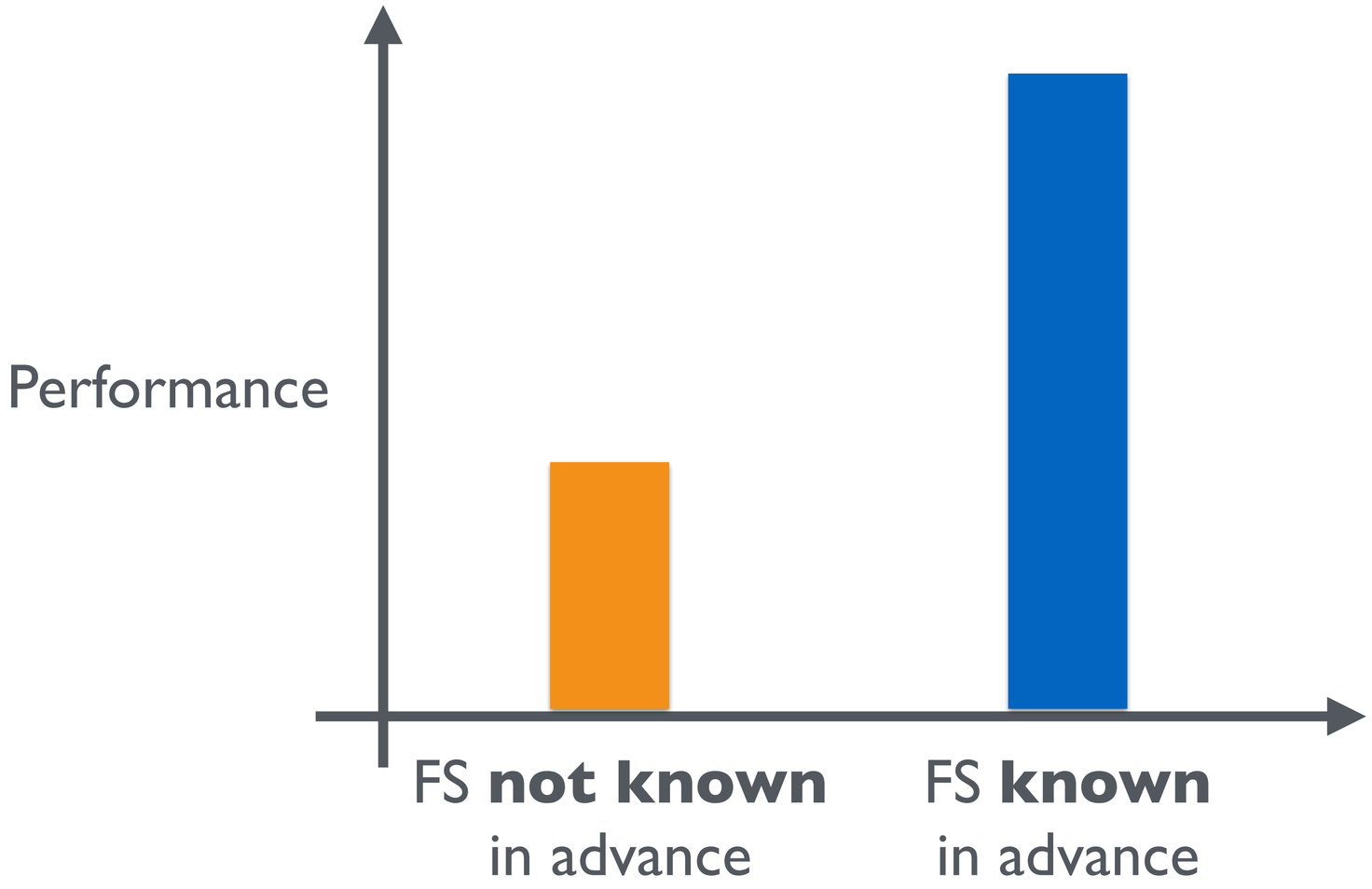
Is advance knowledge of flow sizes a plausible assumption?

Vojislav Dukic, Sangeetha Abdu Jyothi, Bojan Karlas,
Muhsen Owaida, Ce Zhang, Ankit Singla

Flow size information can increase efficiency



Flow size information can increase efficiency

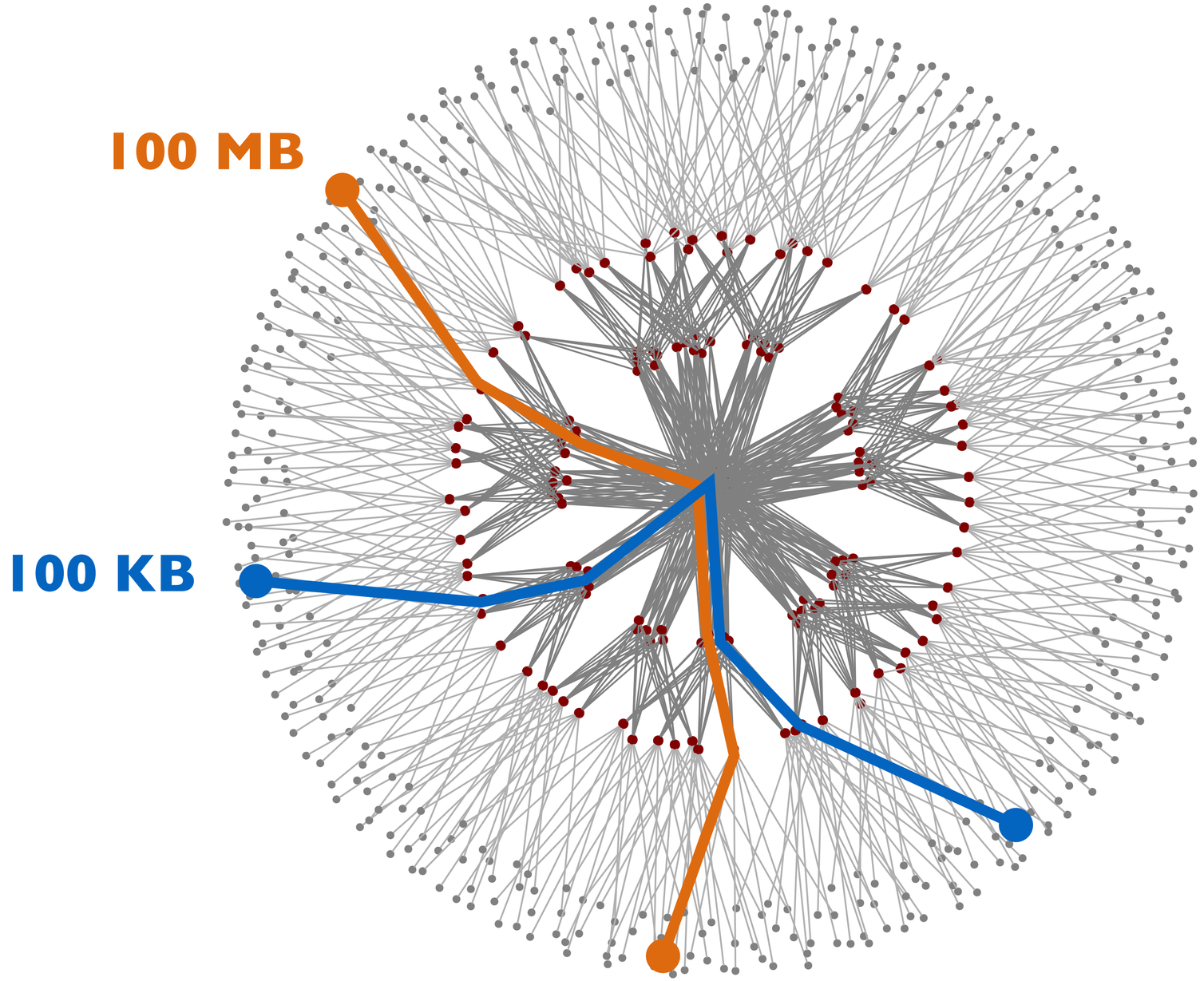
…fine-grained circuit scheduling

…packet scheduling within switches

…deadline-aware prioritization

…adaptive routing

…congestion control schemes


100 MB
100 KB

Packet scheduling at a switch queue



Typically, first-in first-out

Packet scheduling at a switch queue



Typically, first-in first-out

Packet scheduling at a switch queue



But could apply shortest job first idea!

Packet scheduling at a switch queue



Least remaining bytes first

[*pFabric: Minimal Near-Optimal Datacenter Transport; SIGCOMM '13]

Packet scheduling at a switch queue



Least remaining bytes first

[*pFabric: Minimal Near-Optimal Datacenter Transport; SIGCOMM '13]

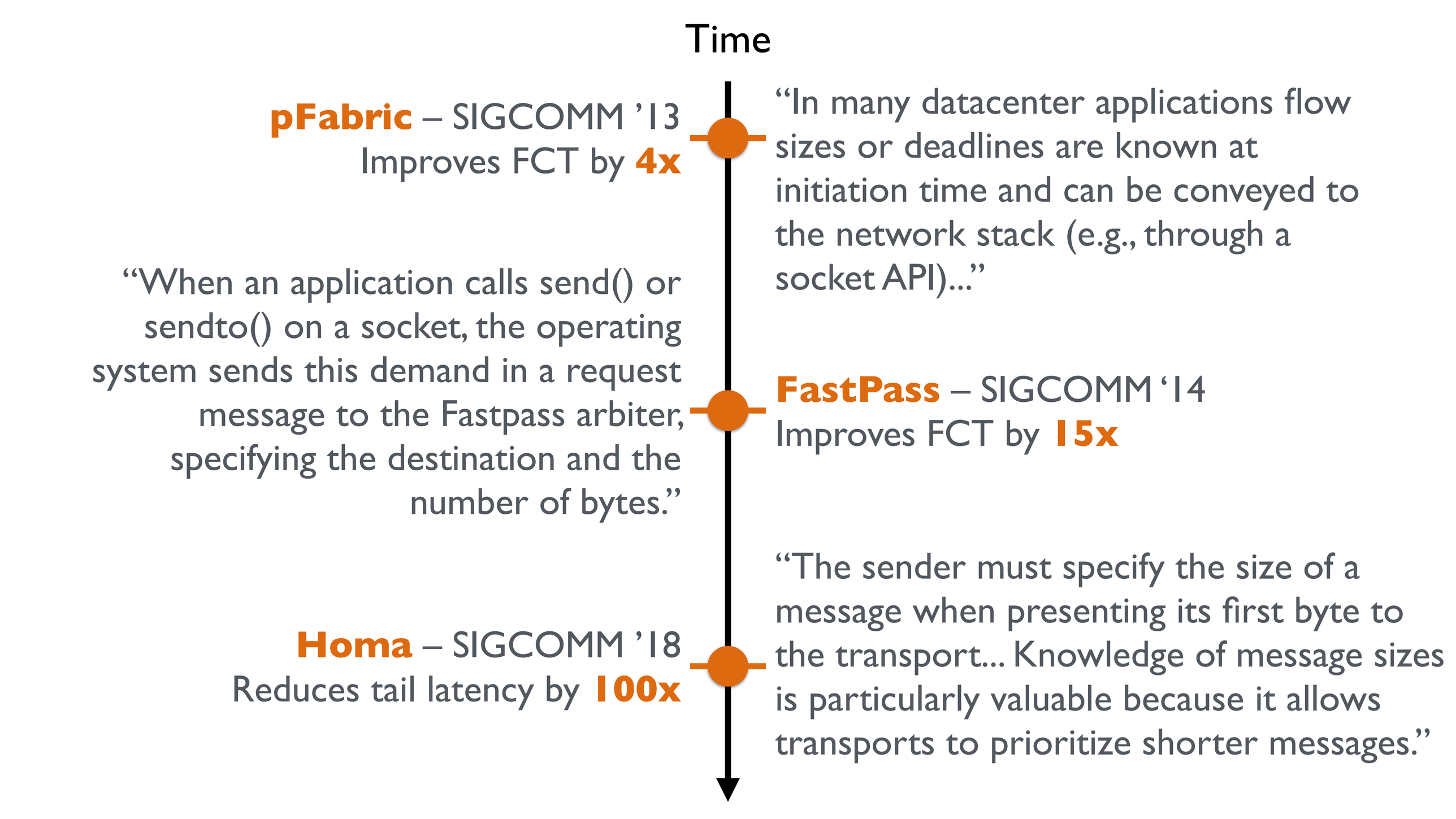
Packet scheduling at a switch queue



Simple, but problematic: do we *know* the shortest flow?

Assumption:

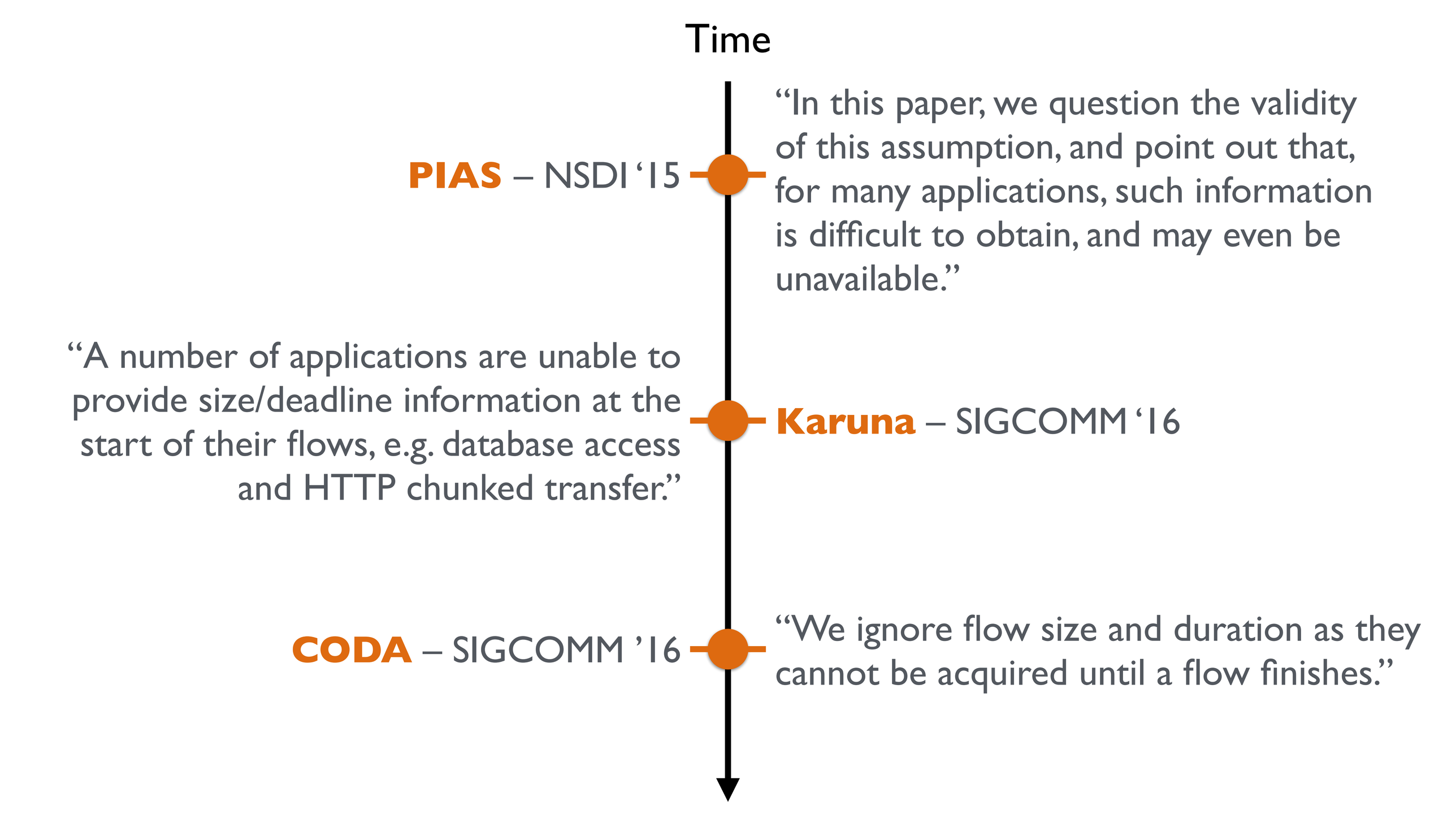
Flow size is know in advance


Time
pFabric – SIGCOMM '13
Improves FCT by **4x**
"In many datacenter applications flow sizes or deadlines are known at initiation time and can be conveyed to the network stack (e.g., through a socket API)..."
"When an application calls send() or sendto() on a socket, the operating system sends this demand in a request message to the Fastpass arbiter, specifying the destination and the number of bytes."
FastPass – SIGCOMM '14
Improves FCT by **15x**
Homa – SIGCOMM '18
Reduces tail latency by **100x**
"The sender must specify the size of a message when presenting its first byte to the transport... Knowledge of message sizes is particularly valuable because it allows transports to prioritize shorter messages."

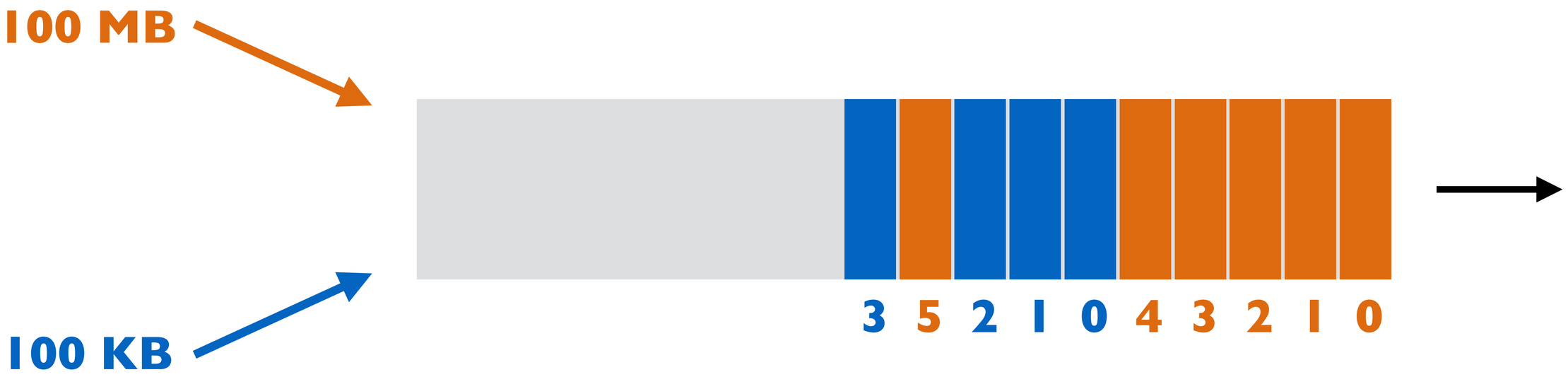
Assumption:

Flow size is know in advance




Time
PIAS – NSDI '15
"In this paper, we question the validity of this assumption, and point out that, for many applications, such information is difficult to obtain, and may even be unavailable."
"A number of applications are unable to provide size/deadline information at the start of their flows, e.g. database access and HTTP chunked transfer."
Karuna – SIGCOMM '16
CODA – SIGCOMM '16
"We ignore flow size and duration as they cannot be acquired until a flow finishes."

Example alternative: Flow aging



Packets sent = packets remaining

Example alternative: Flow aging



Packets sent = packets remaining

Possible alternative: flow aging

Works well for long tail flow size distributions



Possible alternative: flow aging

Works well for long tail flow size distributions



Works only when relative flow size is needed

Doesn't work for equally sized flows



Is advance knowledge of flow sizes a plausible assumption?

Is advance knowledge of flow sizes a plausible assumption?

100% knowledge is likely intractable*

* or at least too expensive

Is advance knowledge of flow sizes a plausible assumption?

100% knowledge is likely intractable*

* or at least too expensive

… but partial knowledge is plausible and useful

Sources of flow size information?

How useful is imprecise / partial knowledge?

Incentives for operators and users?

Sources of flow size information?

How useful is imprecise / partial knowledge?

Incentives for operators and users?

Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces

Many apps know flow sizes

Doable in private DCs



Flow size estimation: design space

Exact sizes given by application

Many apps know flow sizes

Doable in private DCs



Some apps don’t

Change a lot of applications

Change network API



TCP buffer occupancy

Monitoring system calls

Learning from past traces

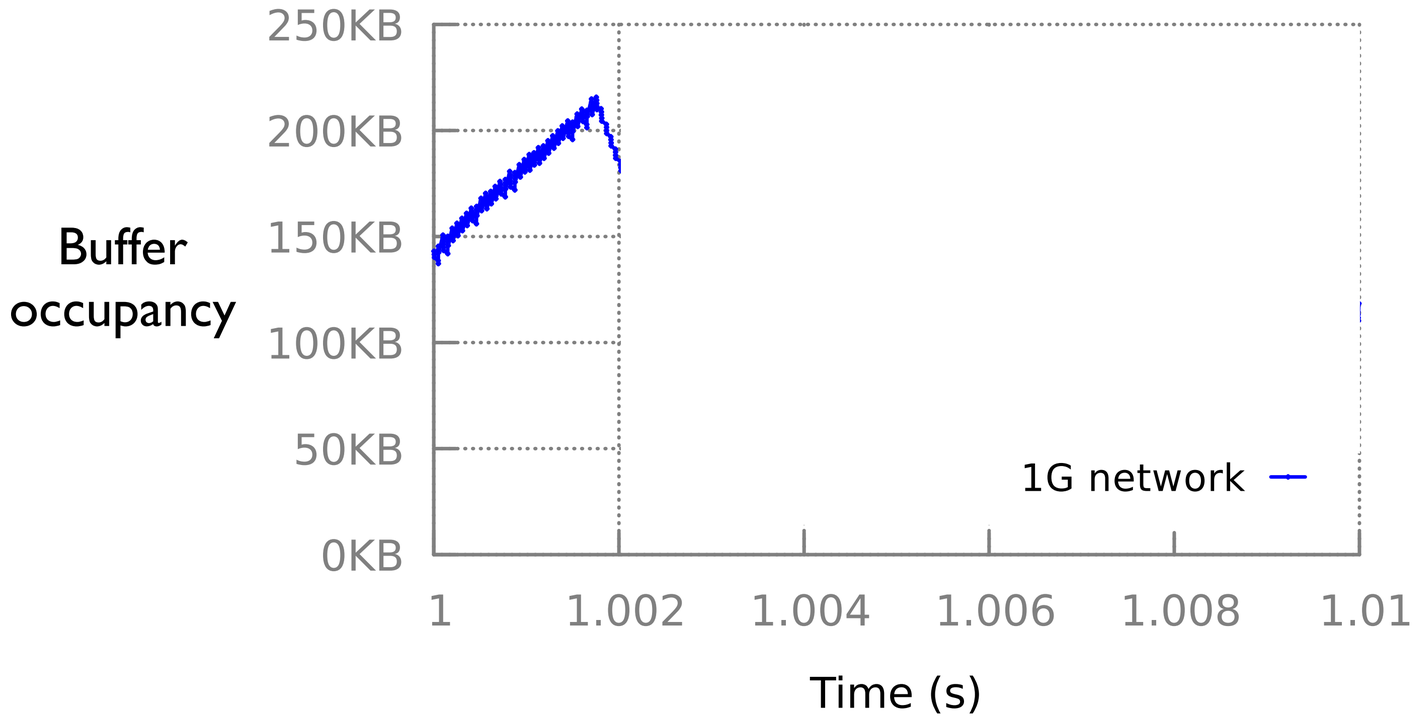
Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



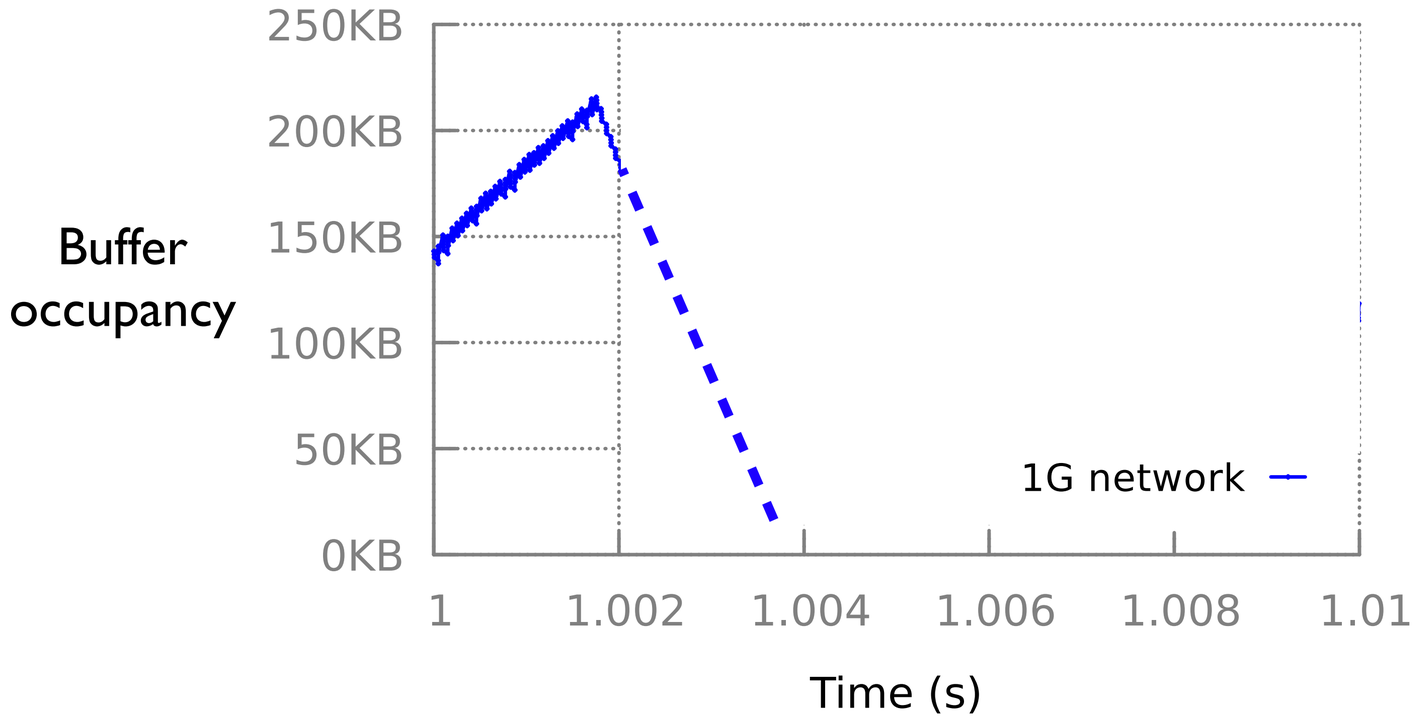
Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



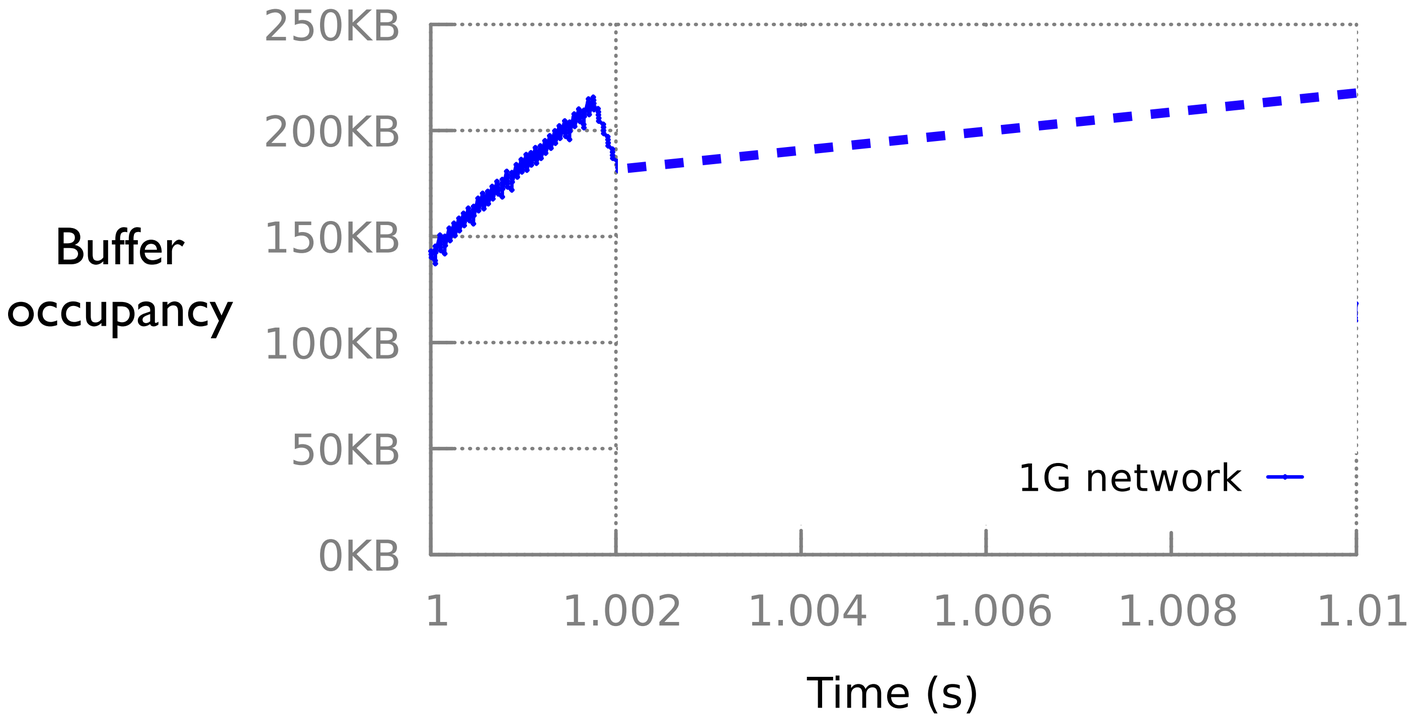
Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



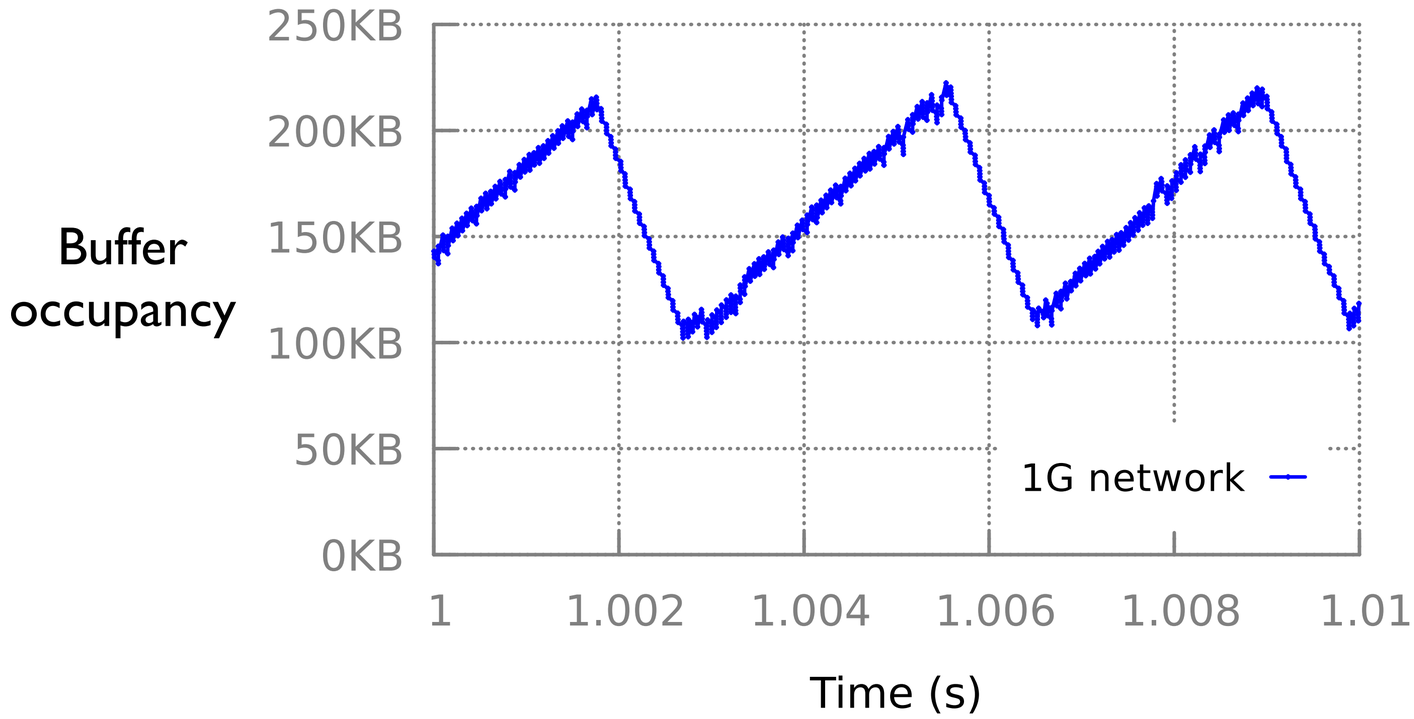
Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



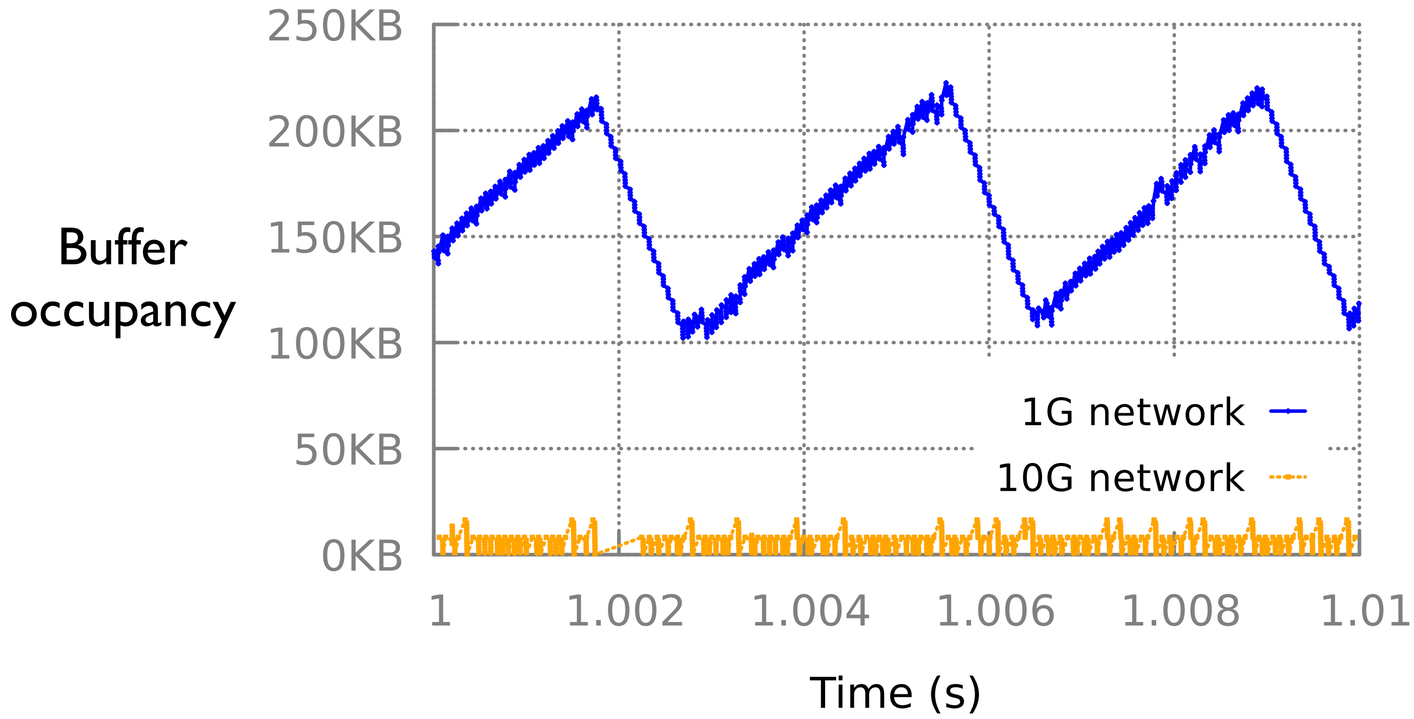
Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces

```
while(data in buffer):
  y = write(
   socket_desc,
   buffer+x,
   100KB
  )
```

Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces

```
flow_size = f(
 disk I/O,
 memory I/O,
 past network traffic,
 computation
)
```

Learning flow size

Workload		Trace		Model		Performance
Web server	+	Disk I/O	+	GBDT	=	**0.96**
TensorFlow		Memory I/O		FFNN		**0.97**
PageRank		CPU cycles		LSTM		**0.83**
KMeans		Past traffic				**0.88**
SGD						**0.79**

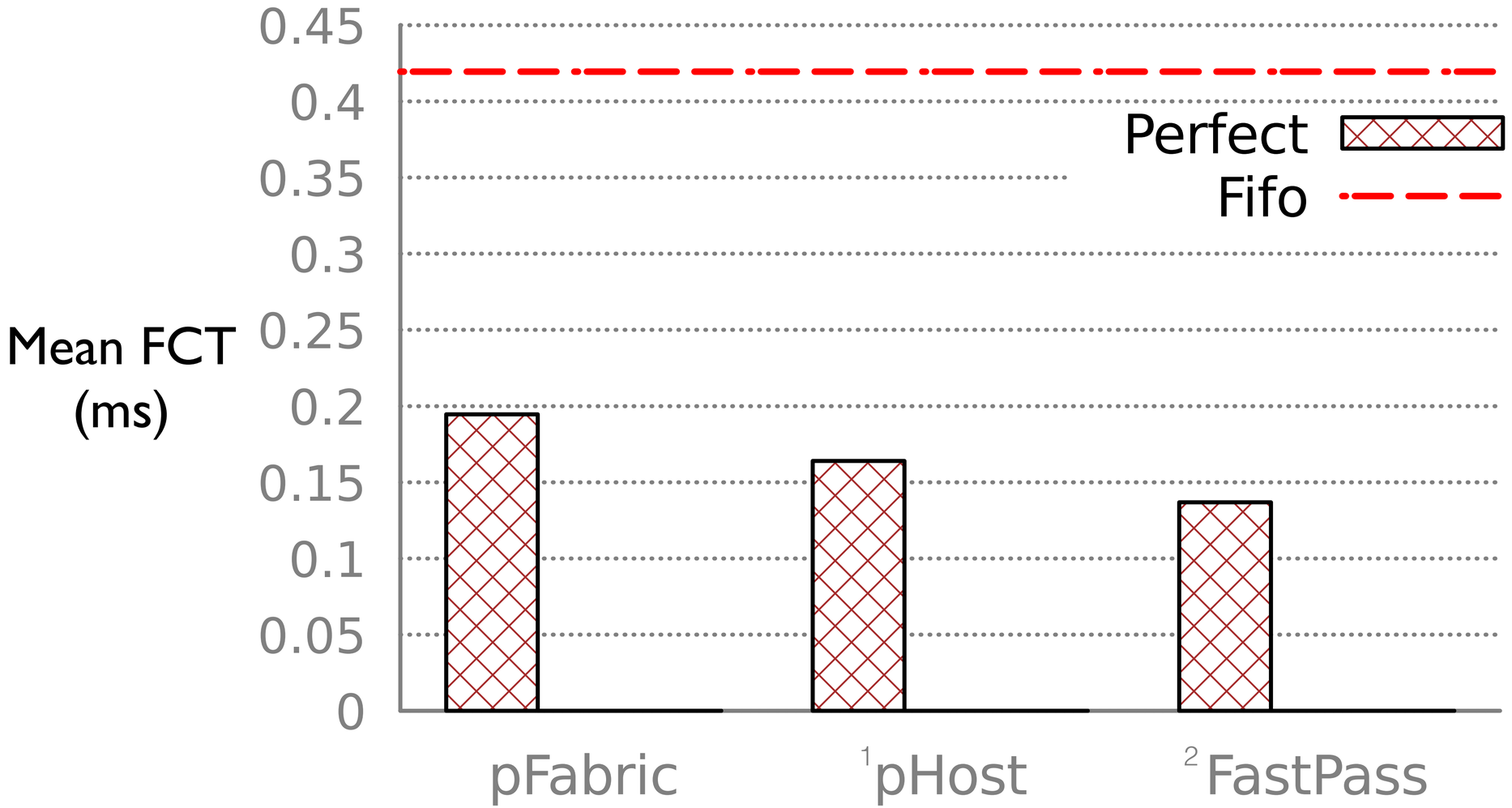
*Values are in R^2
1 – perfect prediction
0 – mean value prediction

Sources of flow size information?

How useful is imprecise / partial knowledge?

Incentives for operators and users?

Flow scheduling using imprecise knowledge

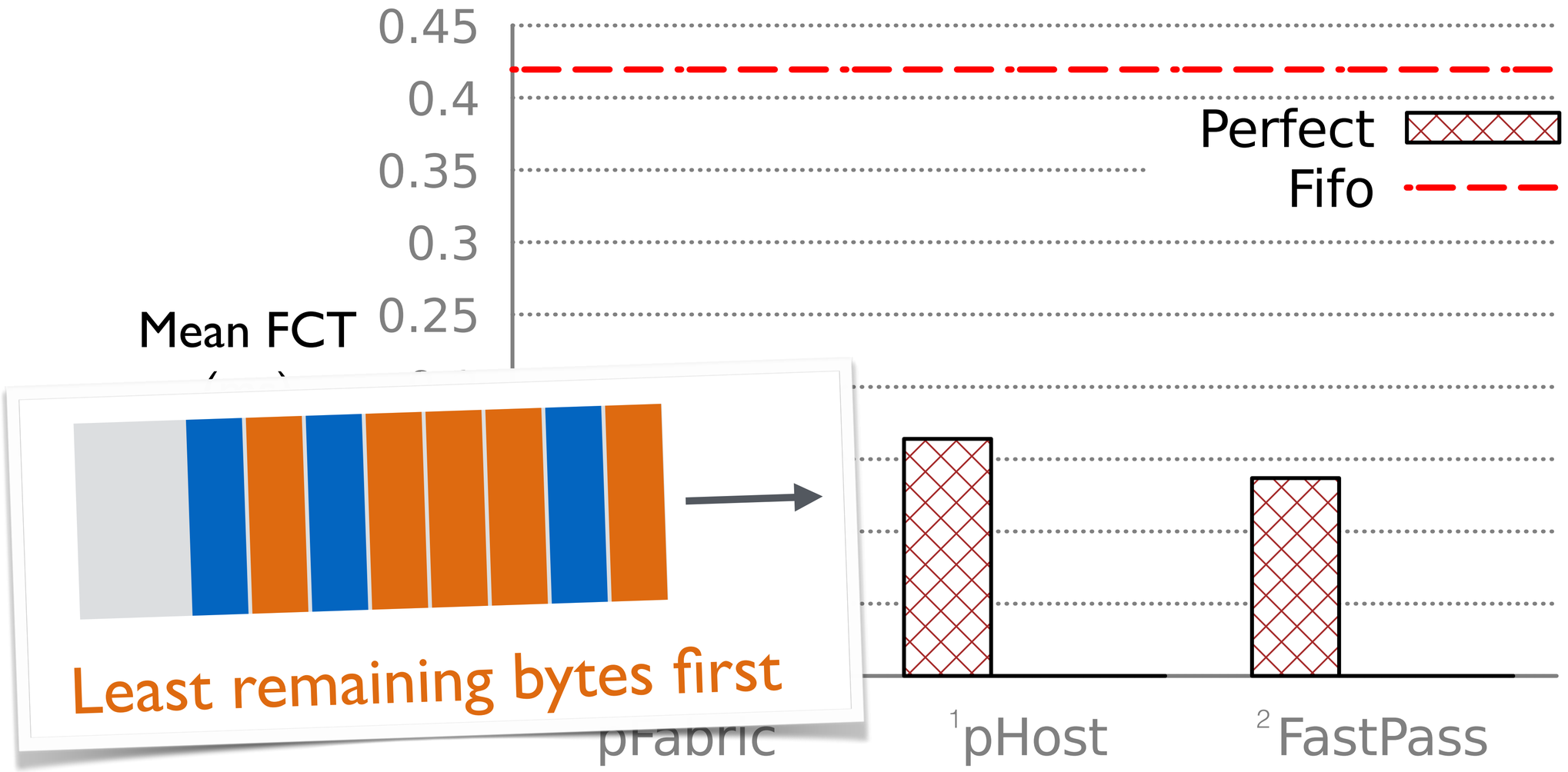


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Flow scheduling using imprecise knowledge

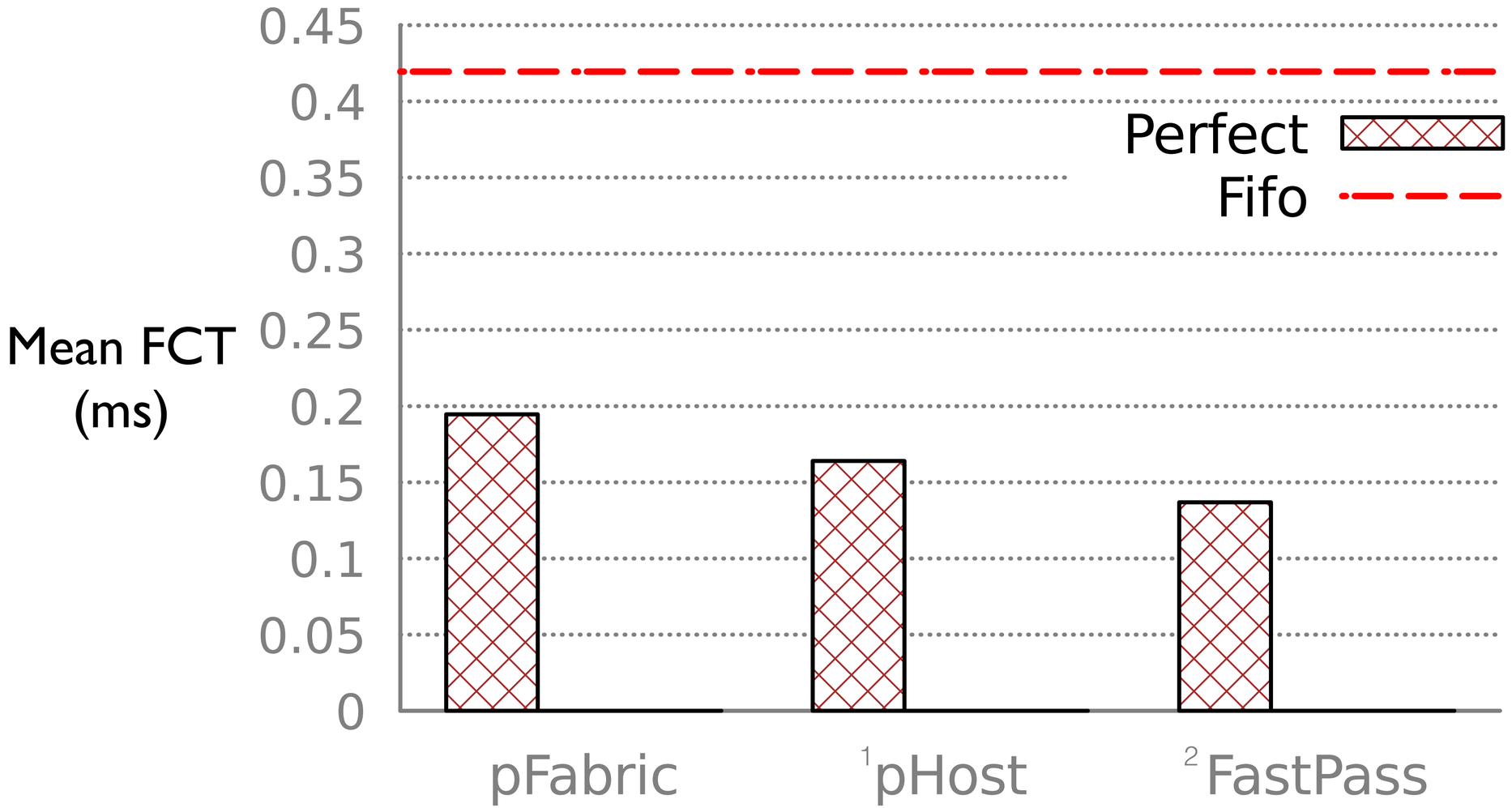


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Flow scheduling using imprecise knowledge

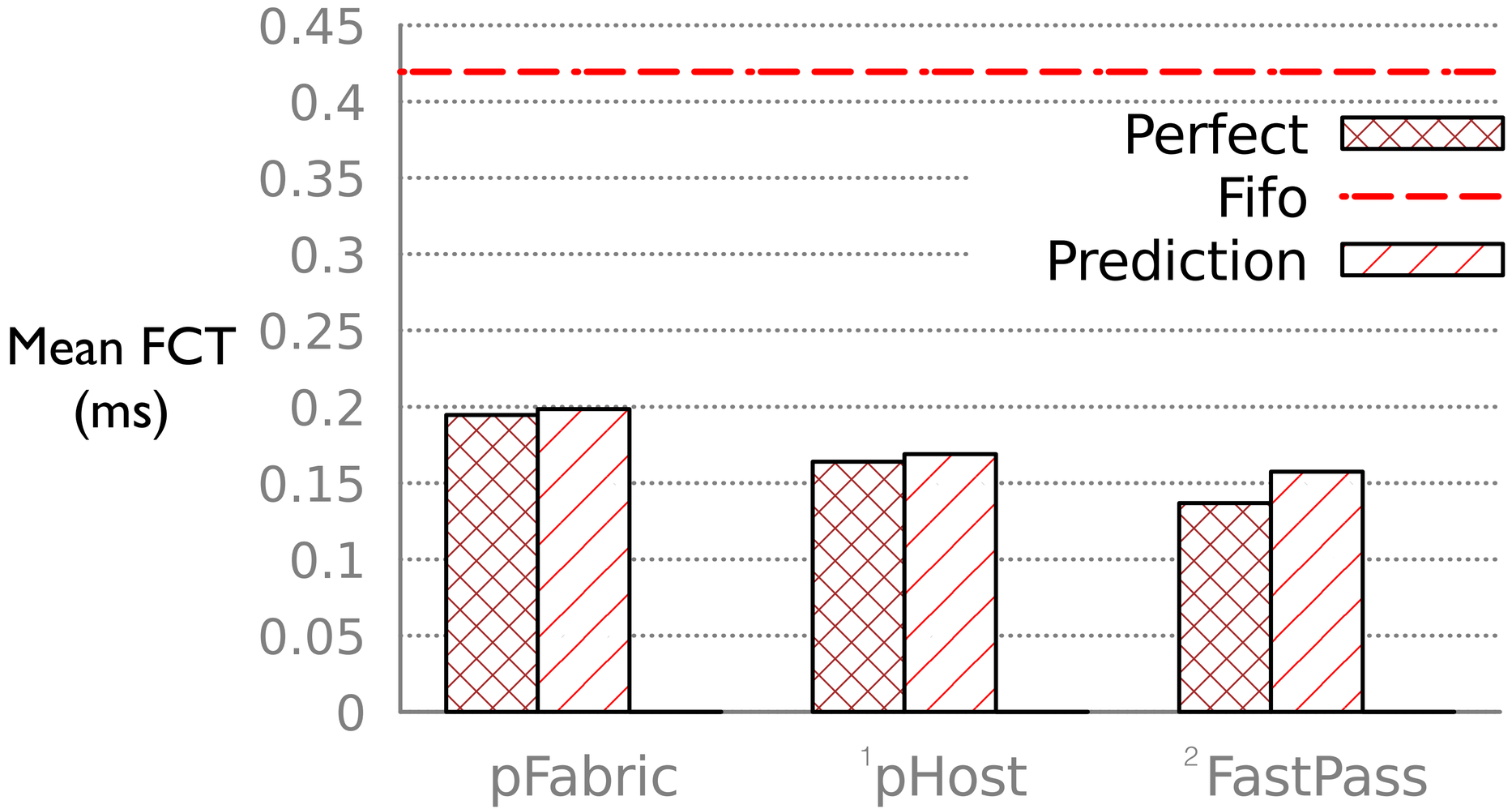


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Flow scheduling using imprecise knowledge

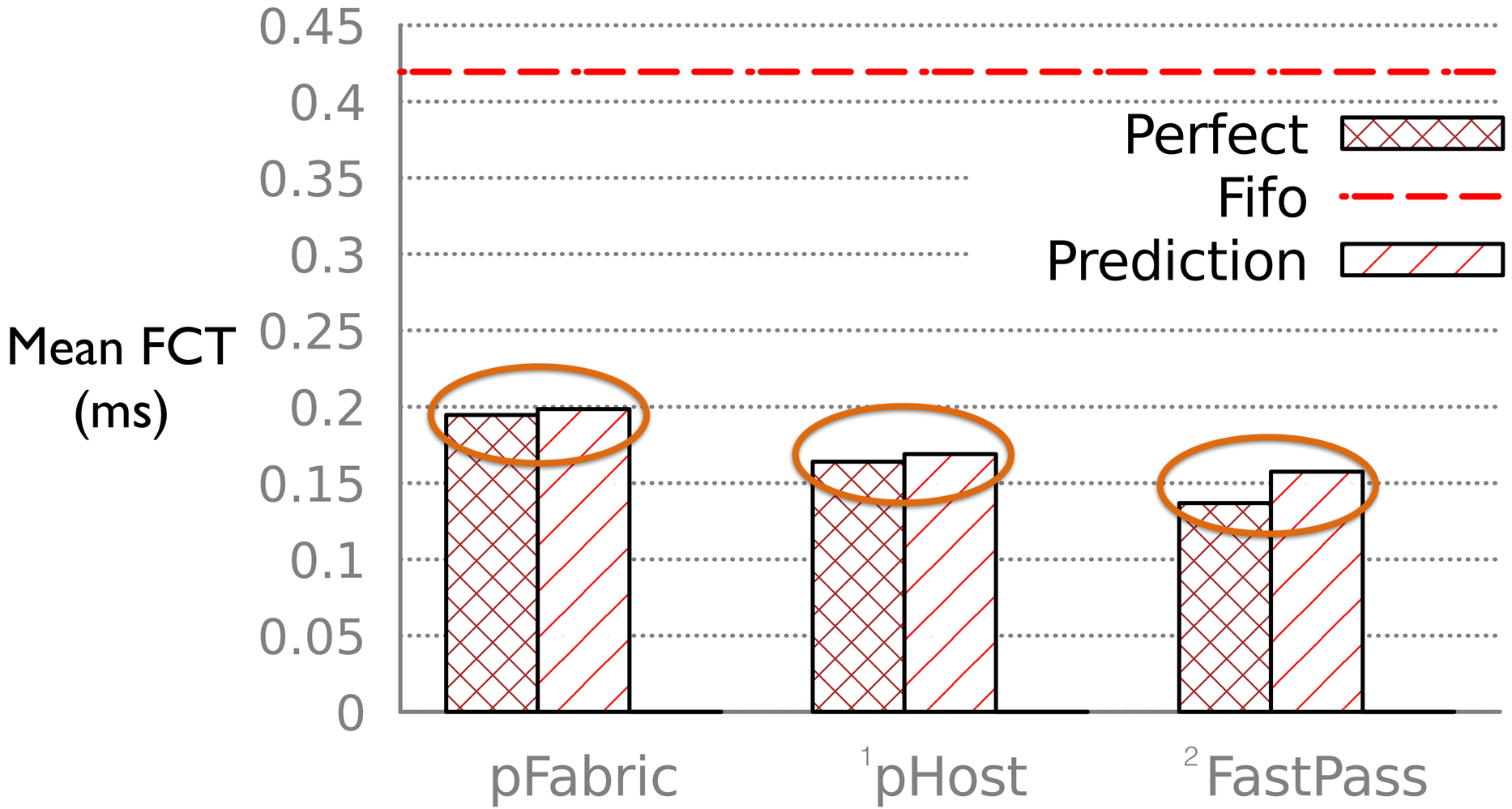


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Flow scheduling using imprecise knowledge

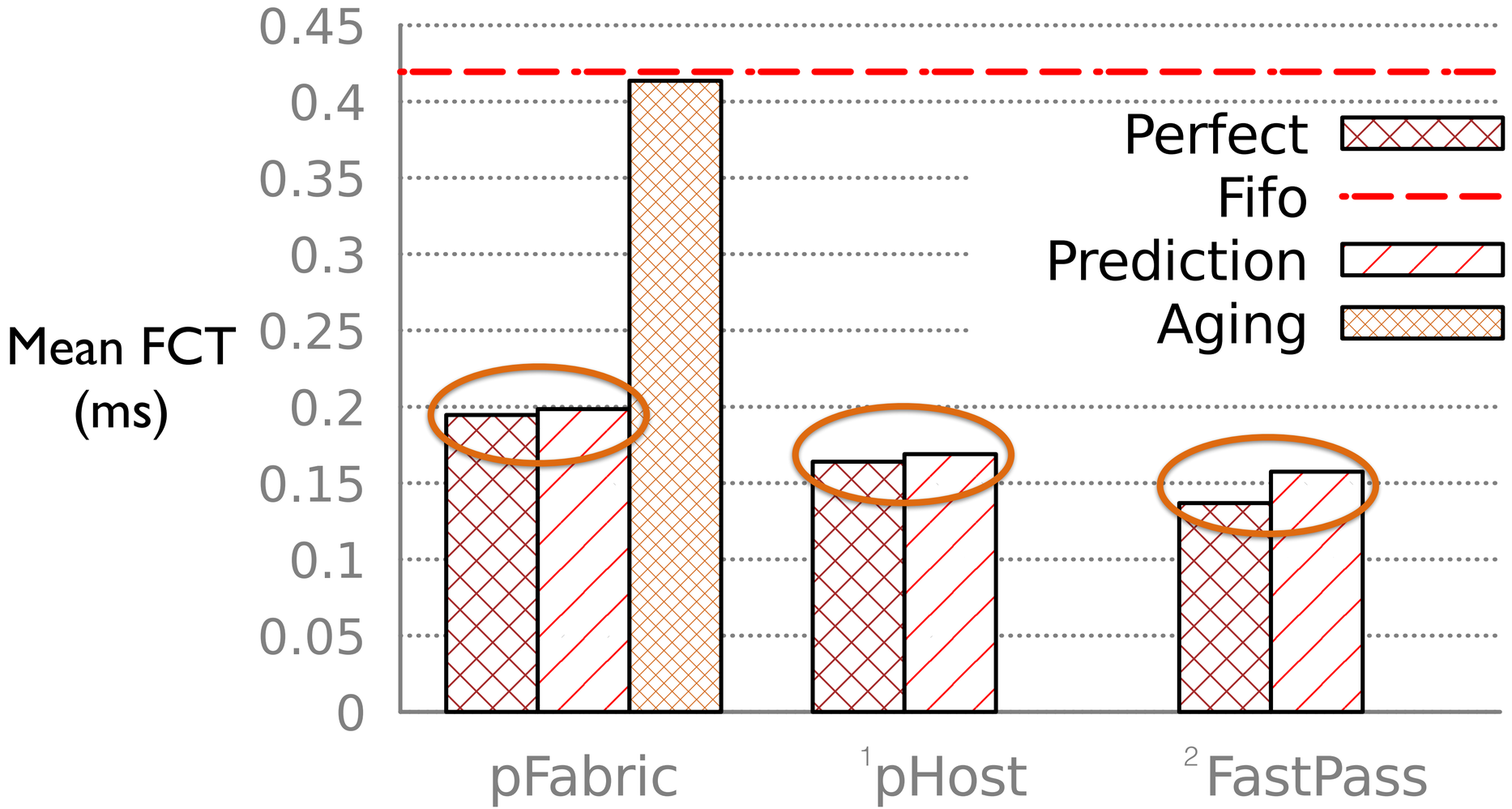


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Flow scheduling using imprecise knowledge

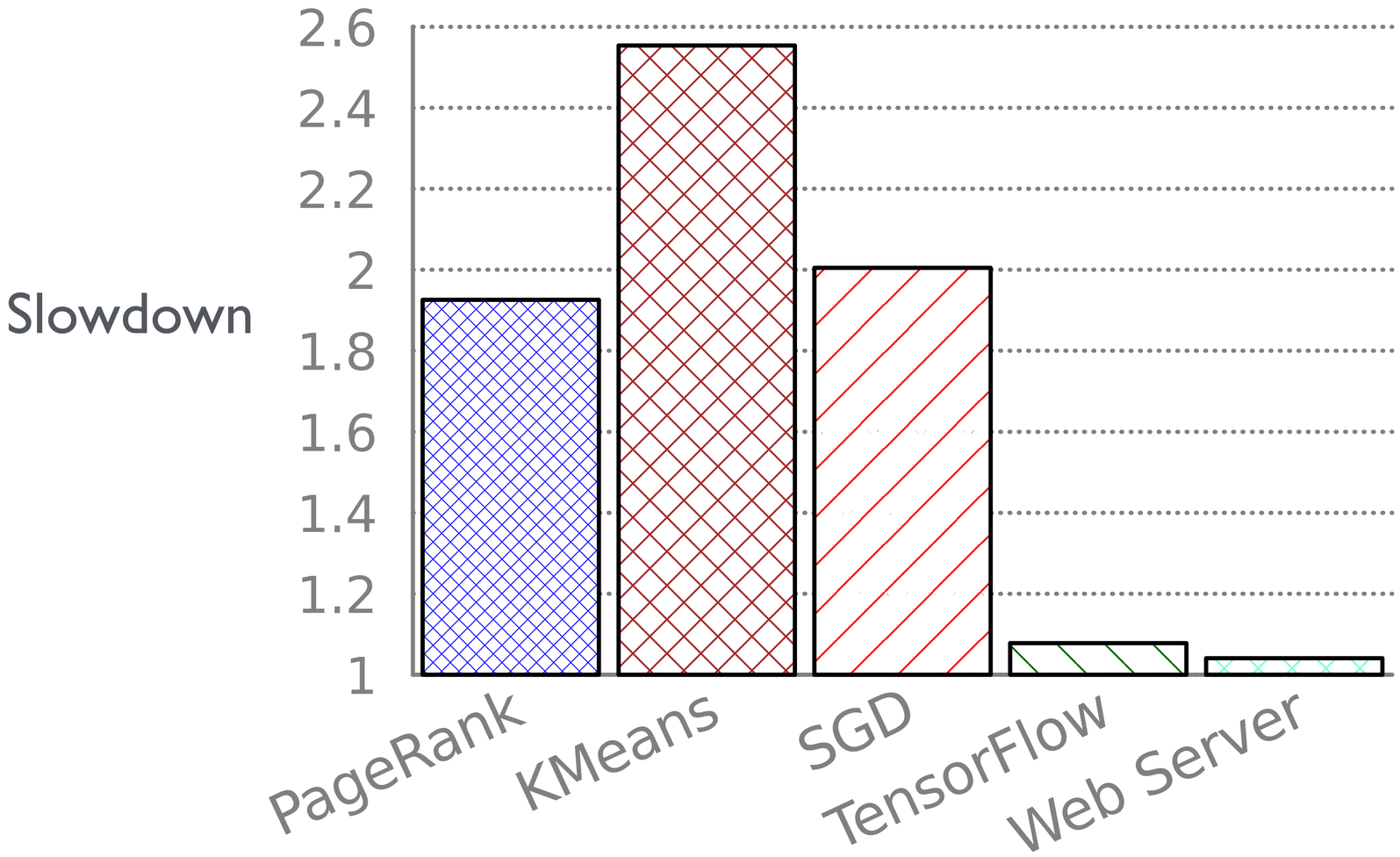


[TensorFlow workload]

[1. pHost: Distributed Near-Optimal Datacenter Transport Over Commodity Network Fabric; CoNEXT '15]

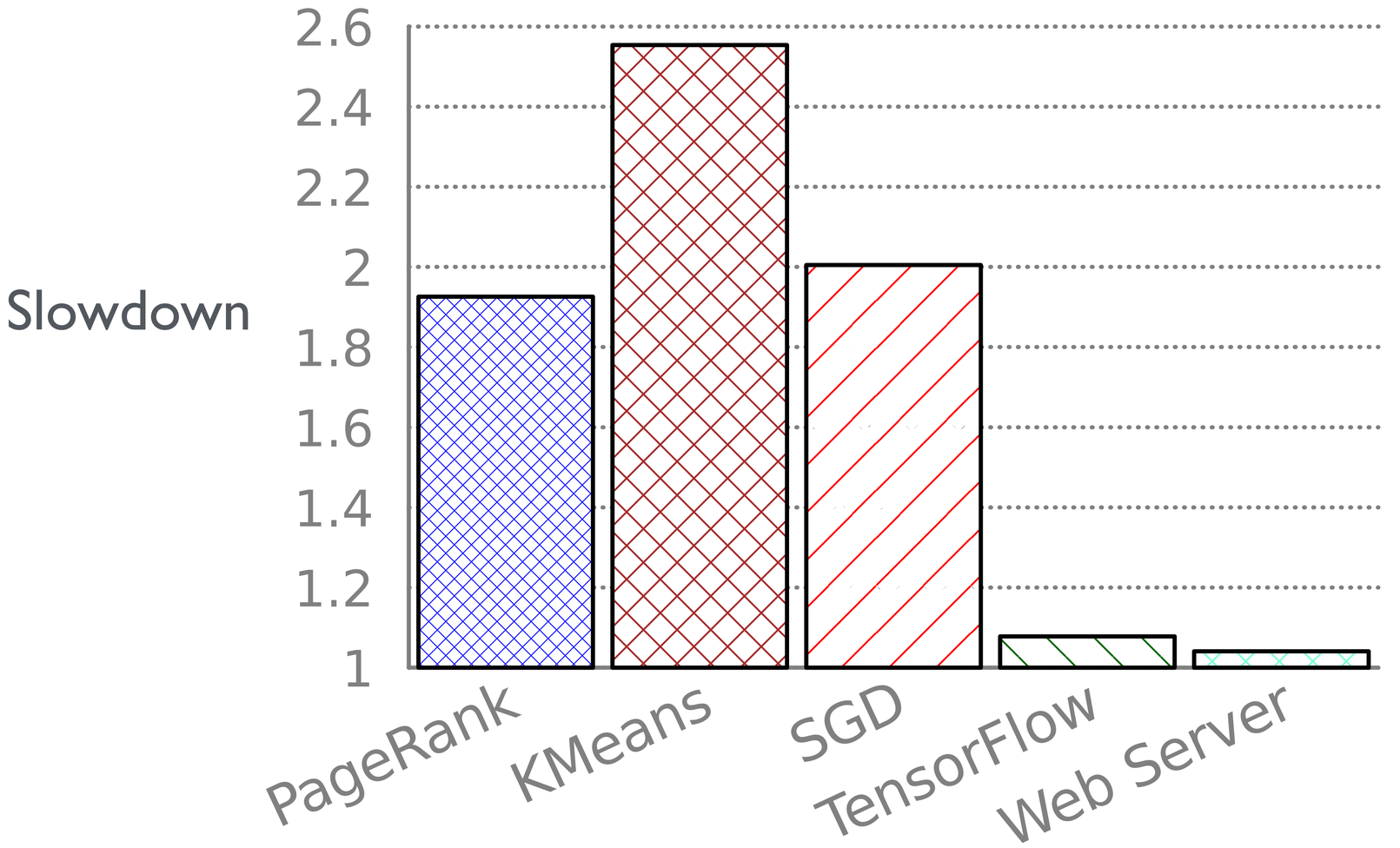
[2. Fastpass: A Centralized "Zero-Queue" Datacenter Network; SIGCOMM '14]

Coflow scheduling using imprecise knowledge (Sincronia*)



[*Sincronia: Near-Optimal Network Design for Coflows; SIGCOMM '18]

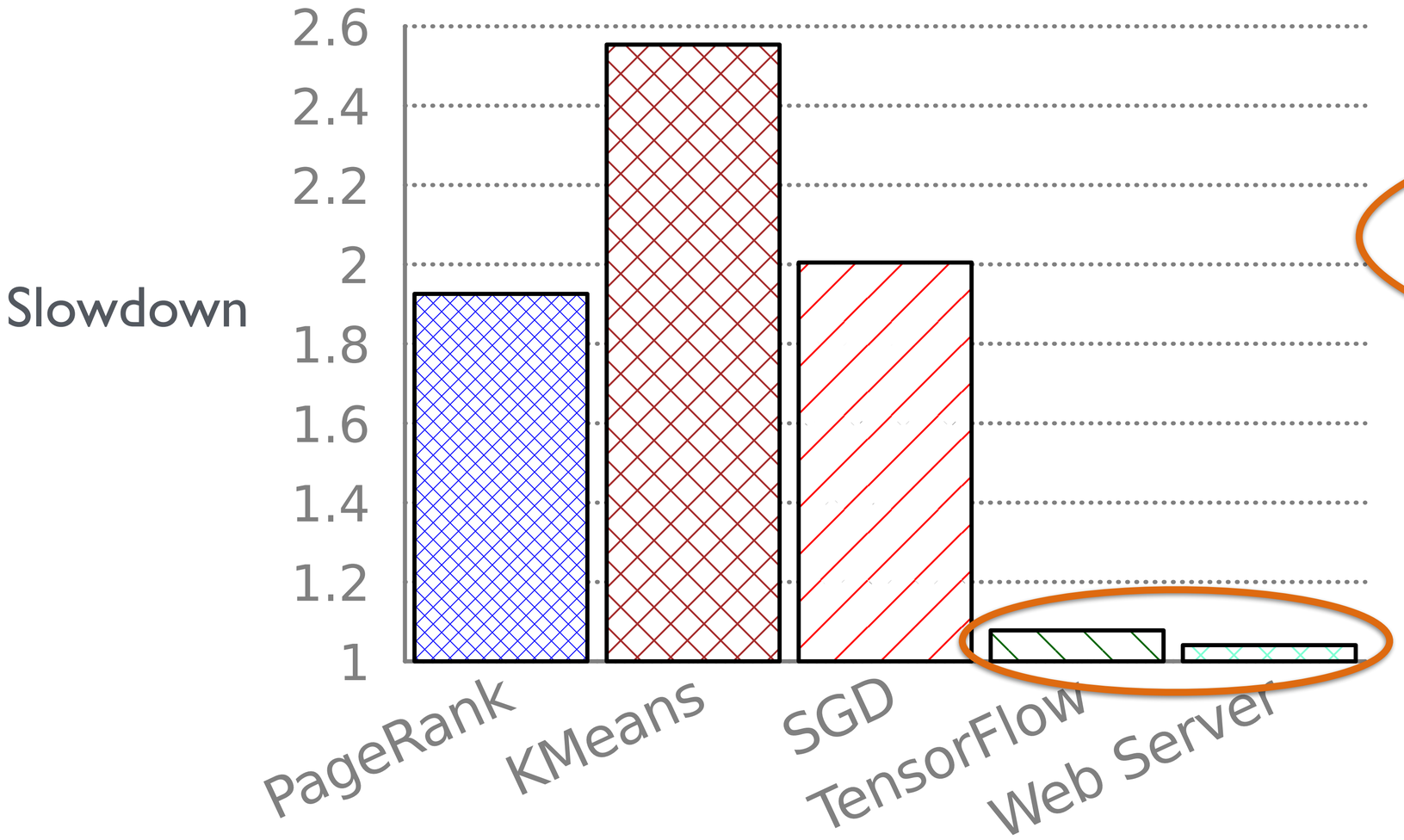
Coflow scheduling using imprecise knowledge (Sincronia*)



Workload	R^2
Web server	0.96
TensorFlow	0.97
PageRank	0.83
KMeans	0.88
SGD	0.79

[*Sincronia: Near-Optimal Network Design for Coflows; SIGCOMM '18]

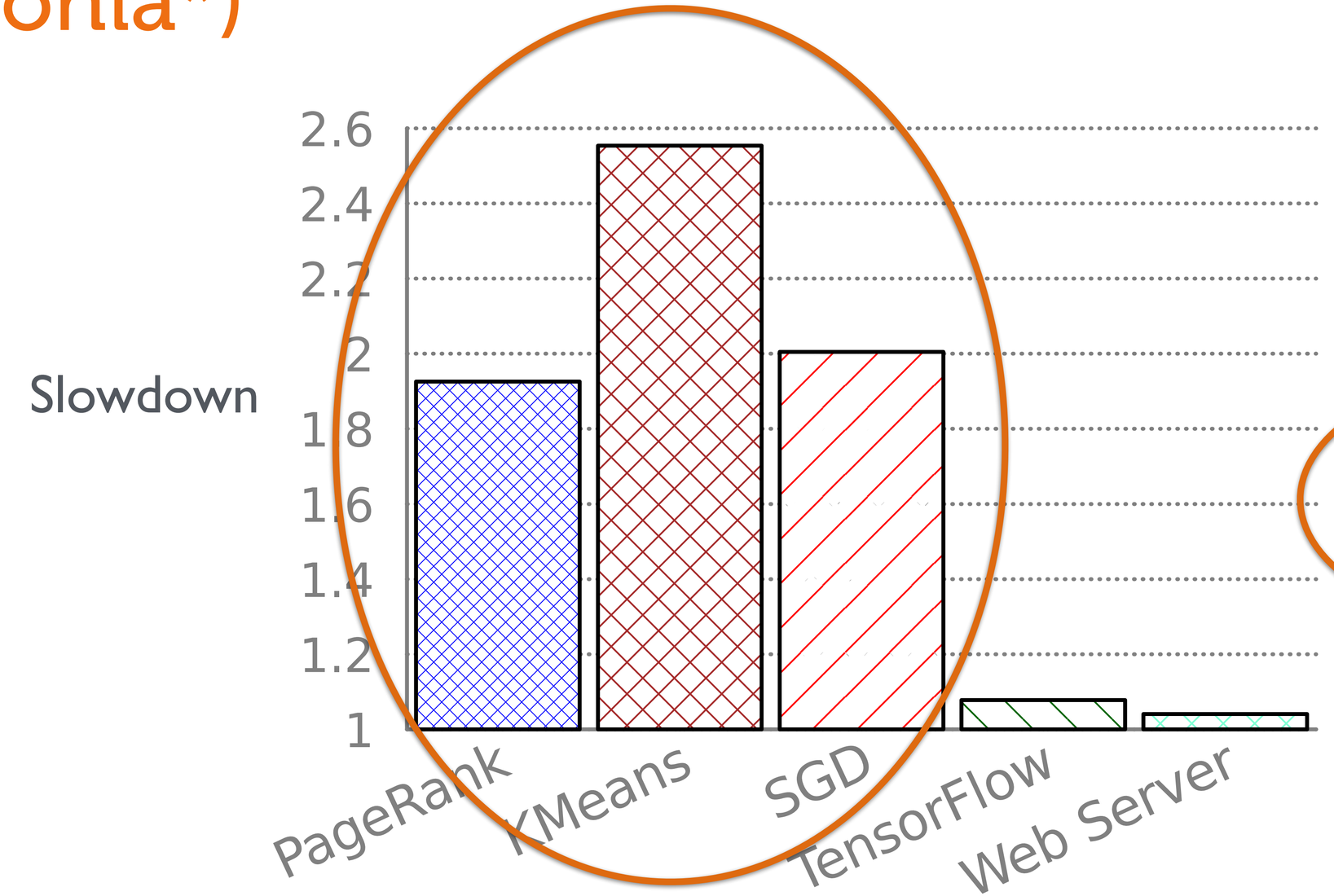
Coflow scheduling using imprecise knowledge (Sincronia*)



Workload	R^2
Web server	0.96
TensorFlow	0.97
PageRank	0.83
KMeans	0.88
SGD	0.79

[*Sincronia: Near-Optimal Network Design for Coflows; SIGCOMM '18]

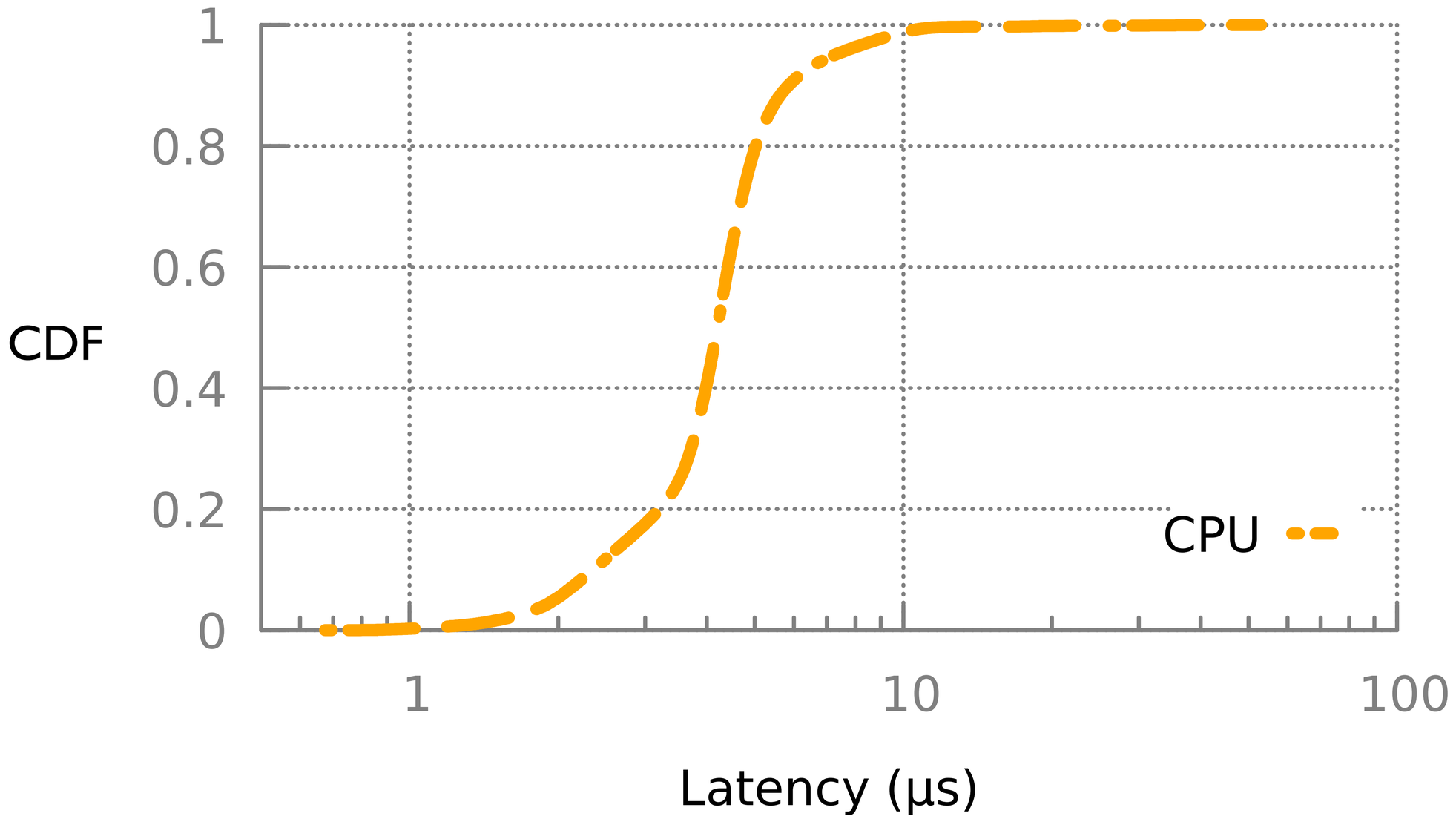
Coflow scheduling using imprecise knowledge (Sincronia*)



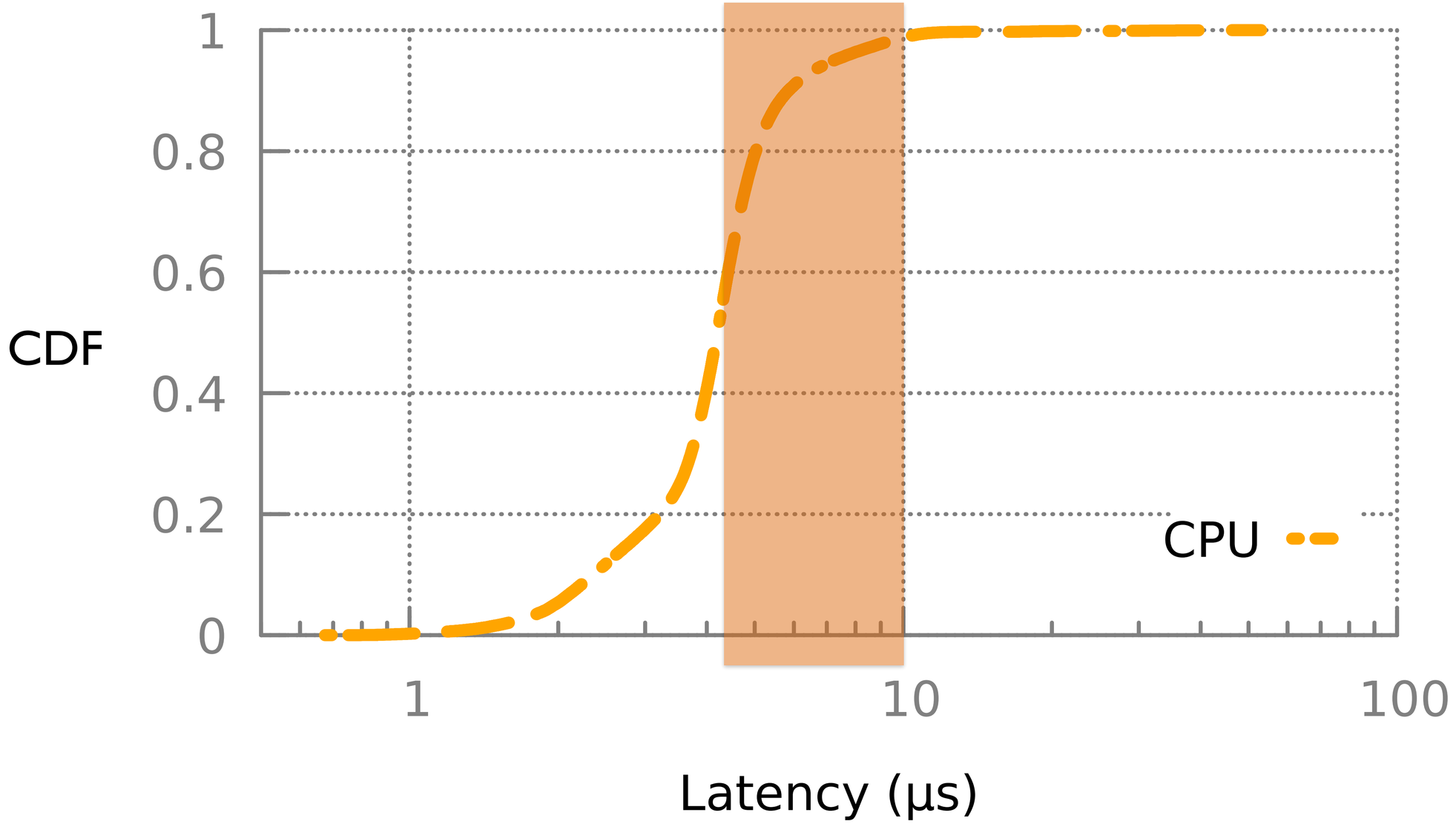
Workload	R^2
Web server	0.96
TensorFlow	0.97
PageRank	0.83
KMeans	0.88
SGD	0.79

[*Sincronia: Near-Optimal Network Design for Coflows; SIGCOMM '18]

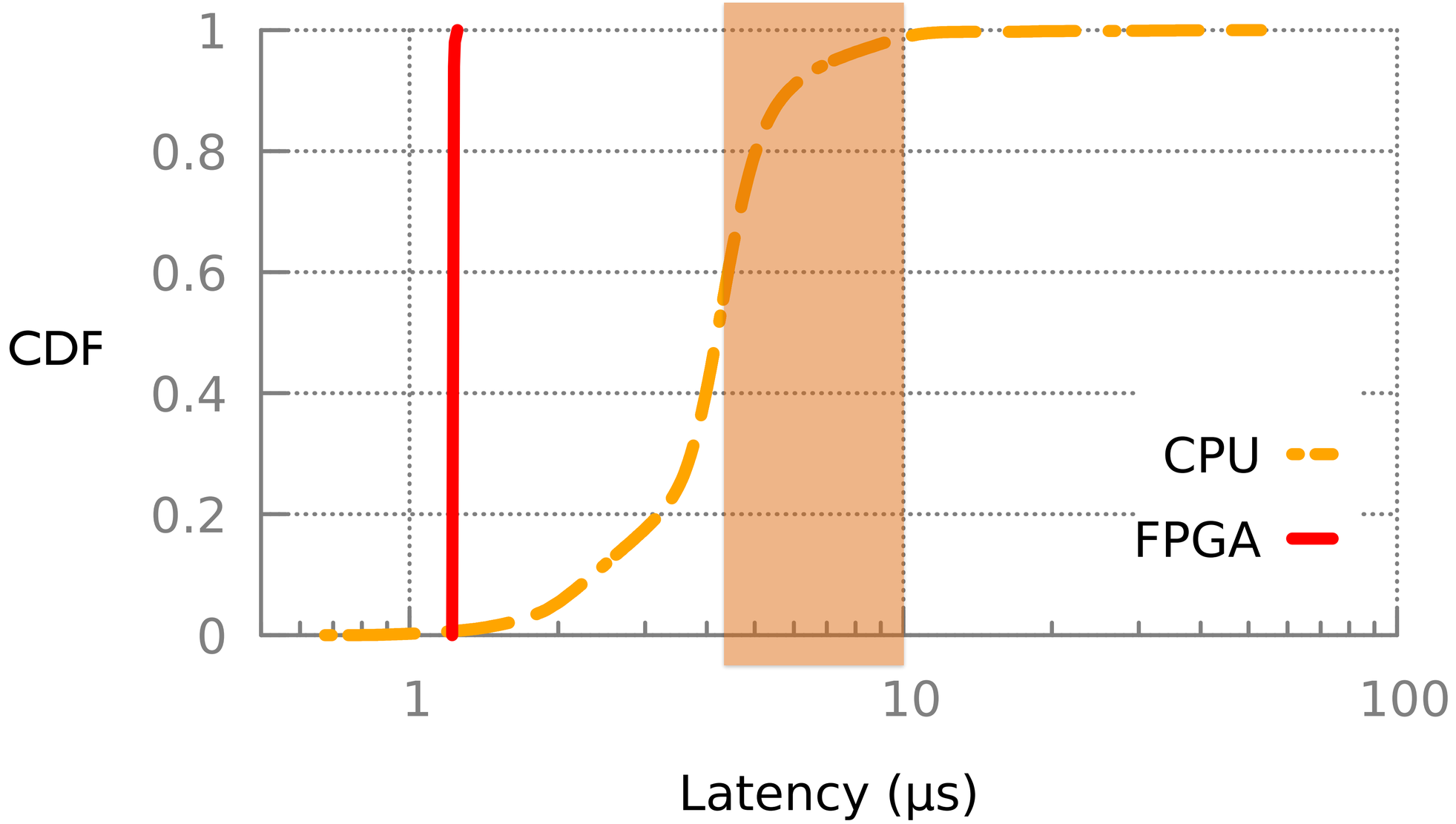
Fast enough, deployable learning?



Fast enough, deployable learning?



Fast enough, deployable learning?



Flow size estimation: design space

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces

Works only for repetitive workloads

Must identify the application

Substantial implementation effort.

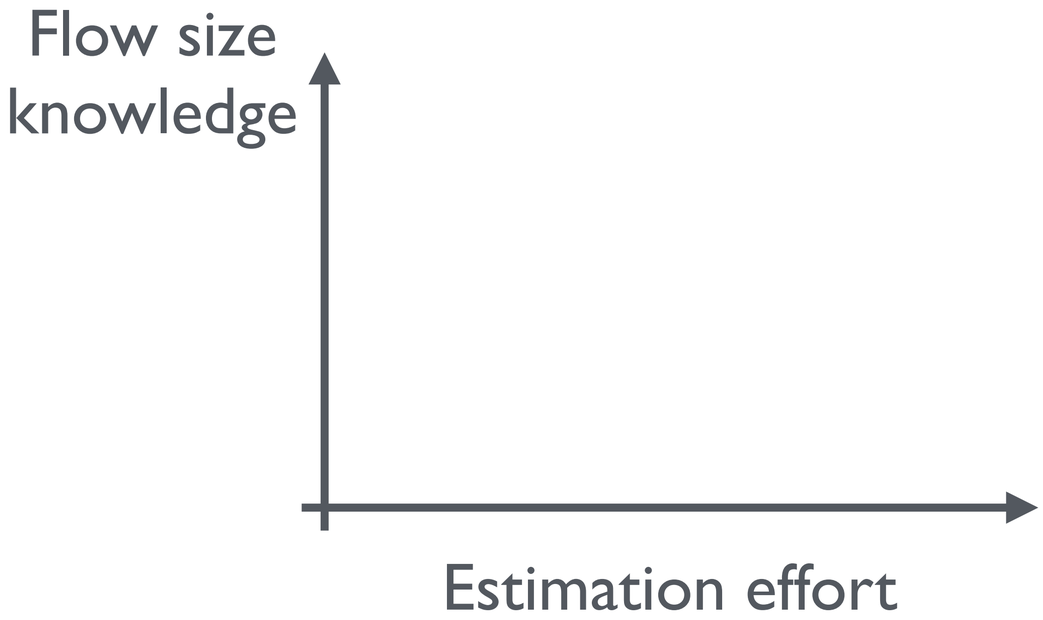
Every technique provides some knowledge

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



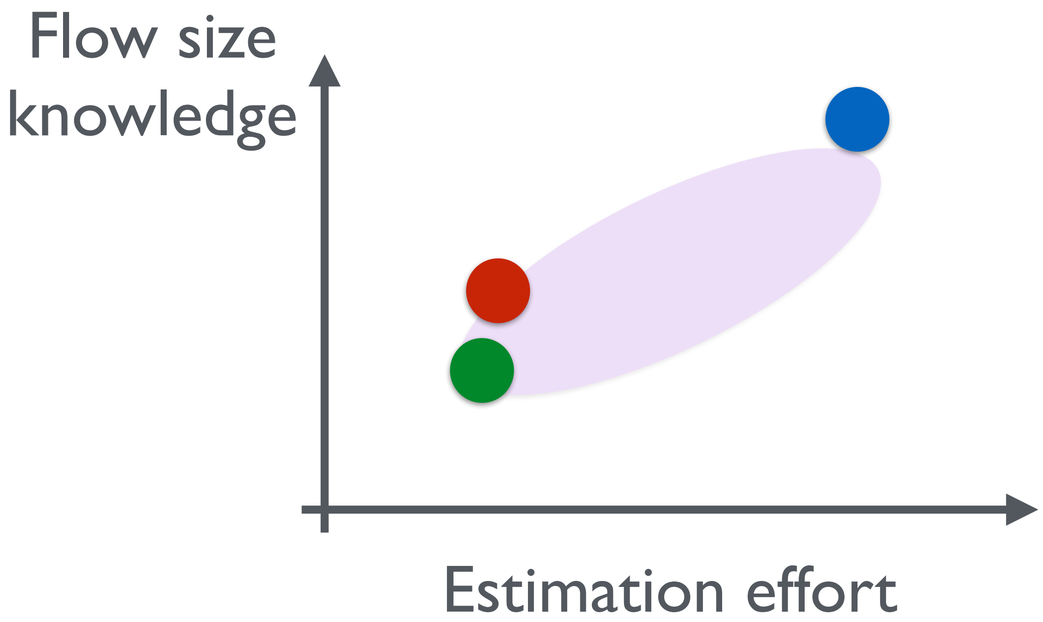
Every technique provides some knowledge

Exact sizes given by application

TCP buffer occupancy

Monitoring system calls

Learning from past traces



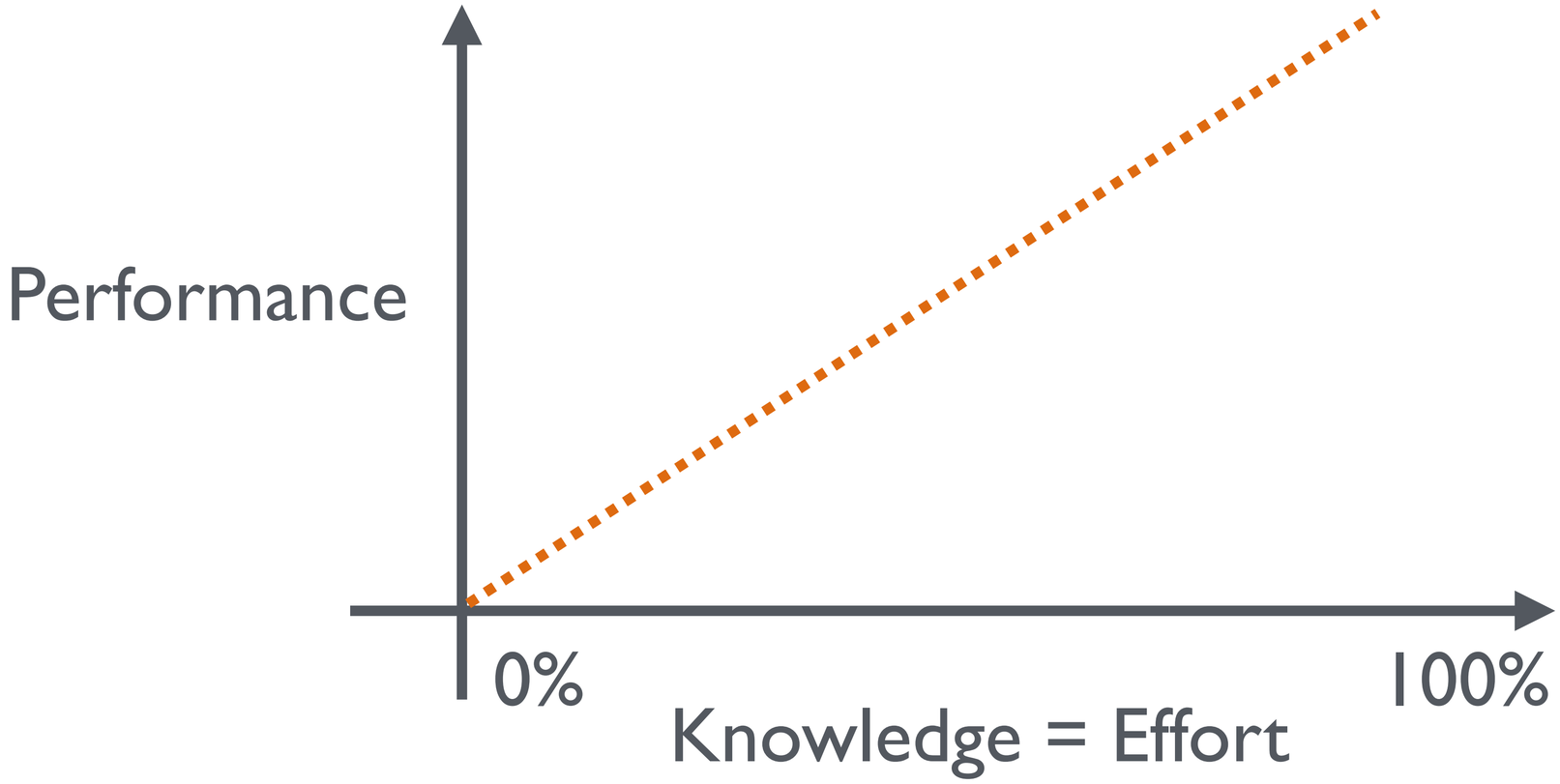
Sources of flow size information?

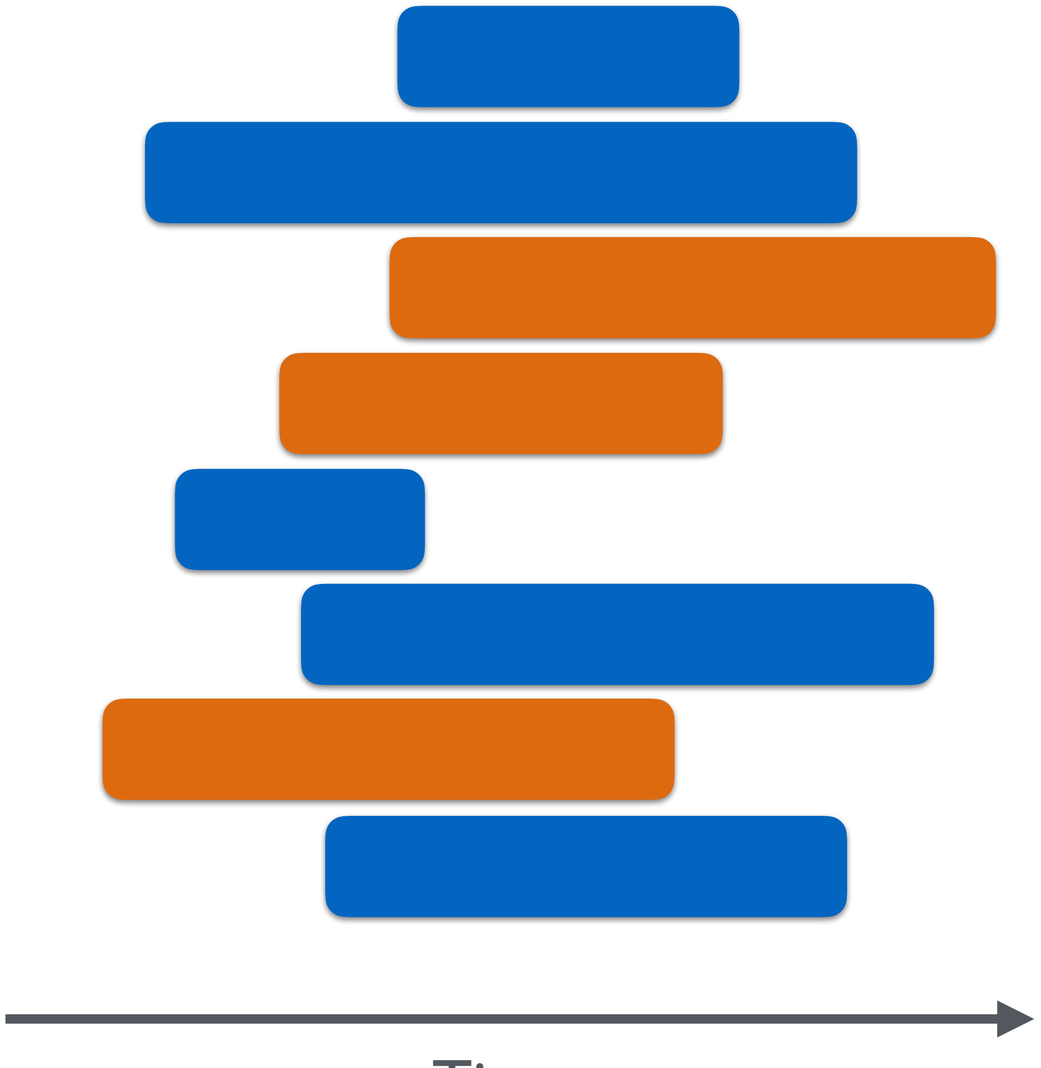
How useful is imprecise / partial knowledge?

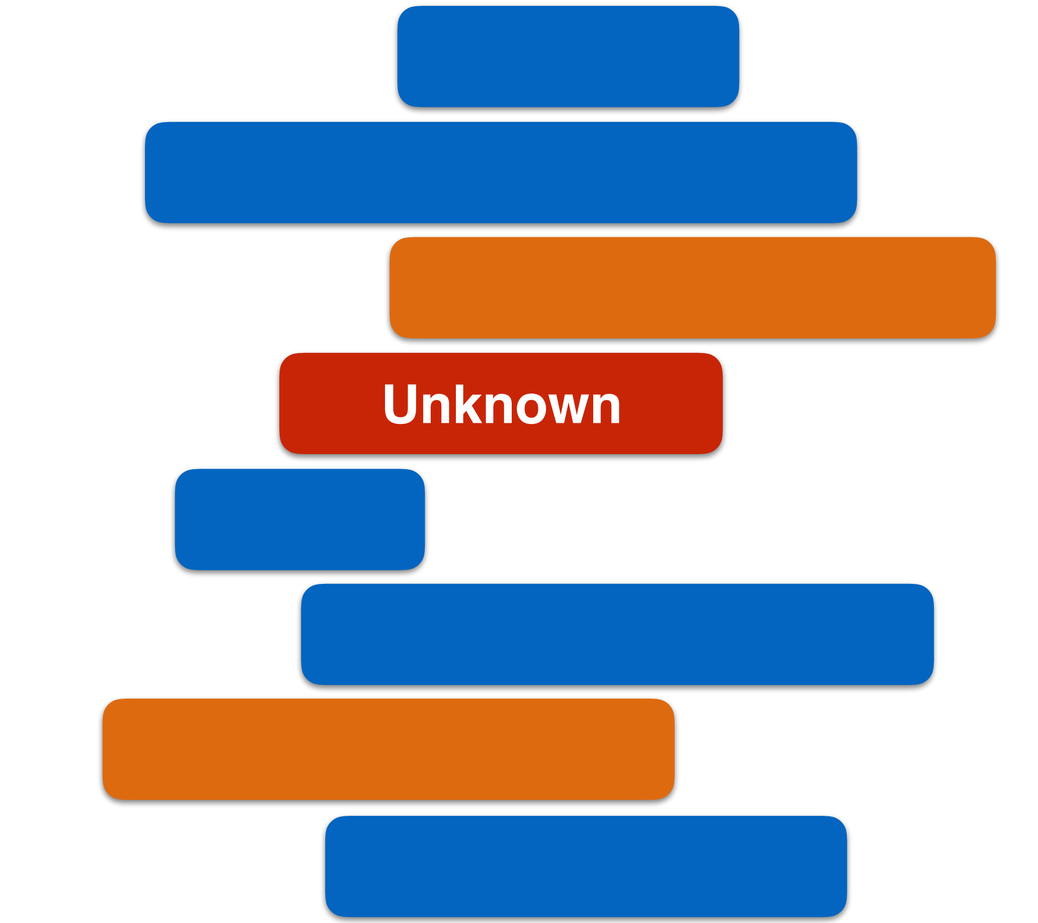
Incentives for operators and users?

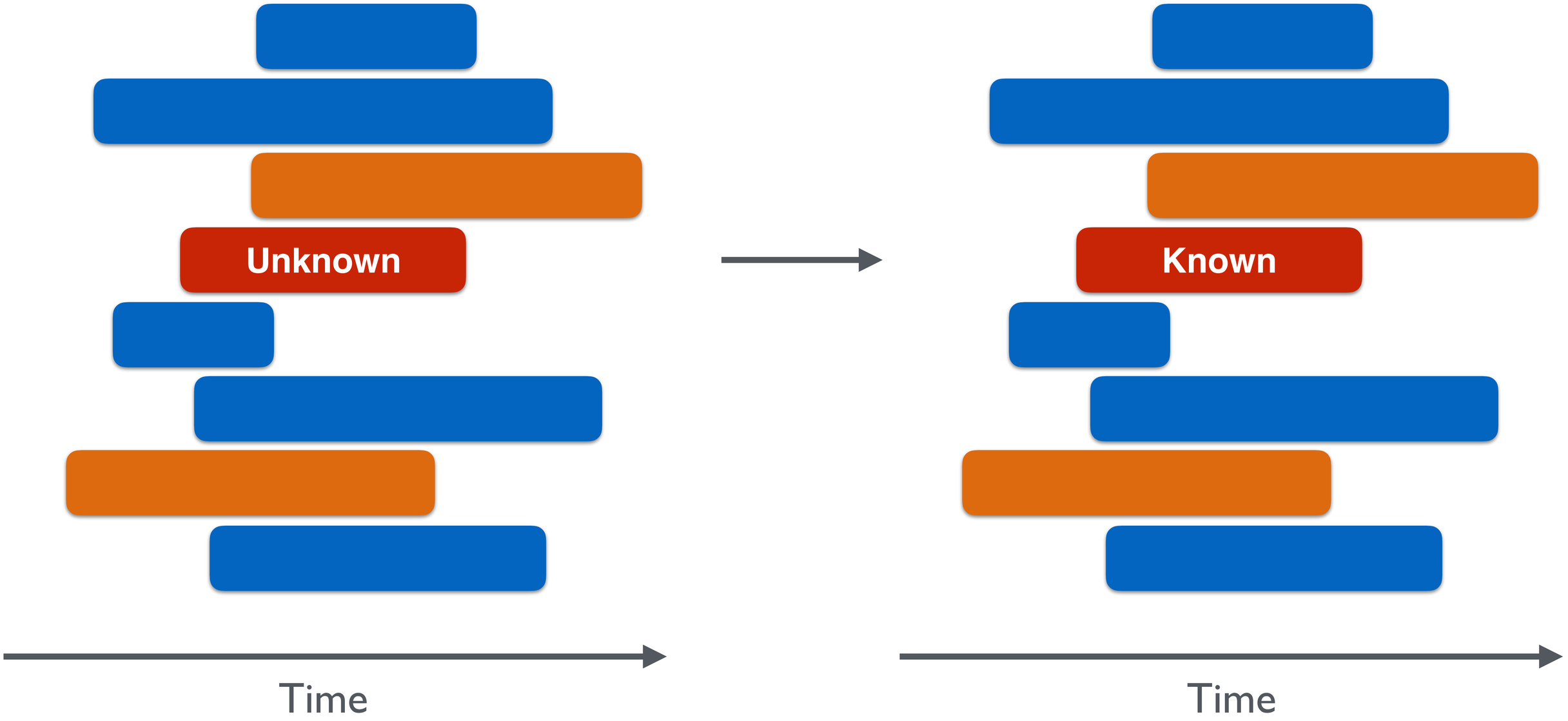
What we want:

More knowledge ⇒ better performance

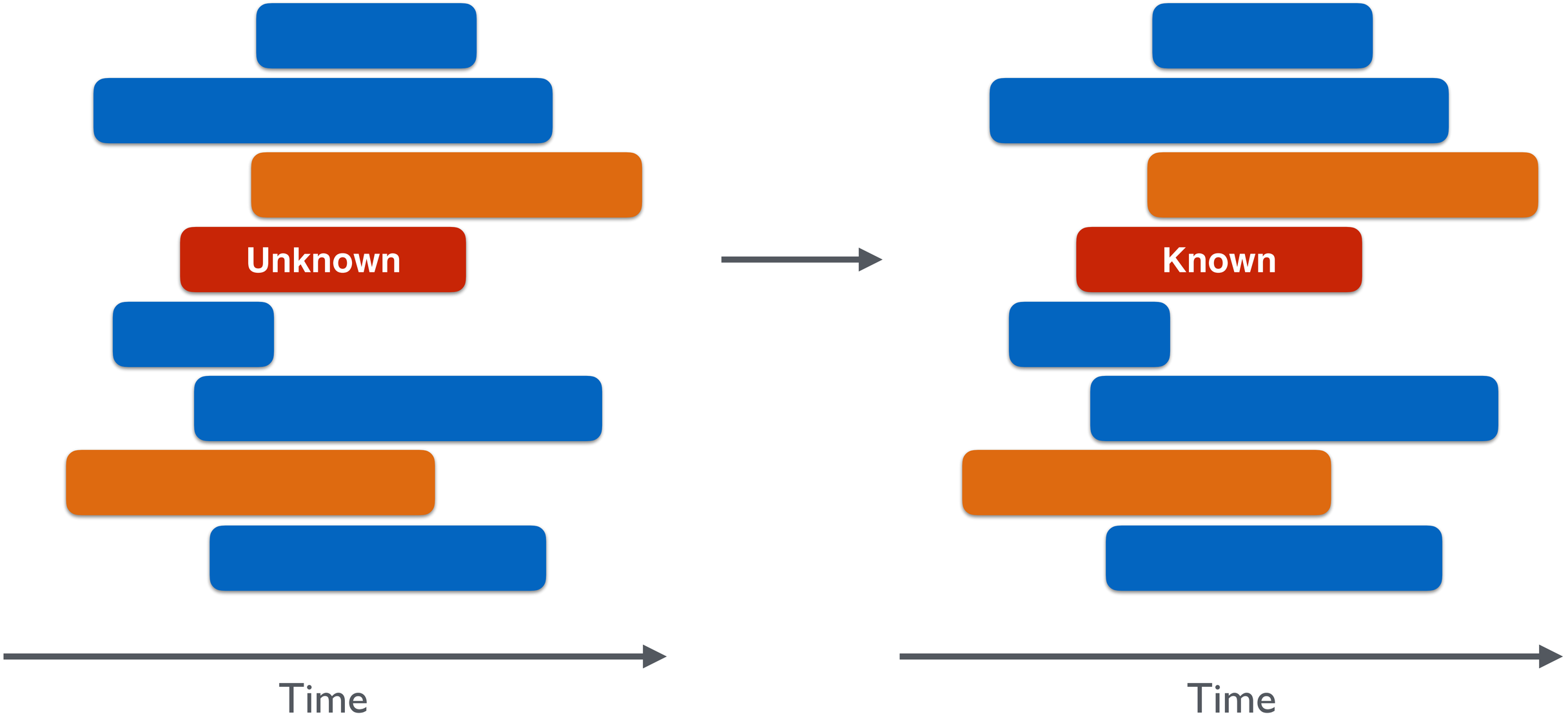



Time

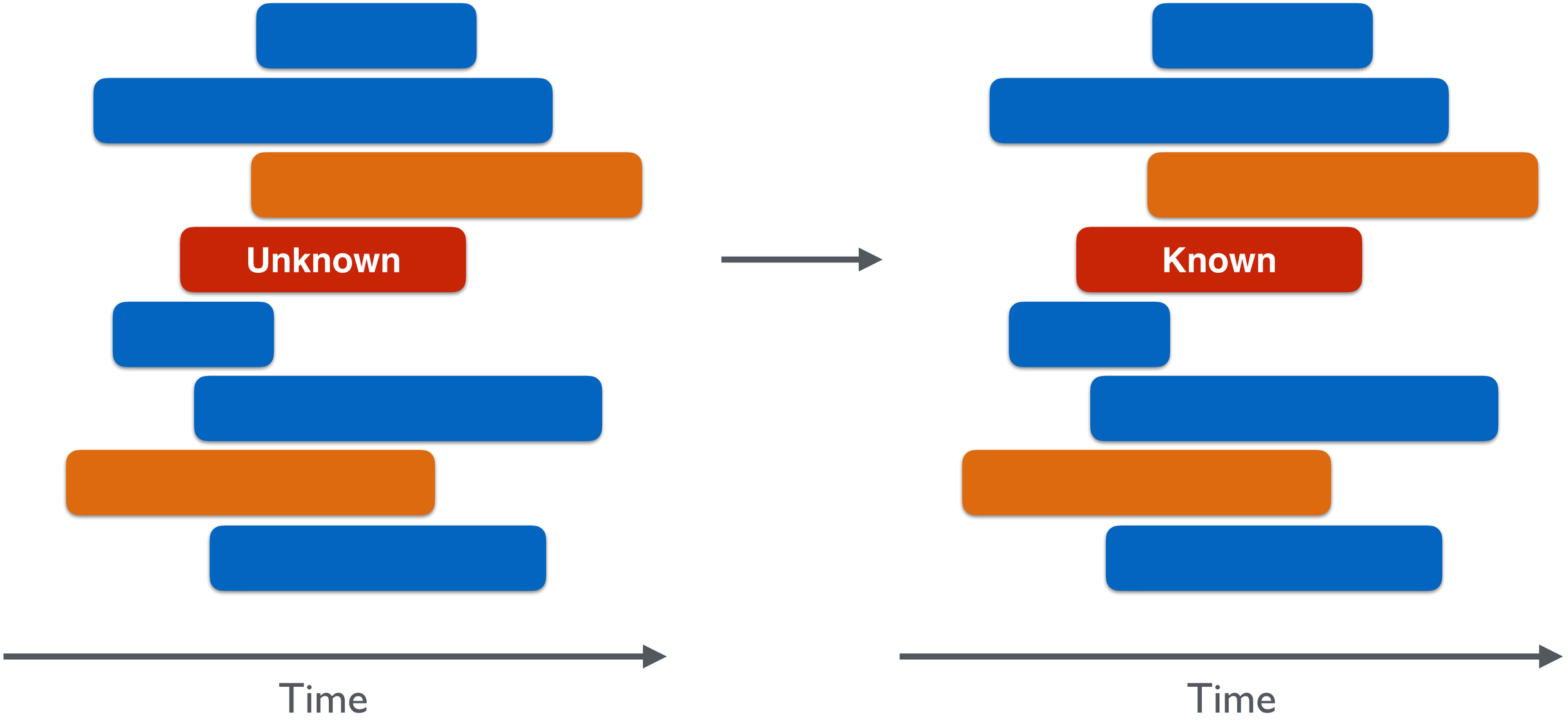

Unknown
Time


Unknown
Known
Time
Time

Q1: Can this flow's performance deteriorate?



Q2: Can the whole system's performance deteriorate?





Q1: Can this flow's performance deteriorate?

Q2: Can the whole system's performance deteriorate?



Q1: Can this flow's performance deteriorate?

Q2: Can the whole system's performance deteriorate?

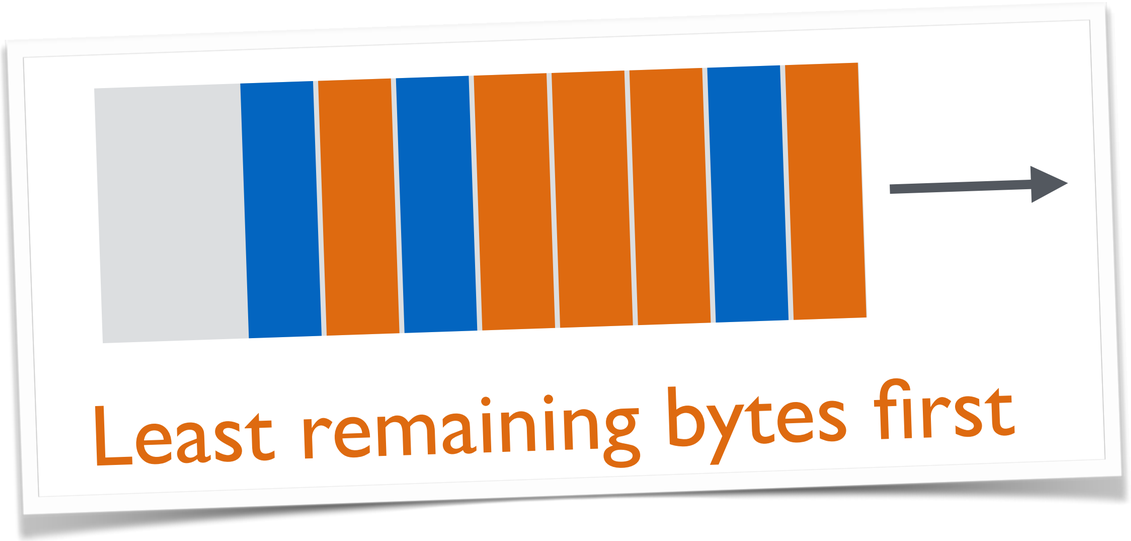
Old scheduling problem, but novel, interesting questions raised by our unique angle!



Q1: Can this flow's performance deteriorate?

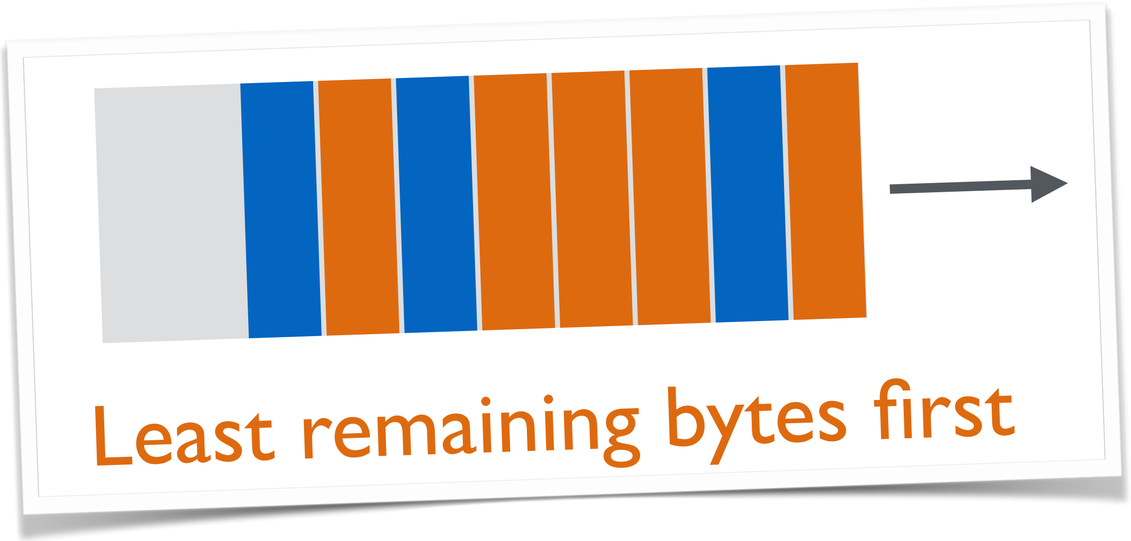
Unknown → Known

Q1: Can this flow's performance deteriorate?



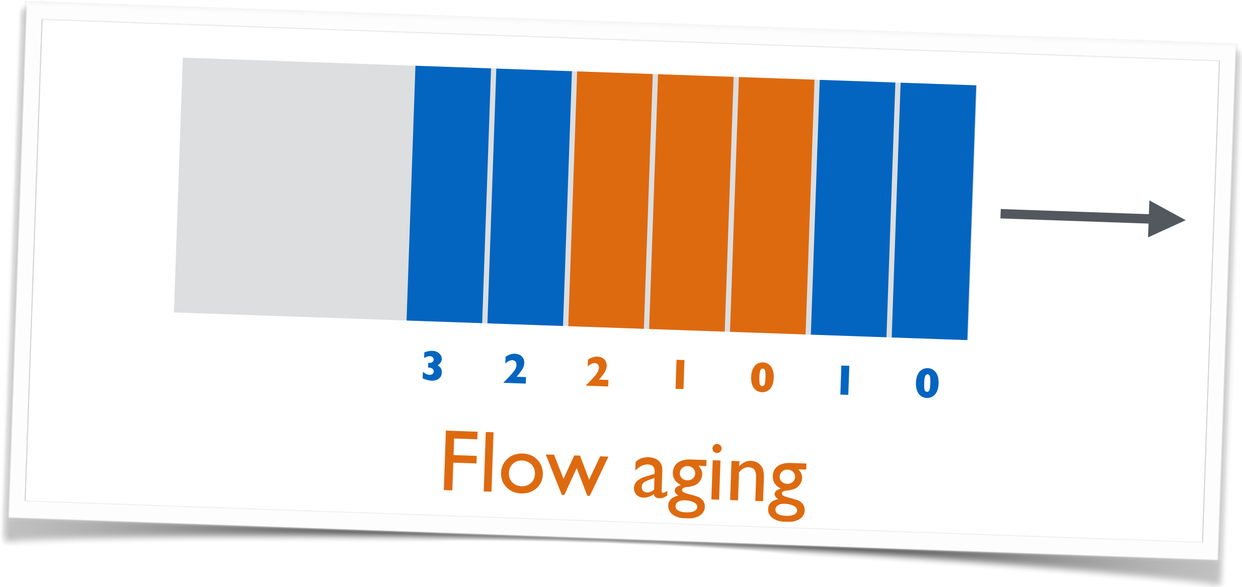
Unknown → Known

Q1: Can this flow's performance deteriorate?



Least remaining bytes first

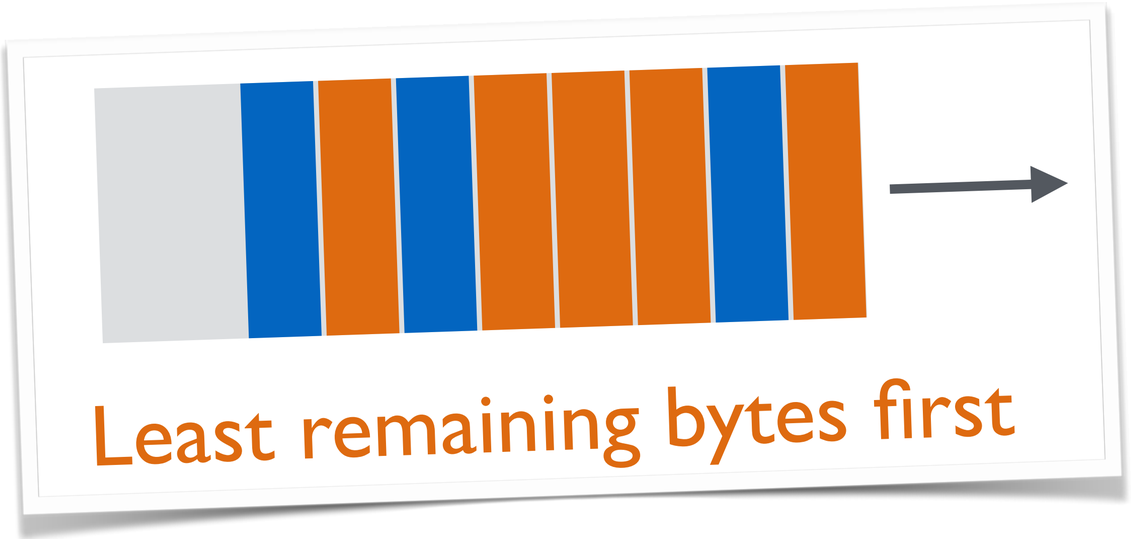
+



Flow aging

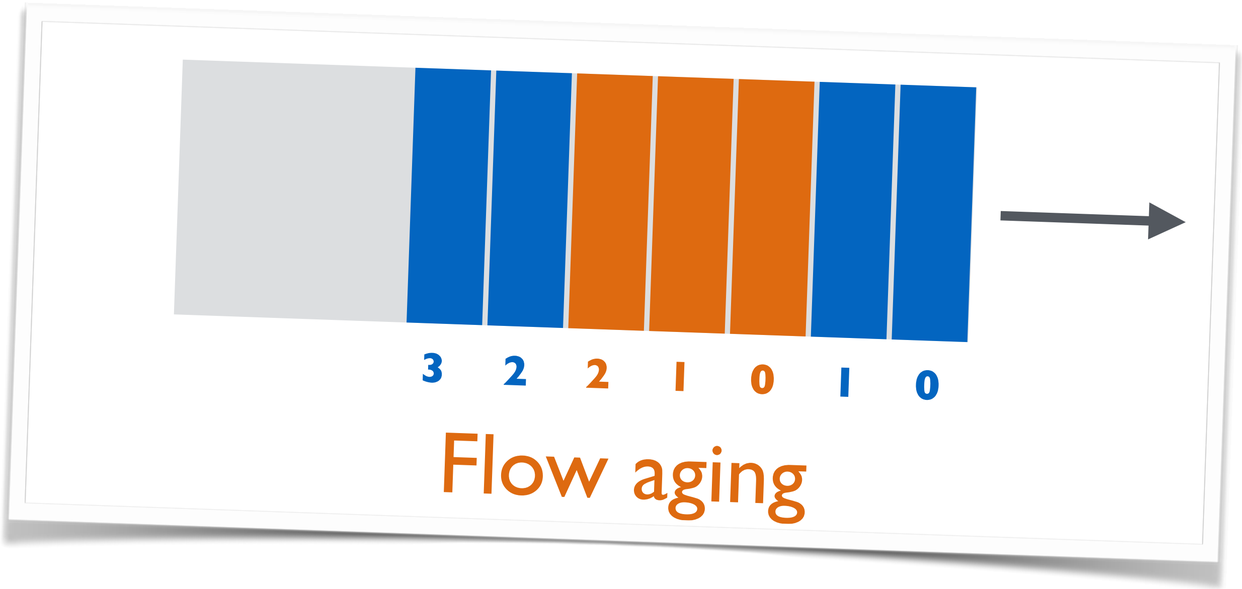
Unknown → Known

Q1: Can this flow's performance deteriorate?



Least remaining bytes first

+



Flow aging

A flow's performance cannot deteriorate!



Q1: Can this flow's performance deteriorate?

For coflow scheduling with mean flow size used for unknowns,
performance can in fact deteriorate



Q2: Can the whole system's performance deteriorate?

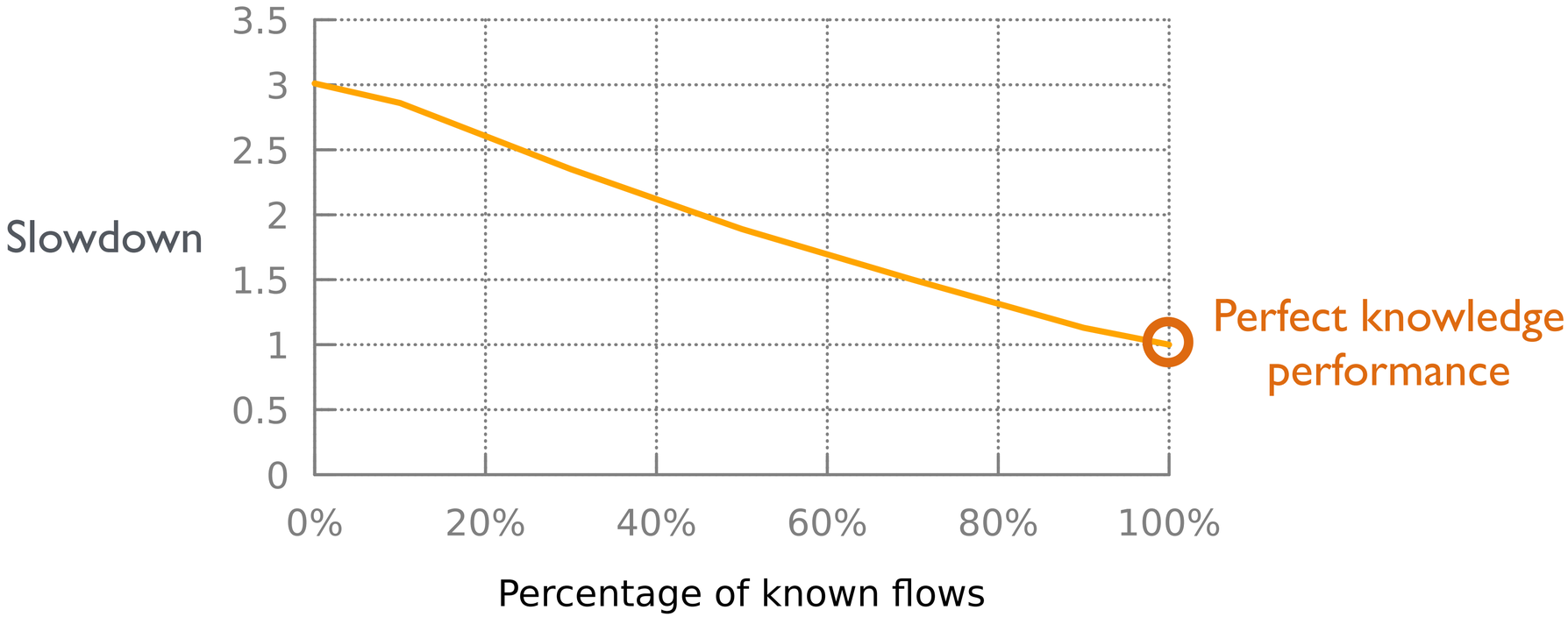


Q2: Can the whole system's performance deteriorate?

Average co/flow completion time system-wide can deteriorate!

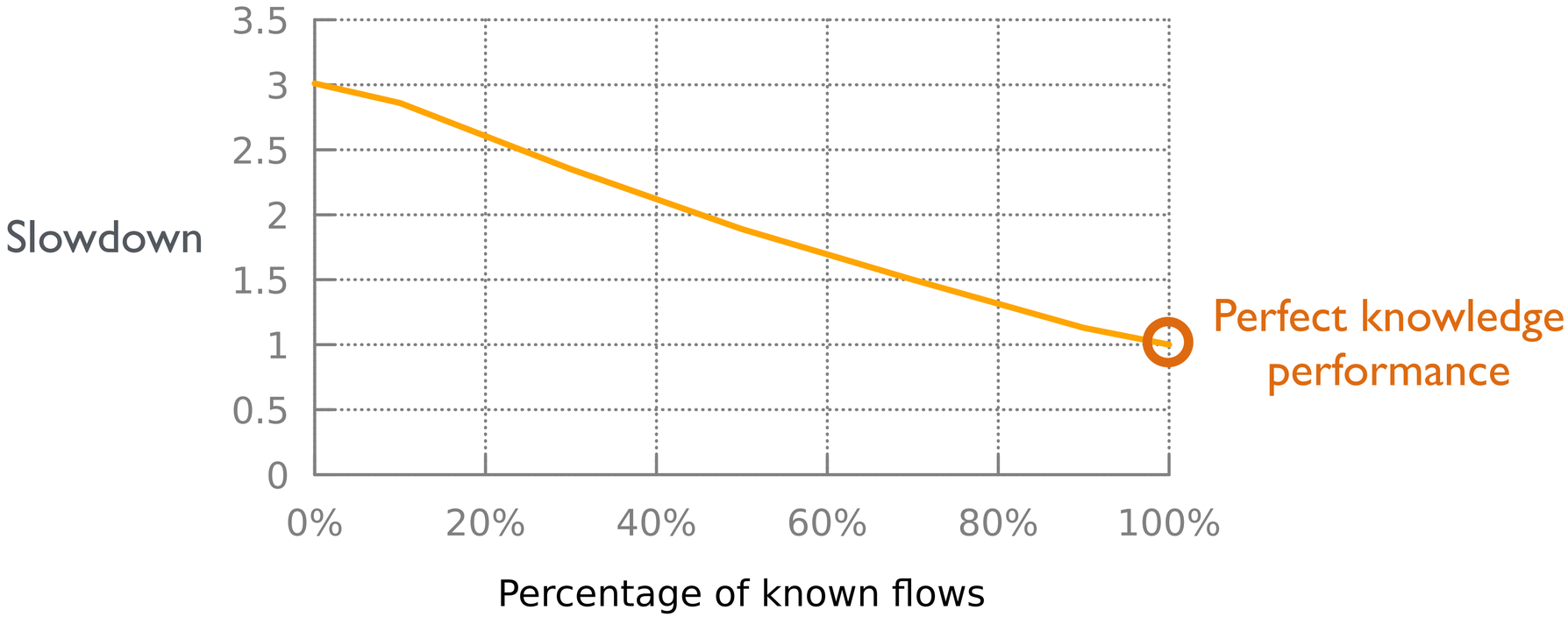
Unknown → Known

Q2: Can the whole system's performance deteriorate?



Unknown → Known

Q2: Can the whole system's performance deteriorate?



Open question: positive results with some restrictions?

Is advance knowledge of flow sizes a plausible assumption?

Sources of flow size information?

Partial knowledge available from many sources

How useful is imprecise / partial knowledge?

Can still provide performance improvements

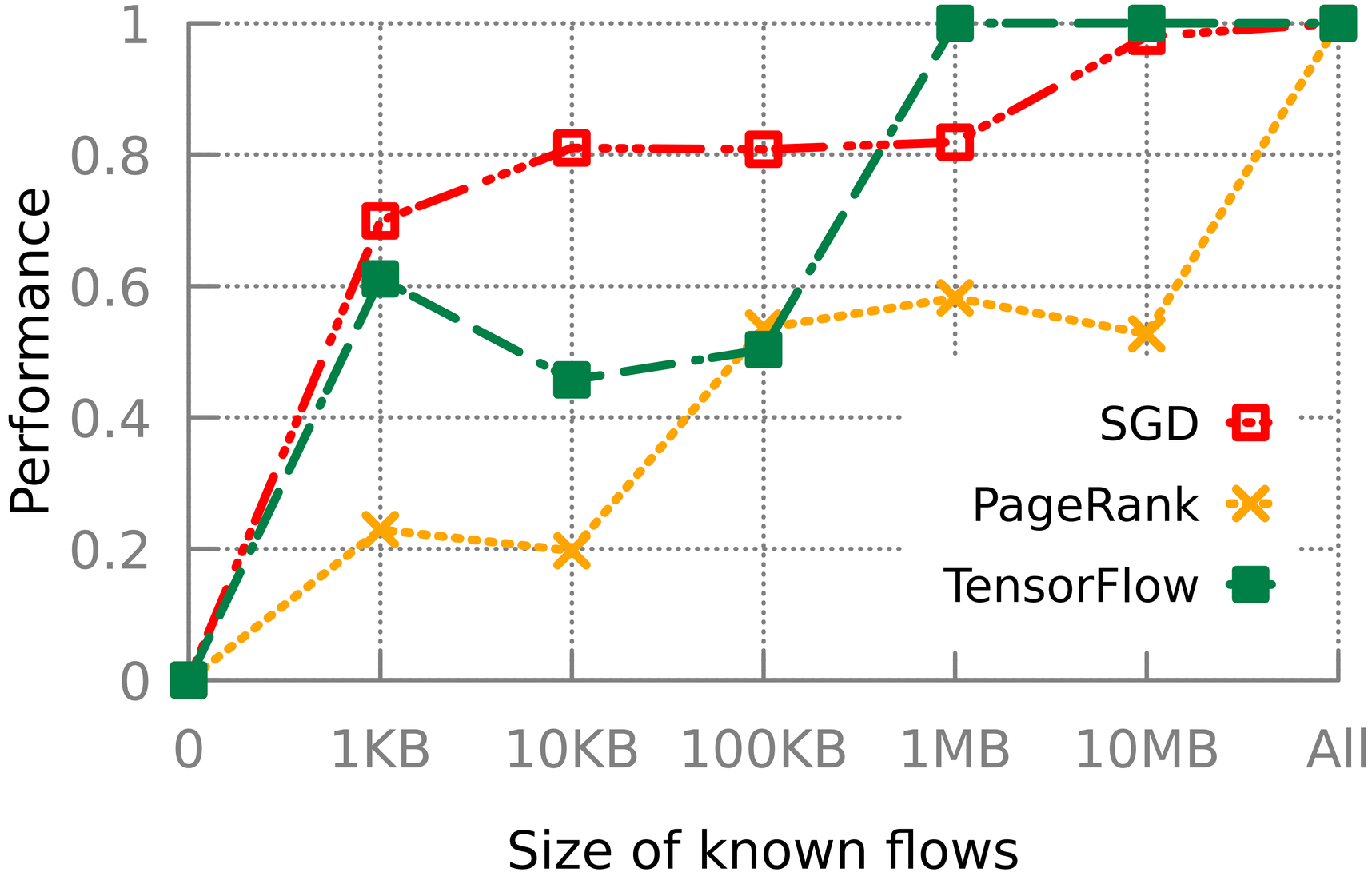
Incentives for operators and users?

Need better guarantees on value of investment

Thank you for your attention

vojislav.dukic@inf.ethz.ch

Performance improvement



Features

Feature	Description
Start time, tf	Start time of f relative to job start time
Flow gap	Time since the end of the previous flow
First call	Size of the first system call tf
Network in	Data received until tf
Network out	Data sent until tf
Network in(d)	Data received from flow's dest. d until tf
Network out(d)	Data sent by this host to d until tf
CPU	CPU cycles used until tf
Disk I/O	Total disk I/O until tf
Memory I/O	Total memory I/O until tf
Previous flows	Flow sizes for last k flows