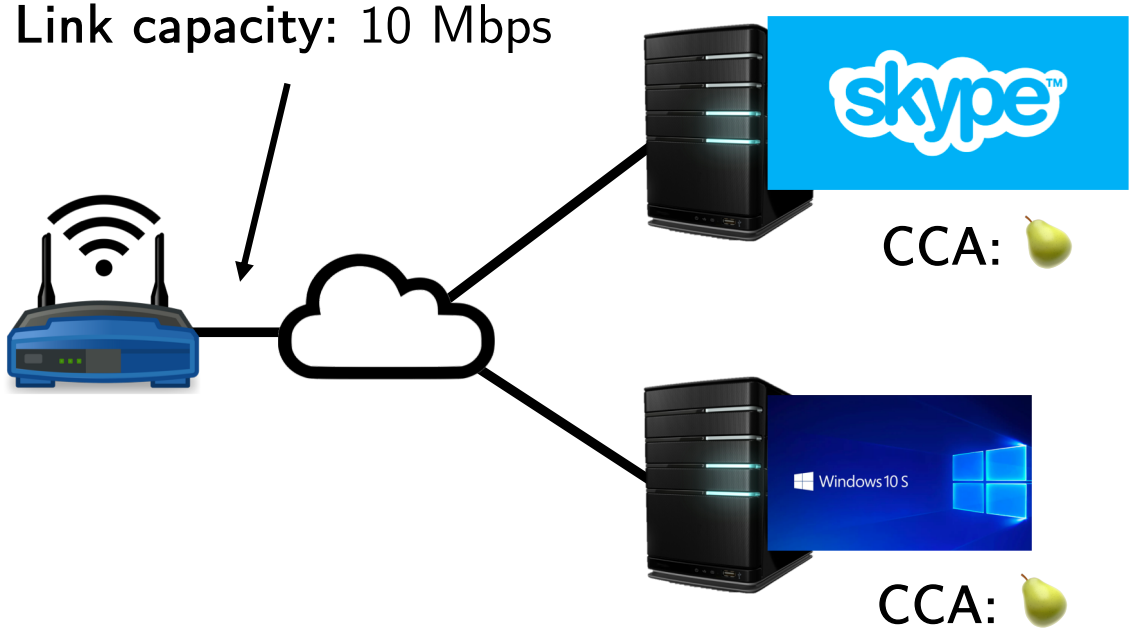
One possible harm-based threshold: equivalent-bounded harm.

Latency: 10 ms
Download speed: 5 Mbps



Download speed: 5 Mbps

Link capacity: 10 Mbps



Harm(vs.)

🍐 alone:

- **Latency**: 5 ms
  Download speed: 10 Mbps

🍐 vs. 🌮:

- **Latency:** 100 ms
  Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

🍐 alone:

Latency: 5 ms
Download speed: 10 Mbps

🍐 vs. 🌮:

Latency: 100 ms
Download speed: 5 Mbps

🍐 vs. 🍐:

Latency: 10 ms
Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

🍐 caused throughput harm: $\frac{10-5}{10} = .50$

🍐 caused latency harm: $\frac{10-5}{10} = .50$

🍐 alone:

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 vs. 🌮:

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

🍐 vs. 🍐:

🍐 **Latency:** 10 ms
Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

🍐 caused throughput harm: $\frac{10-5}{10} = .50$

🍐 caused latency harm: $\frac{10-5}{10} = .50$

🍐 alone:

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 vs. 🌮:

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

🍐 vs. 🍐:

🍐 **Latency:** 10 ms
Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

🍐 caused throughput harm: $\frac{10-5}{10} = .50$

🍐 caused latency harm: $\frac{10-5}{10} = .50$

🍐 alone:

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 vs. 🌮:

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

🍐 vs. 🍐:

🍐 **Latency:** 10 ms
Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

🍐 caused throughput harm: $\frac{10-5}{10} = .50$

🍐 caused latency harm: $\frac{10-5}{10} = .50$

Desirable threshold properties:

Practical Demand-Aware Status-Quo Biased Multi-metric ☐Future-Proof

🍐 alone:

🍐 **Latency**: 5 ms
Download speed: 10 Mbps

🍐 vs. 🌮:

🍐 **Latency:** 100 ms
Download speed: 5 Mbps

🍐 vs. 🍐:

🍐 **Latency:** 10 ms
Download speed: 5 Mbps

How to Compute Harm:

x = 🍐 solo performance (demand)

y = 🍐 performance competing with 🌮

For "more is better" metrics (throughput): $\frac{x-y}{x}$

For "less is better" metrics (latency): $\frac{y-x}{y}$

Example:

🌮 caused throughput harm: $\frac{10-5}{10} = .50$

🌮 caused latency harm: $\frac{100-5}{100} = .95$

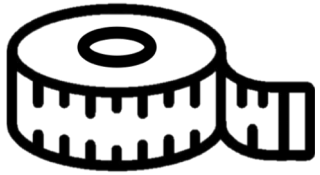
🍐 caused throughput harm: $\frac{10-5}{10} = .50$

🍐 caused latency harm: $\frac{10-5}{10} = .50$

Desirable threshold properties:

Practical Demand-Aware Status-Quo Biased Multi-metric **Future-Proof**

Is equivalent-bounded harm the answer? It meets all of our criteria.



MULTI-
METRIC

DEMAND-
AWARE

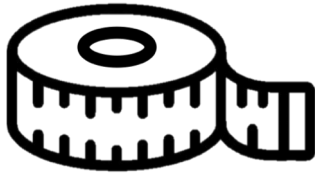
PRACTICAL

STATUS-QUO
BIASED

FUTURE-
PROOF

Fairness and TCP-friendliness do not.

Is equivalent-bounded harm the answer? But has issues.



MULTI-
METRIC

DEMAND-
AWARE

PRACTICAL

STATUS-QUO
BIASED

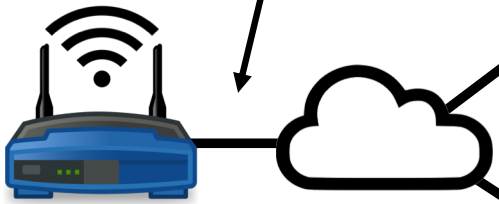
FUTURE-
PROOF

Fairness and TCP-friendliness do not.

Download speed: 7 Mbps

Download speed: 3 Mbps

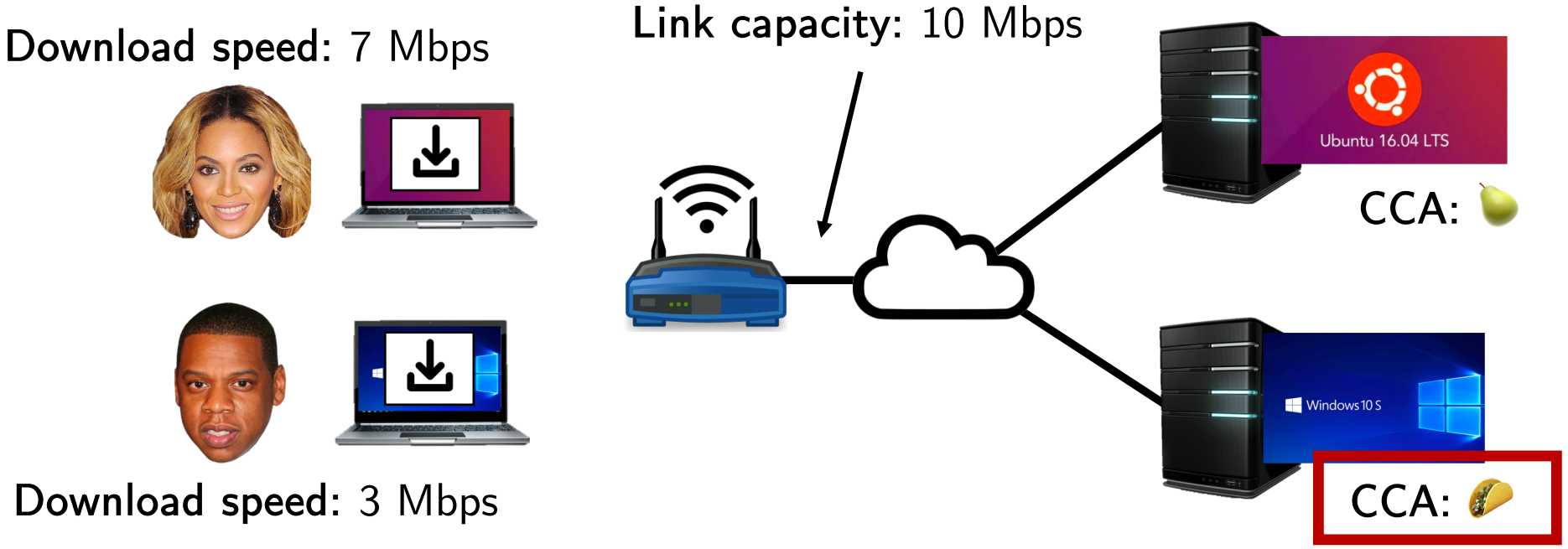
Link capacity: 10 Mbps



CCA:

CCA:

Could 🌮 improve this imbalance? Equivalent-bounded harm says no.



Other open questions:

1. Alternatives to equivalent-bounded harm?
2. Given a distribution of results, is there some ‘leeway in harm’? Should worry about average or worst case results?
3. What are the right workloads and networks for deployability testing?
4. How widely deployed must a legacy CCA be in order to merit protection by our threshold?
5. If we have a threshold, should it be enforced? If so, how?

While we haven't settled (yet) on the perfect threshold, here is what we do believe...

Fairness is not working as a practical threshold.

We need to stop making excuses for why our new algorithms are not meeting an unrealistic goal.

Reasoning about harm is the right way forward to derive a new threshold.

Beyond Jain's Fairness Index: Setting The Bar For the Deployment of Congestion Control Algorithms

Ranysha Ware
rware@cs.cmu.edu
@ranyshware

The Bar For Deployment: Do no more harm to the status quo than it does to itself.

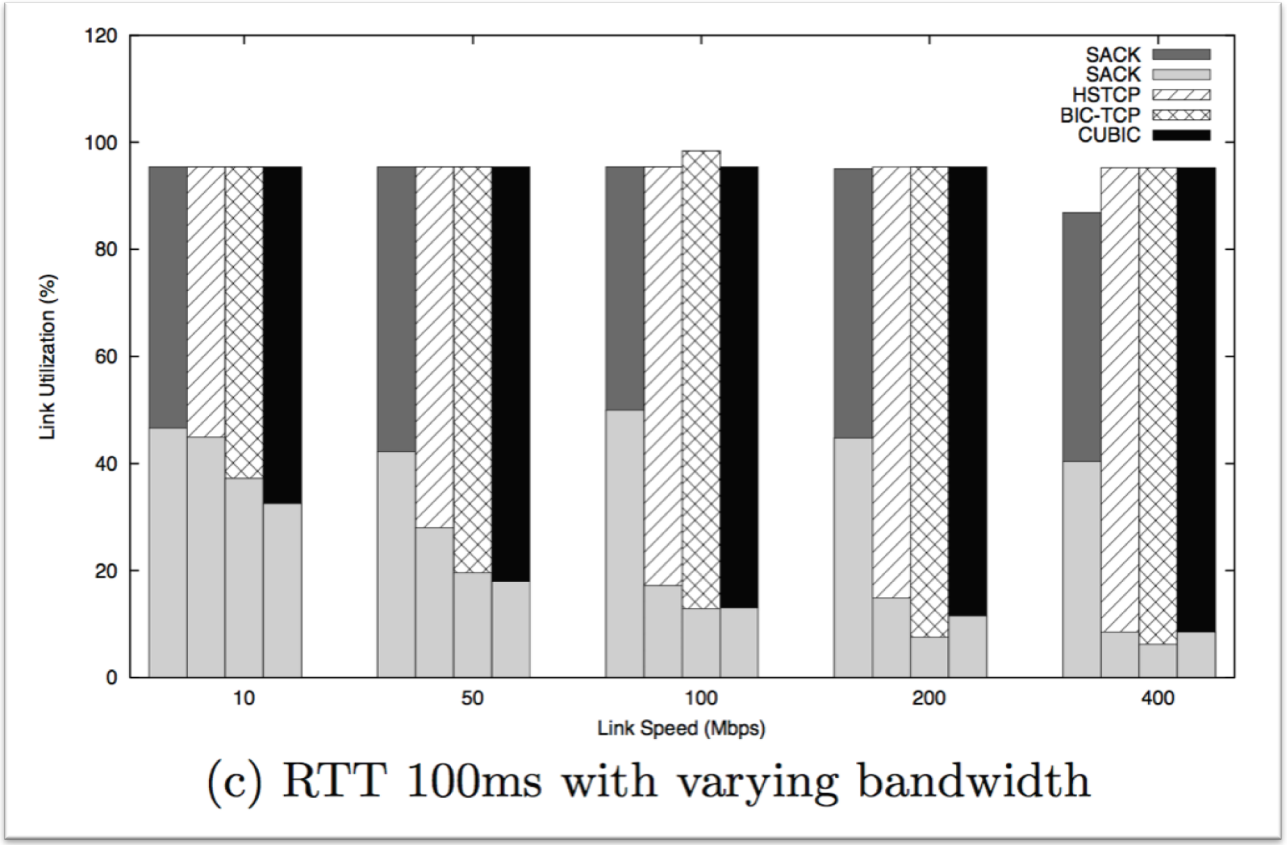
Some open questions:

1. Alternative to equivalent-bounded harm?
2. Given a distribution of results, is there some 'leeway in harm'? Should worry about average or worst case results?
3. What are the right workloads and networks for deployability testing?

BACKUP SLIDES

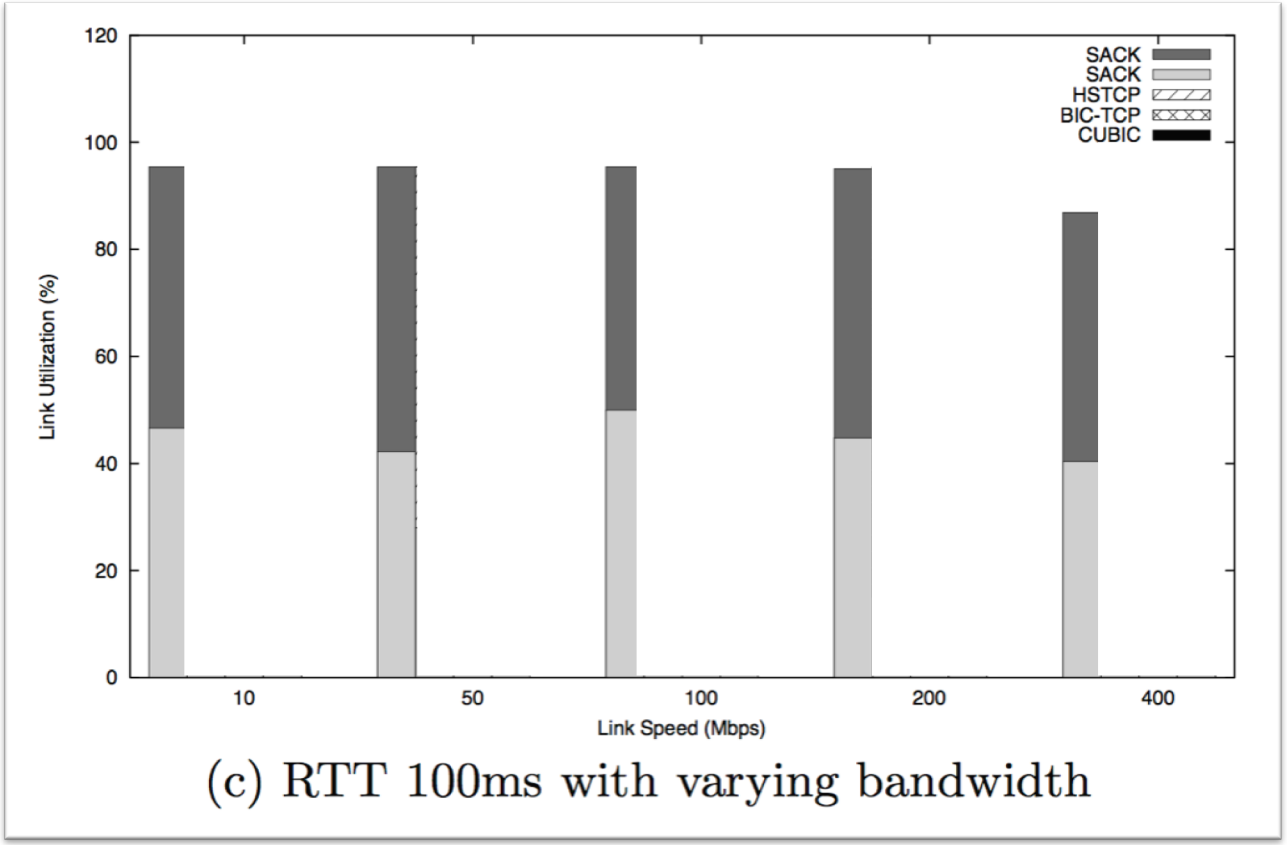
Every algorithm is unfair?

Example of unfair outcomes: Cubic is unfair to Reno.



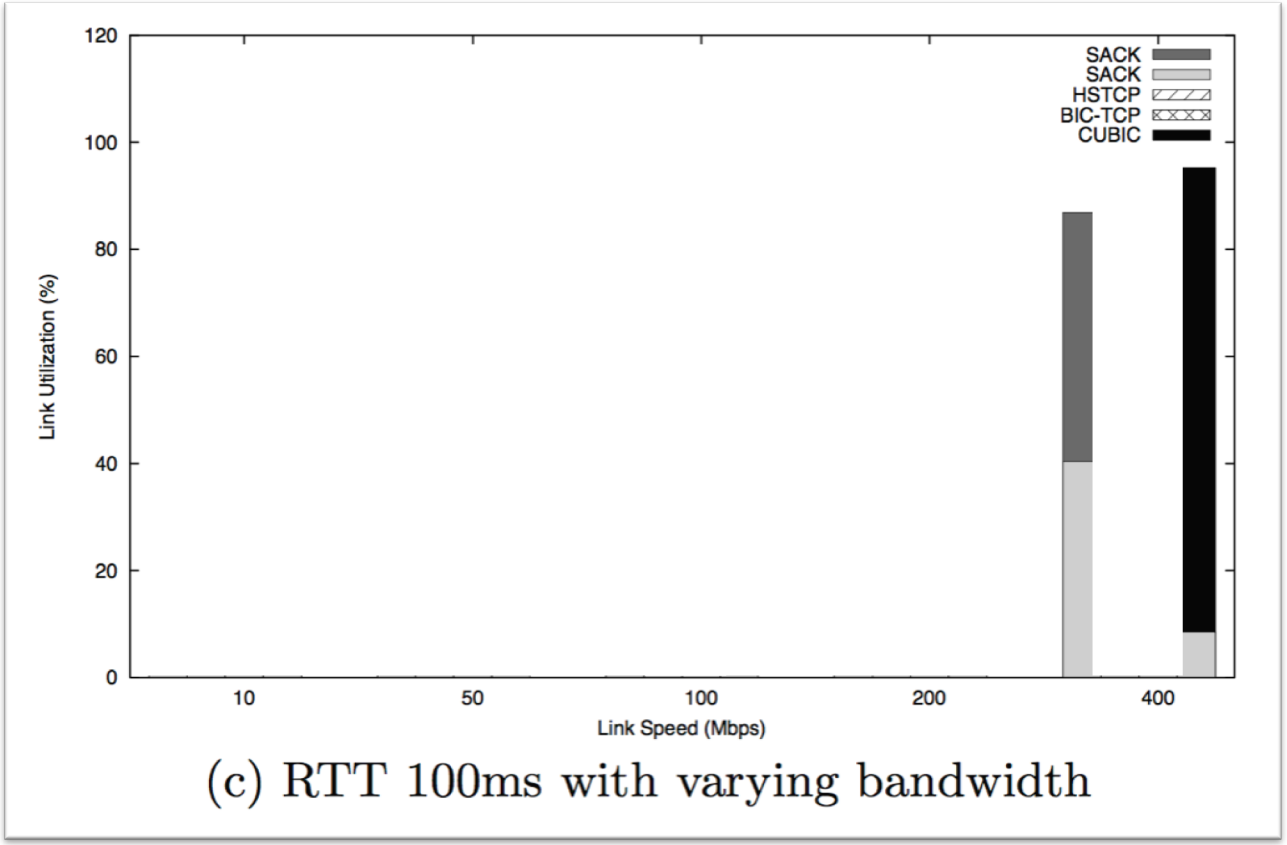
(c) RTT 100ms with varying bandwidth

Example of unfair outcomes: Cubic is unfair to Reno.



(c) RTT 100ms with varying bandwidth

Example of unfair to outcomes: Cubic is unfair to Reno.



(c) RTT 100ms with varying bandwidth

What is TCP-friendliness?

A mimicry-based threshold: If 🌮 mimics the behavior of 🍐 then 🌮 is deployable.

TCP-friendliness: A TCP friendly flow should react to loss the same way that TCP Reno does such that

$$BW < \left(\frac{MSS}{RTT}\right)\frac{1}{\sqrt{p}}$$

TCP-friendliness: A TCP friendly flow should react to loss the same way that TCP Reno does such that

$$BW < \left(\frac{MSS}{RTT}\right) \frac{1}{\sqrt{p}}$$

What do you mean by status-quo?

There are some applications that are more popular than others.

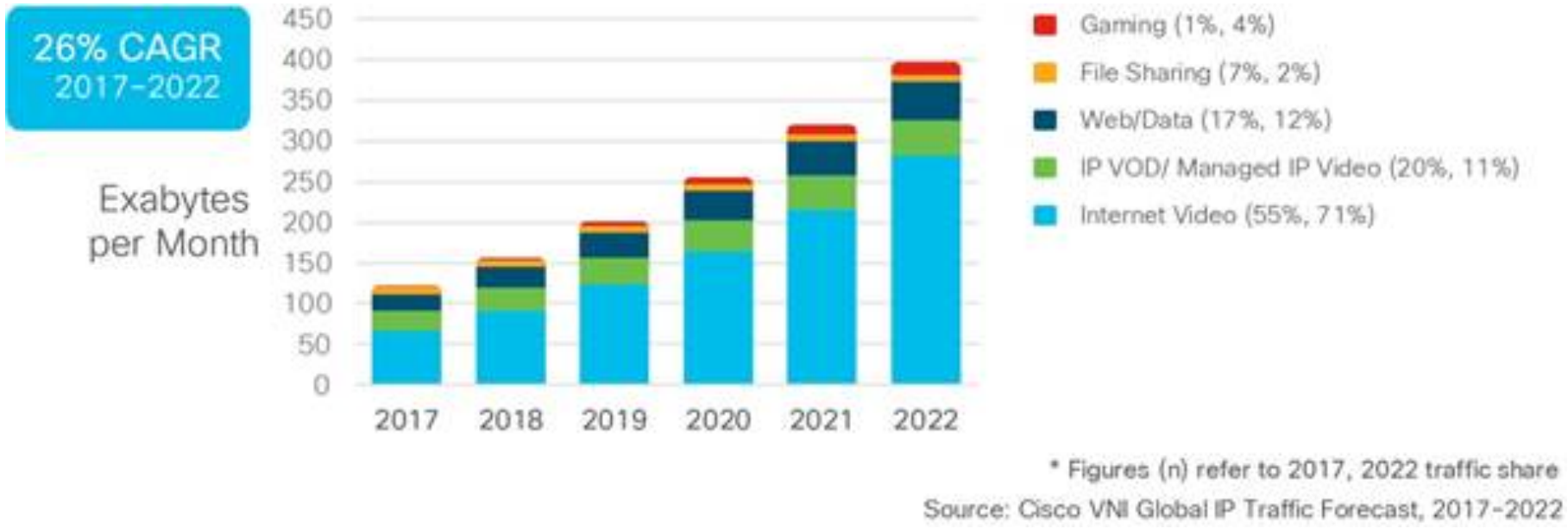


Figure: Internet Video is already more than half of all Internet traffic

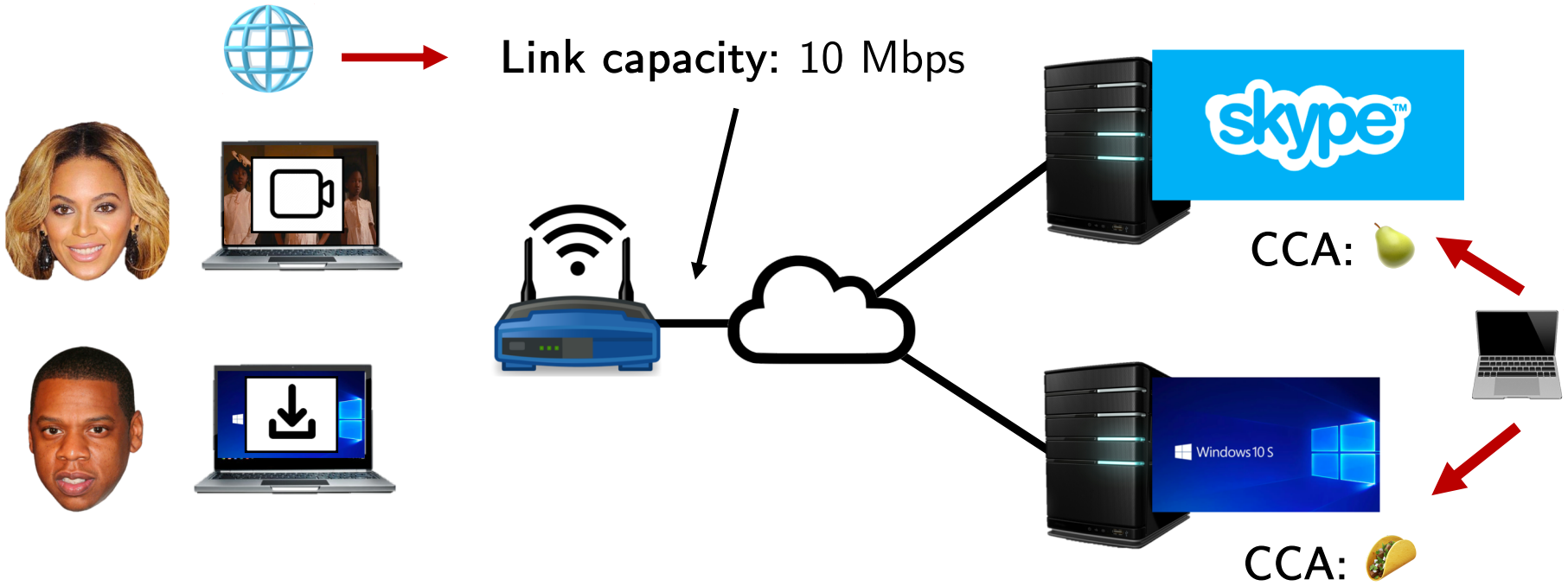
Throughout this talk, this is how we defined harm:

Harm(🌮 vs. 🍐)

?

Harm(🍐 vs. 🍐)

In the paper, we define harm also as a function of the network conditions 🌐 and workload 💻.



In the paper, we define harm also as a function of the network conditions 🌐 and workload 💻.

Harm(🌮 vs. 🍐, 🌐, 💻)

?

Harm(🍐 vs. 🍐, 🌐, 💻)