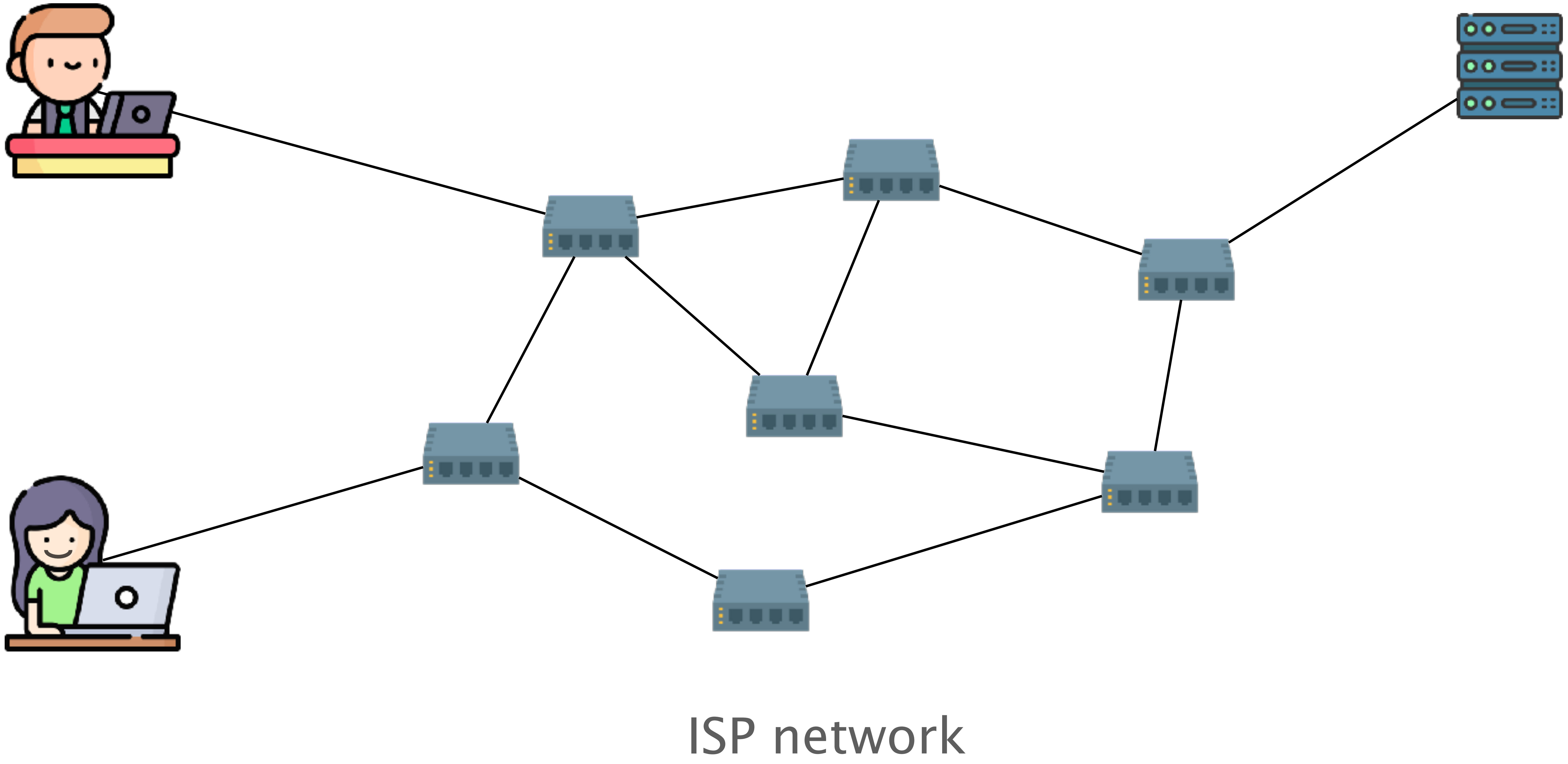
FAst In-Network *Gray* Failure Detection for ISPs

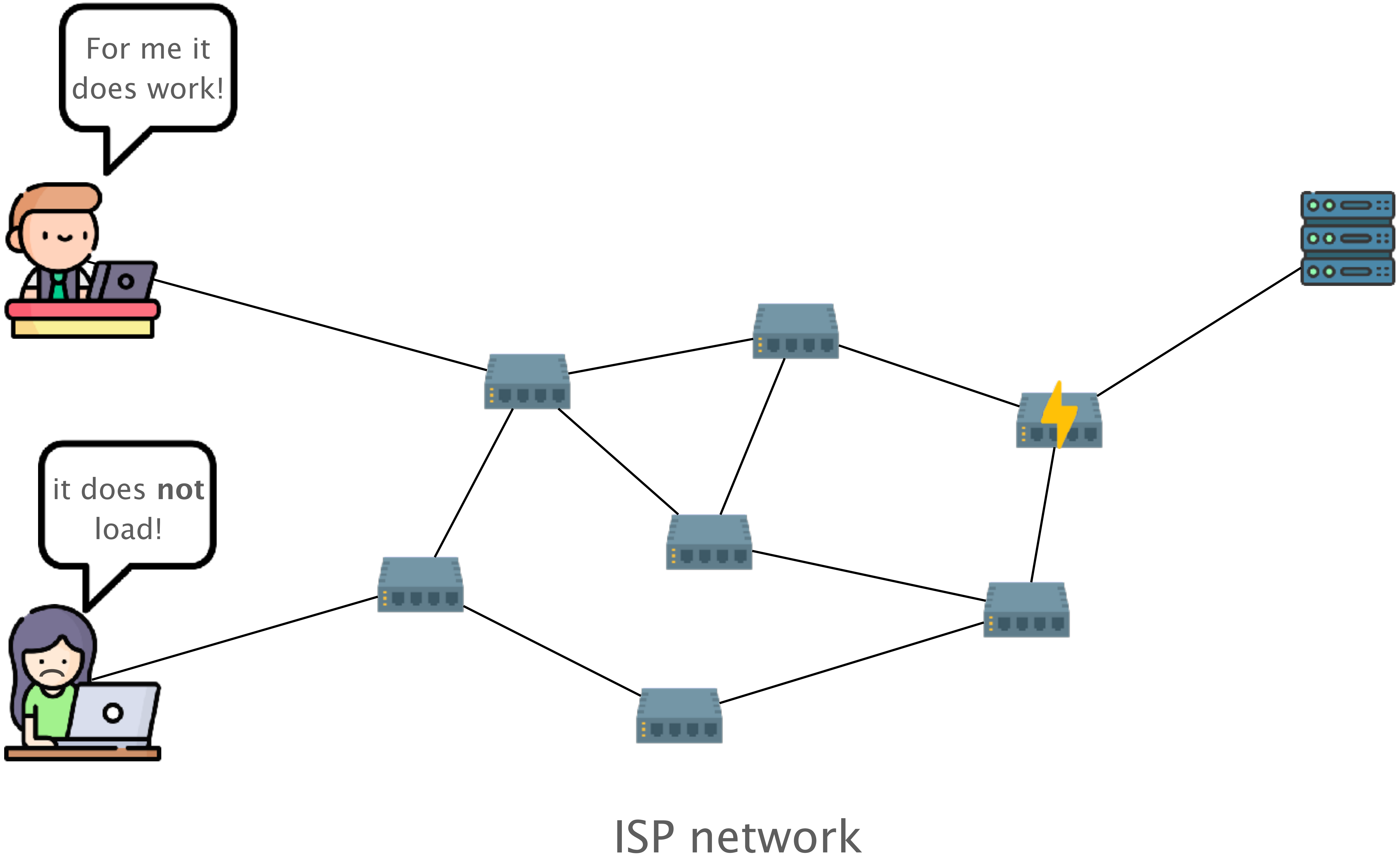
Edgar Costa Molero[1],
Stefano Vissicchio[2],
Laurent Vanbever[1]

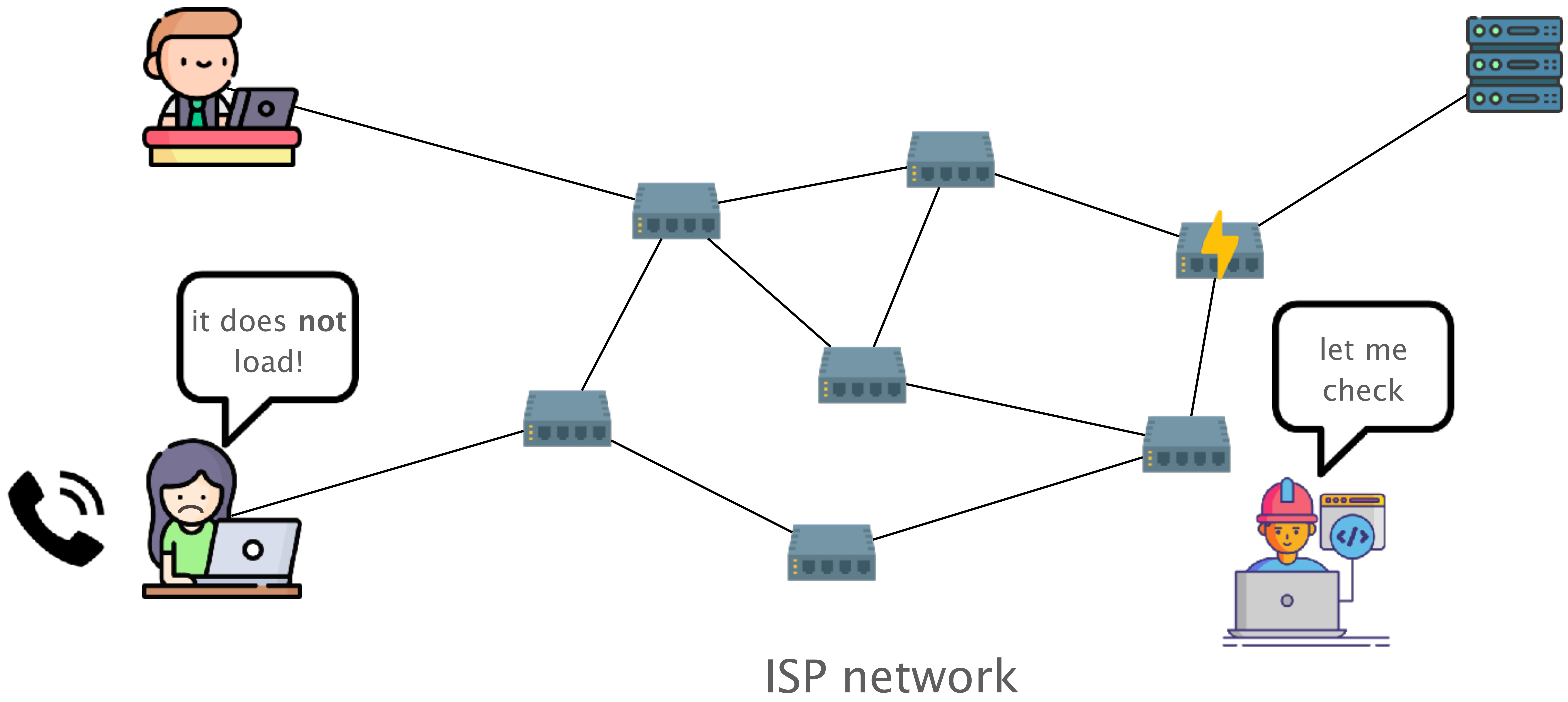
SIGCOMM'22
August, 25 2022

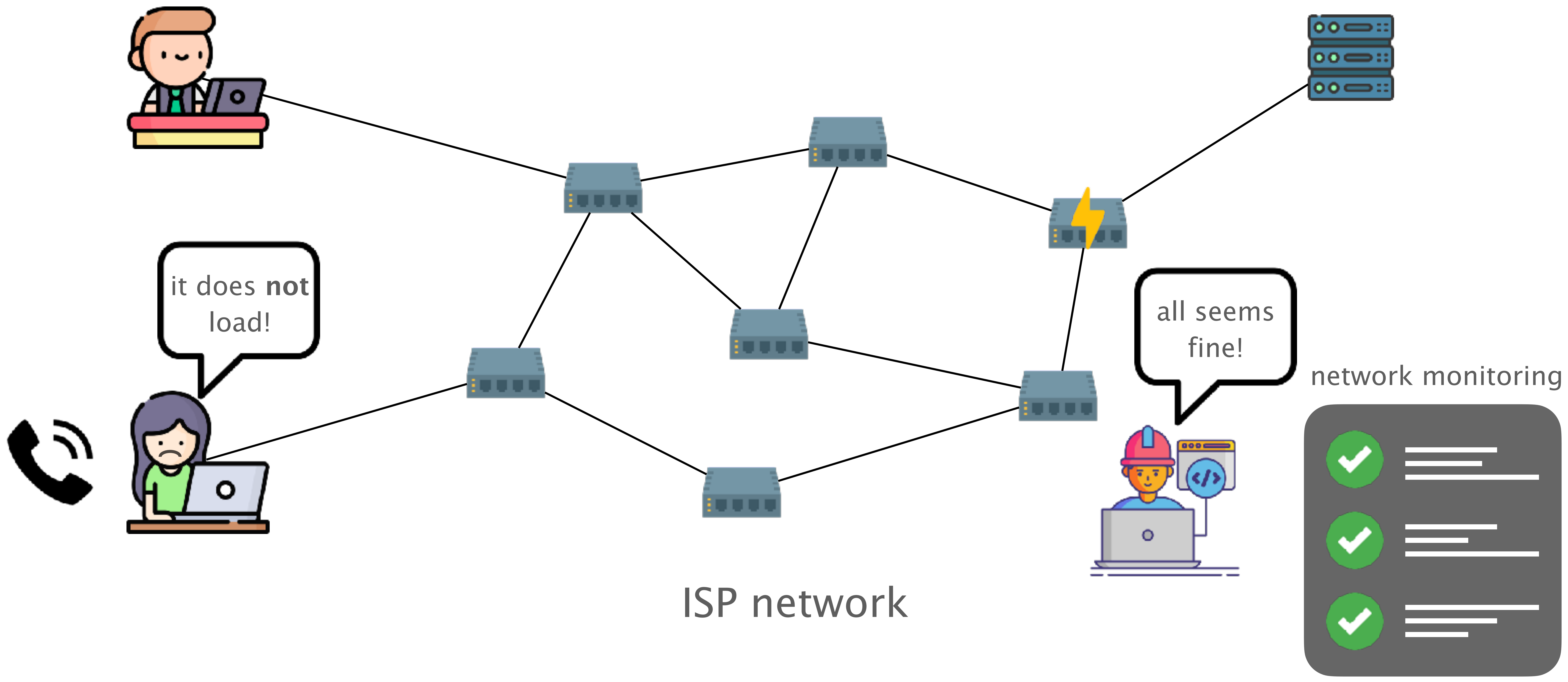
(1) ETH zürich

(2) UCL


ISP network


For me it does work!
it does **not** load!
ISP network


it does **not** load!
let me check
ISP network


it does **not** load!
all seems fine!
network monitoring
ISP network

Gray failures are

Permanent packet loss caused by ***a*** **malfunctioning device** affecting a ***subset of the traffic***

Gray failures…

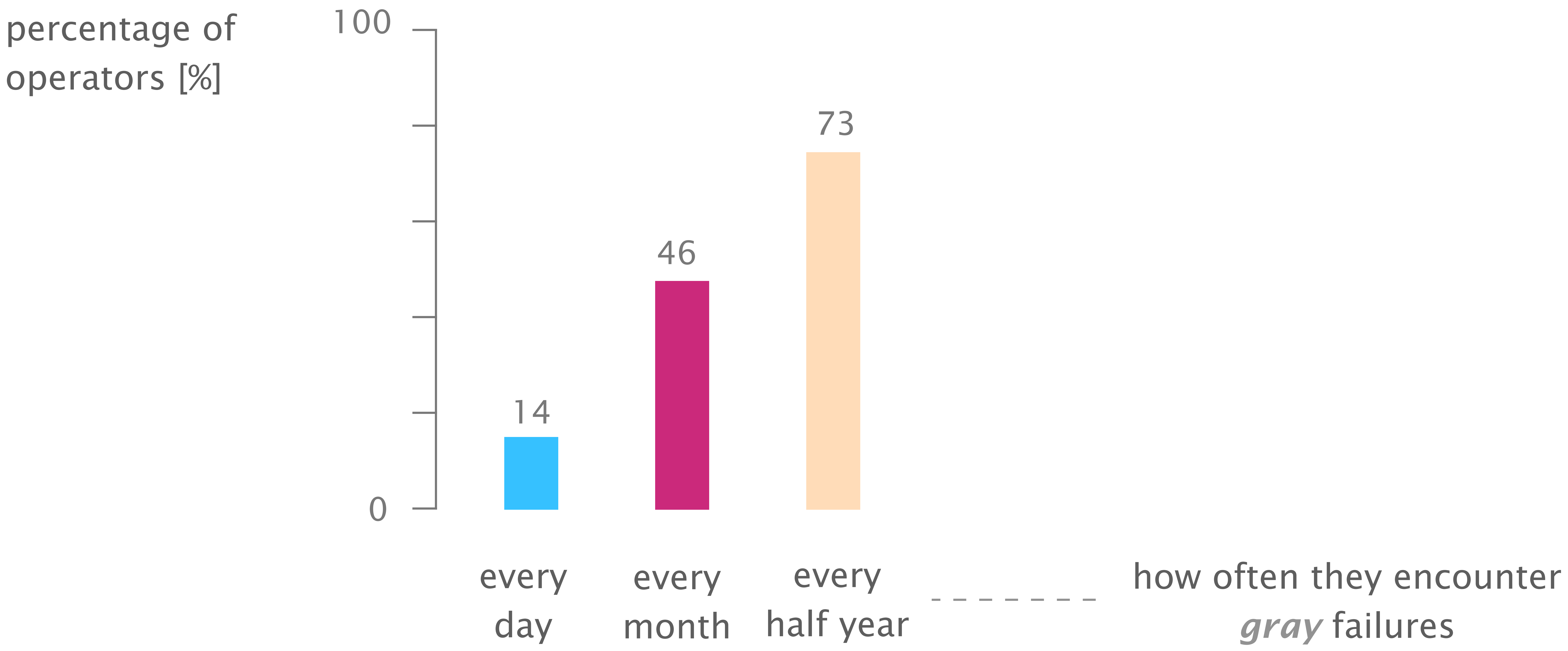
can be caused by

- TCAM bit flips and memory corruption
- bent fibers and not well seated line-cards
- CRC checksum errors
- software bugs and misconfigurations

can affect

- ***single, some*** or ***all*** traffic entries
- ***some*** or ***all*** the packets

Gray failures are a problem for a majority of operators



Detecting and ***locating*** ***gray*** failures requires ***two*** operations

1. **to collect** statistics of ***all*** the traffic

2. ***to compare*** the statistics

Existing *ISP* monitoring techniques fall short because they do not ***collect*** statistics on all the traffic

active

✗ *Heartbeat protocols* (e.g., BFD)
only the heartbeat packets

✗ Sending traffic probes
only selected probes

passive

✗ Packet counters (e.g., SNMP)
only available switch counters

✗ NetFlow or sFlow
only if sampled

Most ***data center*** ***gray*** failure detection solutions do ***collect statistics*** on all traffic and ***compare*** them.

However, they still ***fall*** short in ***ISPs networks.***

Why?

The characteristics of ***ISP networks*** make ***data center*** failure detection systems ***not operational***

- No end-point control
  only control network devices

- High link bandwidth
  100 Gbps and increasing

×

- High latency between devices
  in the order of ms

Data center *gray* failure detection systems require more ***memory*** than available in switches to operate in ***ISP networks***

required memory to operate

collection rate	×	**collection** complexity	×	**min** time to **compare**	>>	switch memory
link bandwidth		*memory required per packet*		*link delay* *control plane reading speed*		*fixed*

Introducing

FANcY: Fast In-network ***Gray*** Failure Detection for ISPs

We designed FANcY to work with **ISP network characteristics**

required memory to operate

collection rate × **collection** complexity × **min** time to **compare** < switch memory

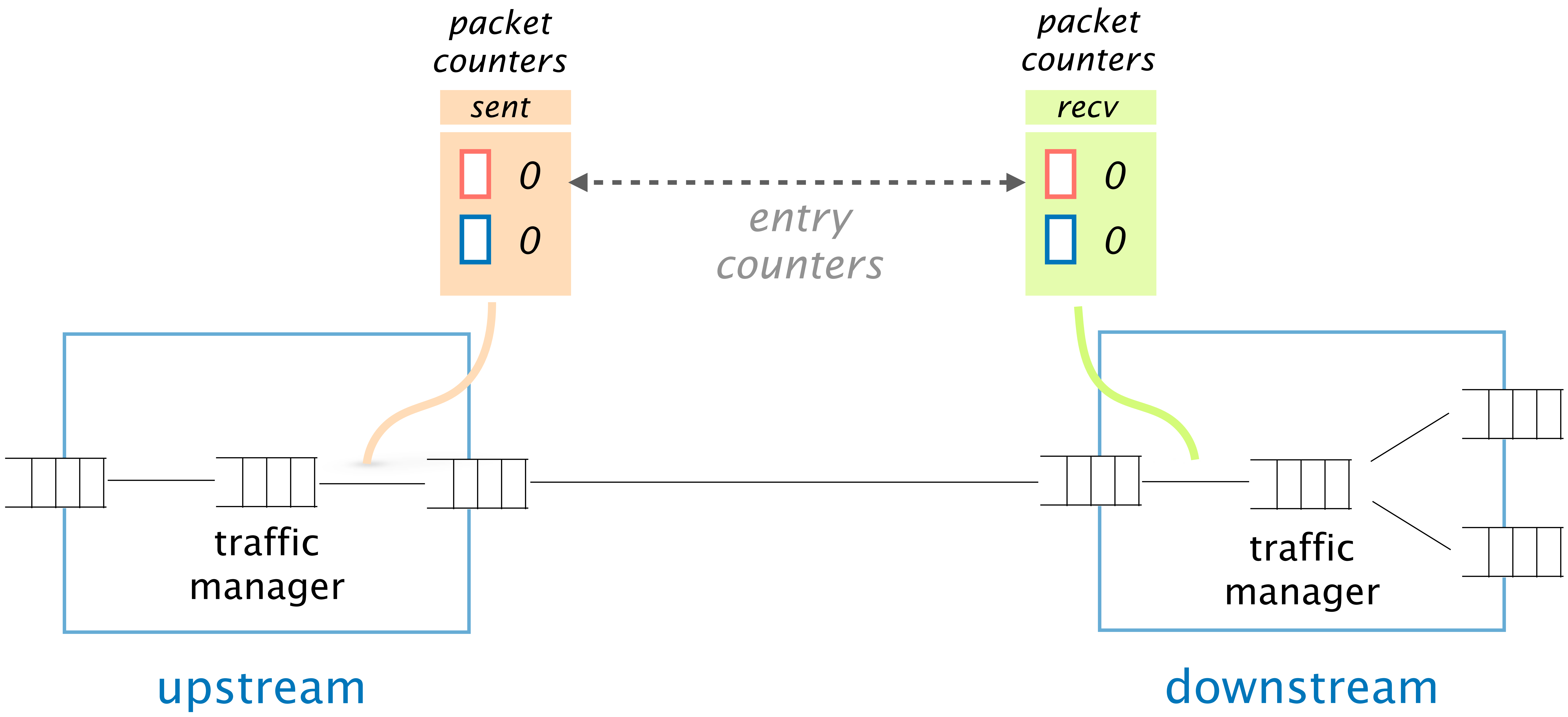
collection rate		**collection** complexity		**min** time to **compare**		switch memory
link bandwidth	×	*memory required per packet*	×	*link delay* / ~~*control plane reading speed*~~	<	*fixed*

#1 Collected statistics are aggregated per traffic entry in simple counters

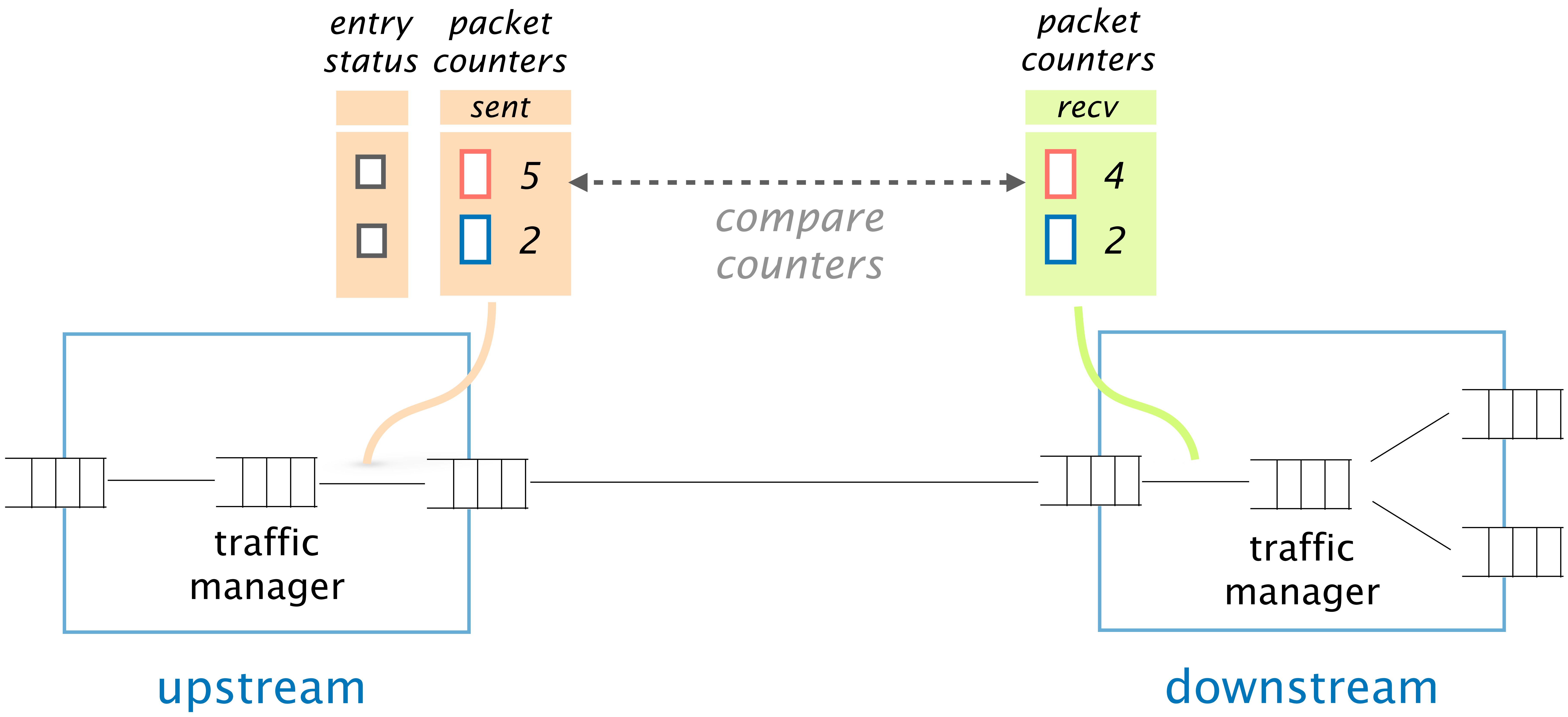
#2 FANcY compares the collected statistics directly in the data plane

FANcY works in switch pairs and detects failures at the port level

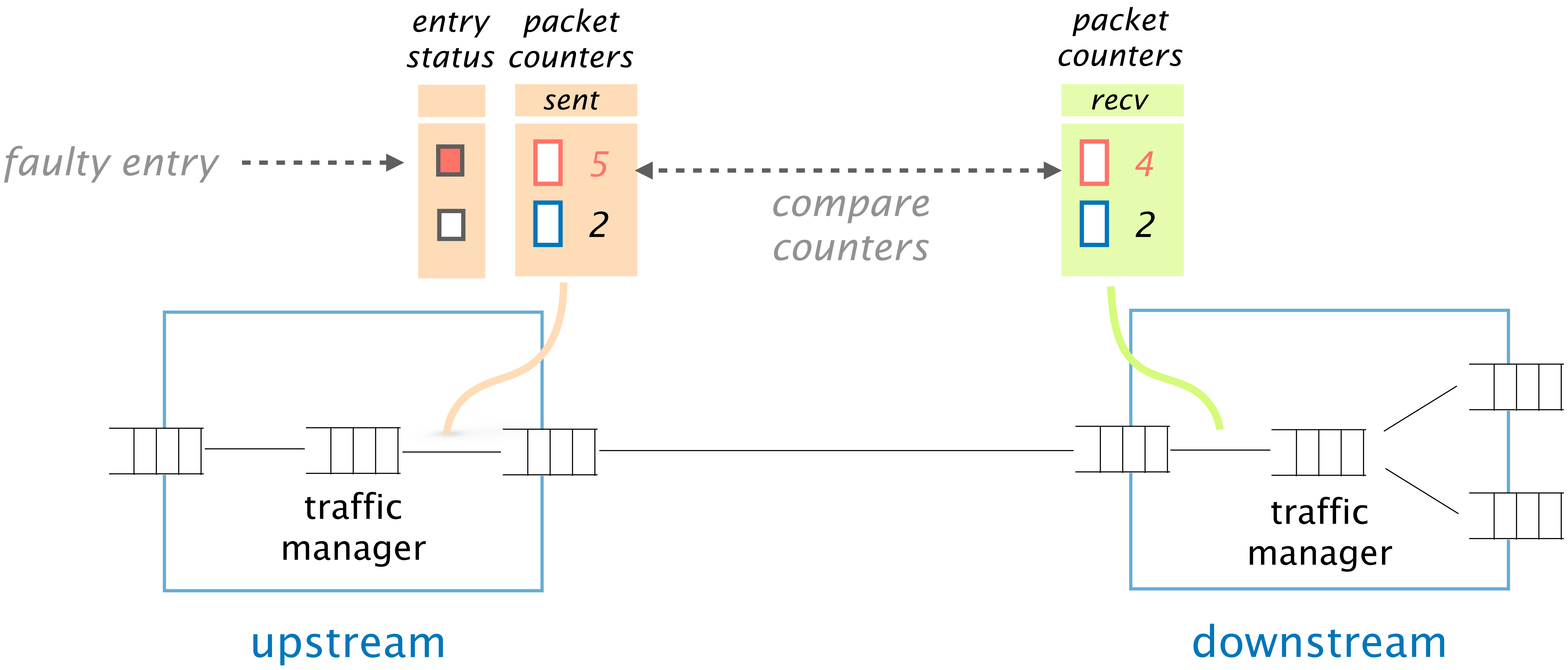
For each ***traffic entry*** the ***upstream*** and ***downstream*** switches use a packet ***counter*** to collect statistics



After collecting statistics, the **upstream** and the ***downstream*** compare its counters in order to find ***discrepancies***



If counters mismatch, the ***upstream*** flags the entry as ***faulty***



Our design has ***two*** main ***challenges***

#1 Synchronizing *our packet counts and make them reliable*

FANcY establishes counting sessions for each counter pair

#2 Scaling *to many traffic entries*

FANcY uses a hybrid approach to support a big number of entries

Our design has ***two*** main ***challenges***

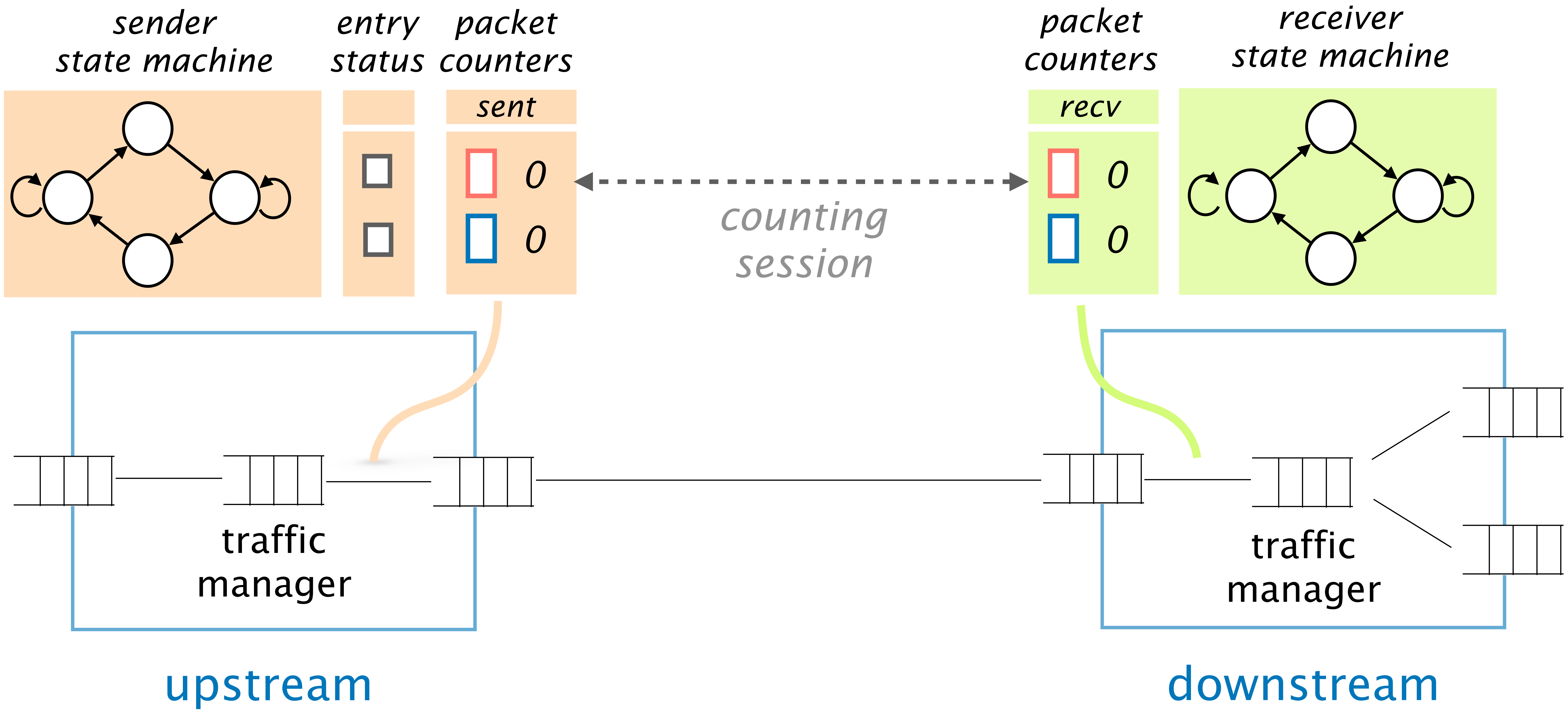
#1 Synchronizing *our packet counts and make them reliable*

FANcY establishes counting sessions for each counter pair

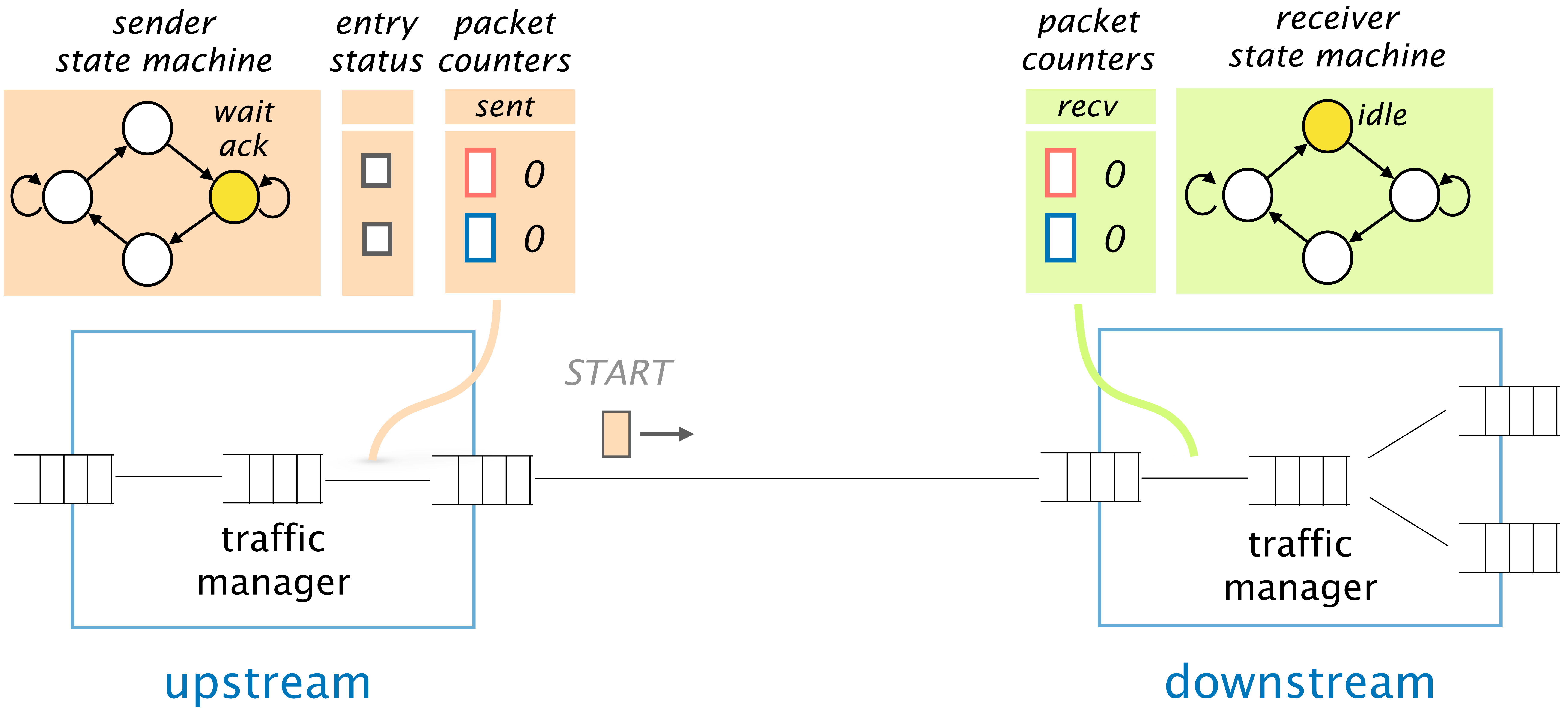
#2 Scaling *to many traffic entries*

FANcY uses a hybrid approach to support a big number of entries

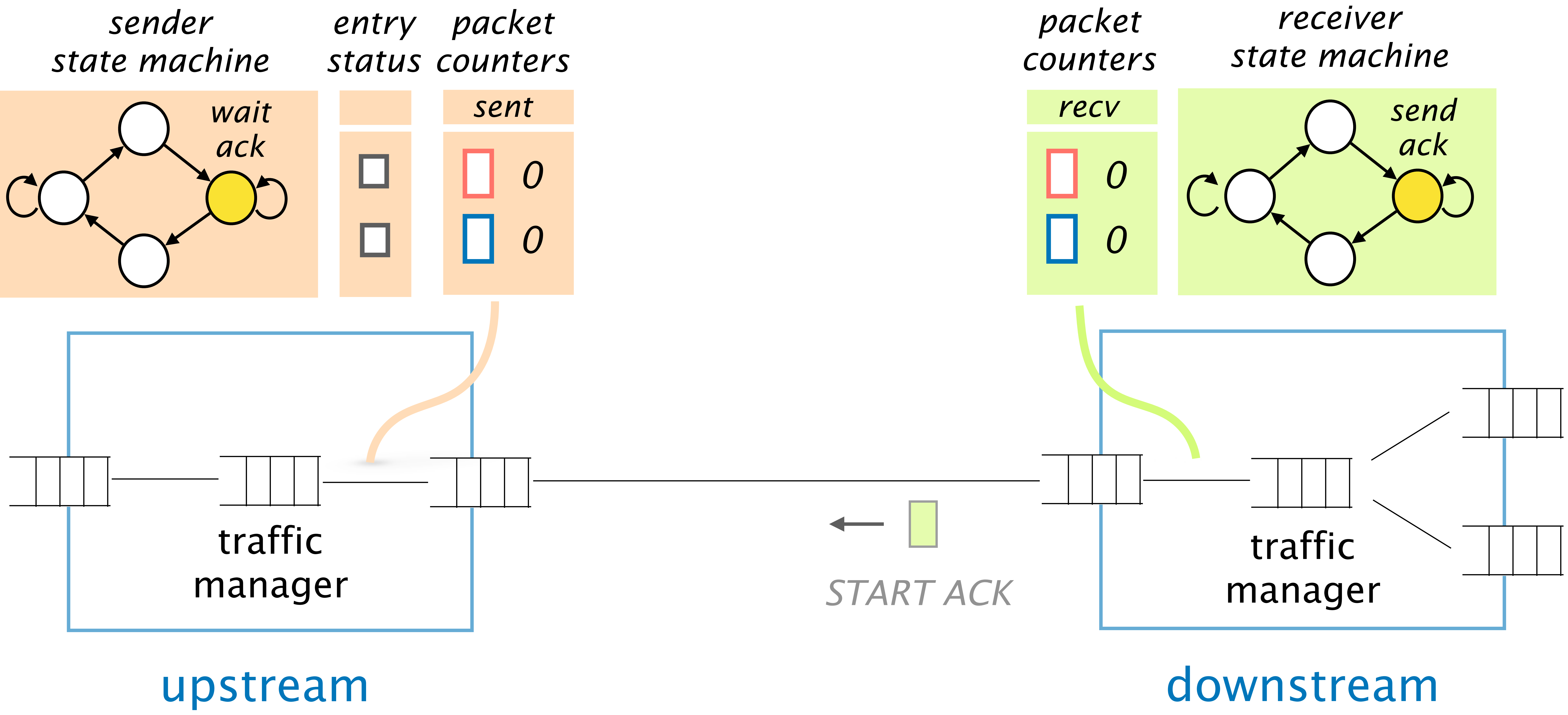
To achieve perfect **synchronization** and **reliability**
FANcY uses **state machines** for each counting session



To achieve perfect **synchronization** and **reliability**
FANcY uses **state machines** for each counting session

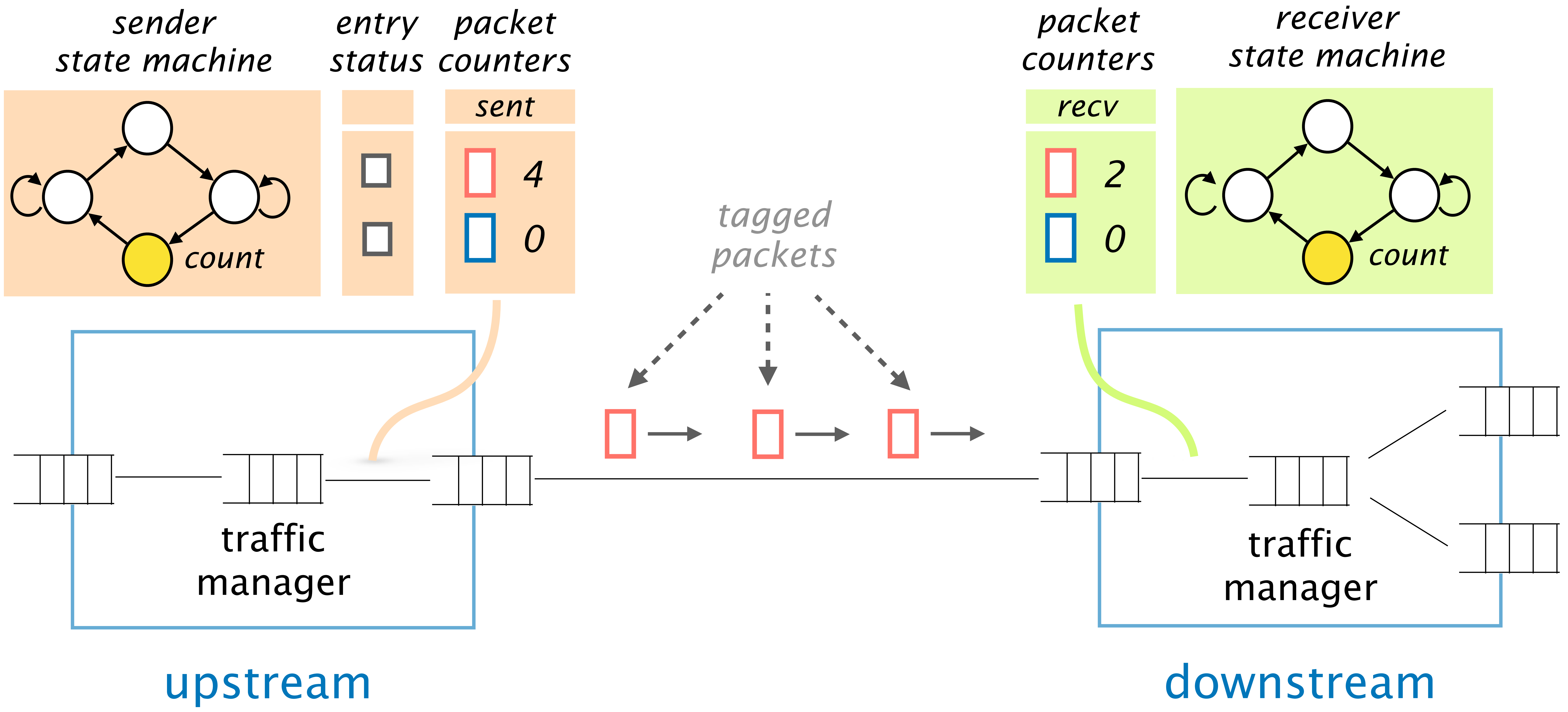


To achieve perfect **synchronization** and **reliability**
FANcY uses **state machines** for each counting session



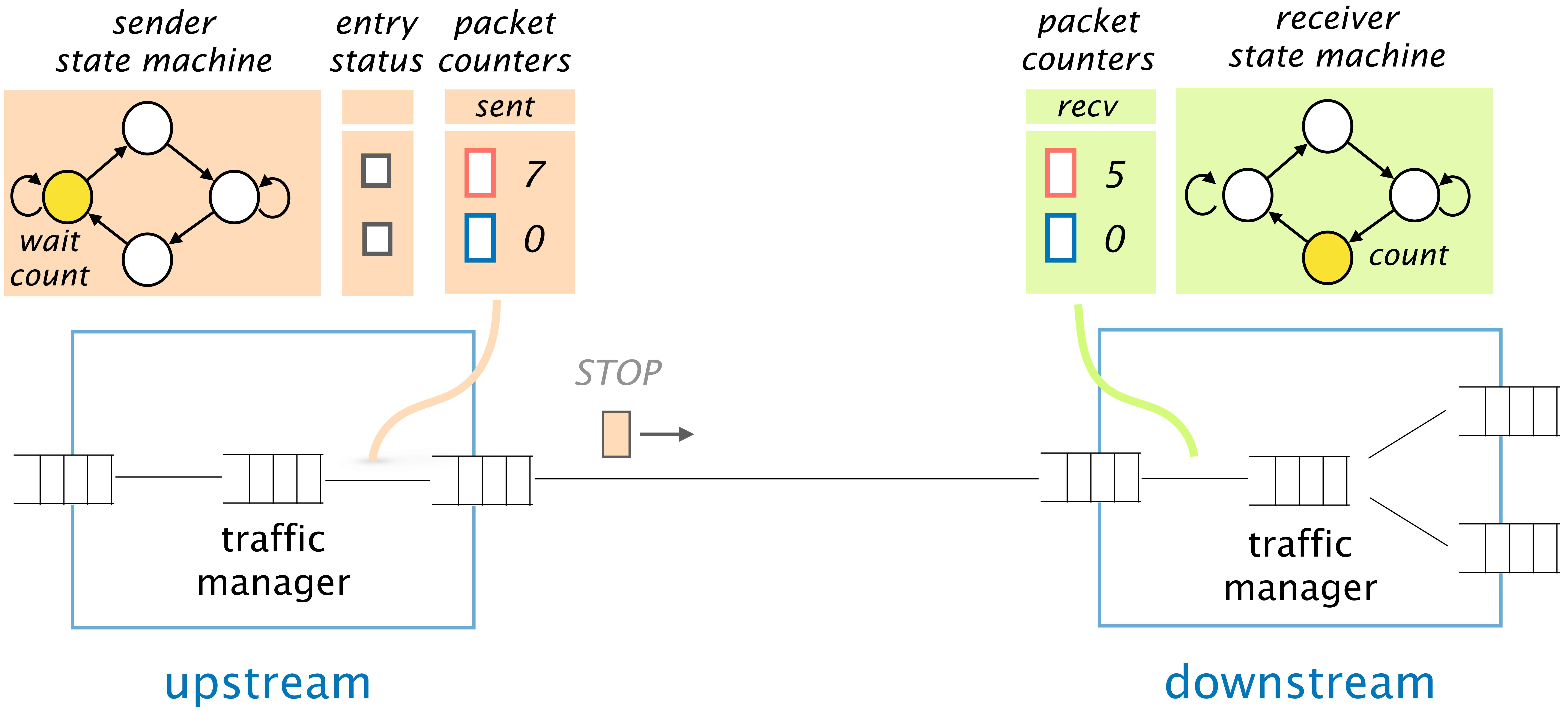
To achieve perfect **synchronization** and **reliability**

FANcY uses **state machines** for each counting session

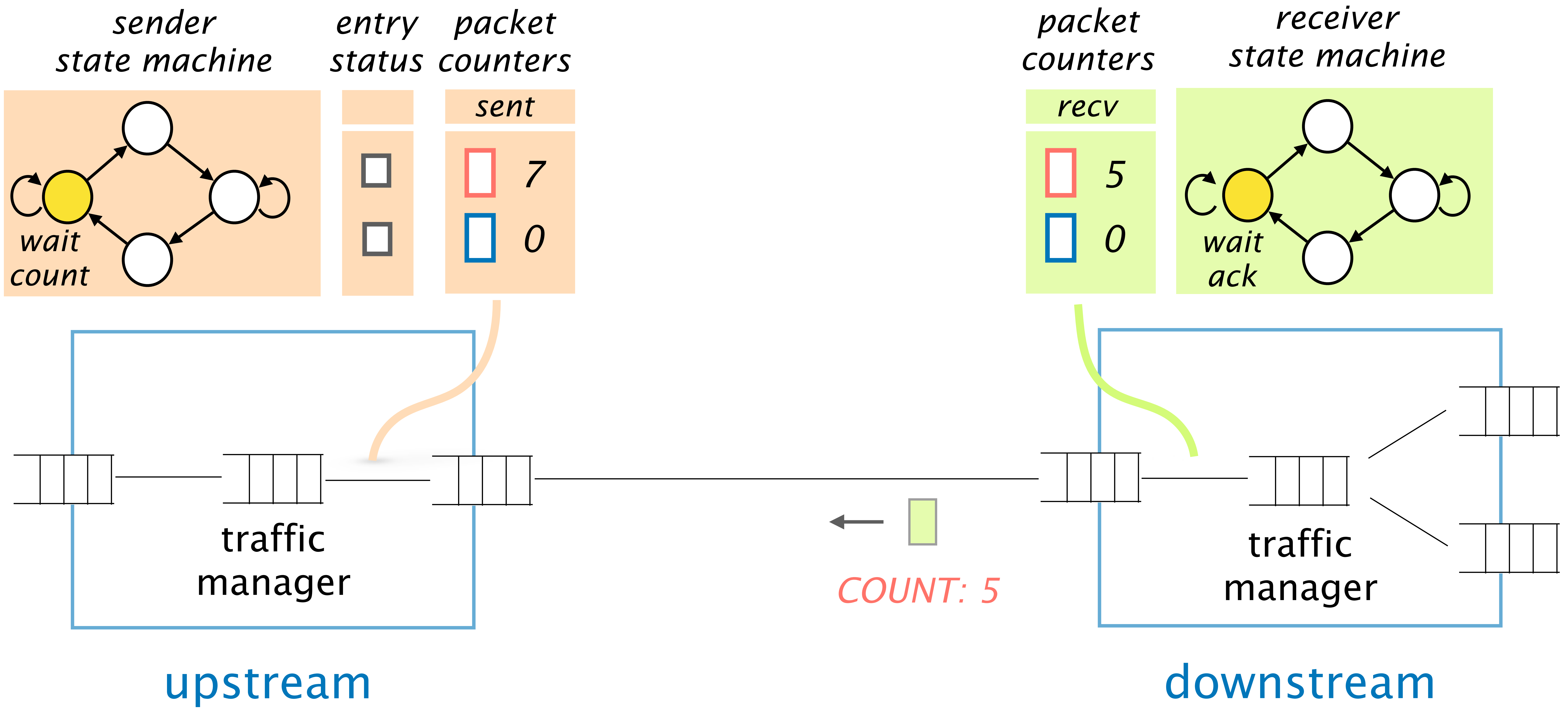


To achieve perfect **synchronization** and **reliability**

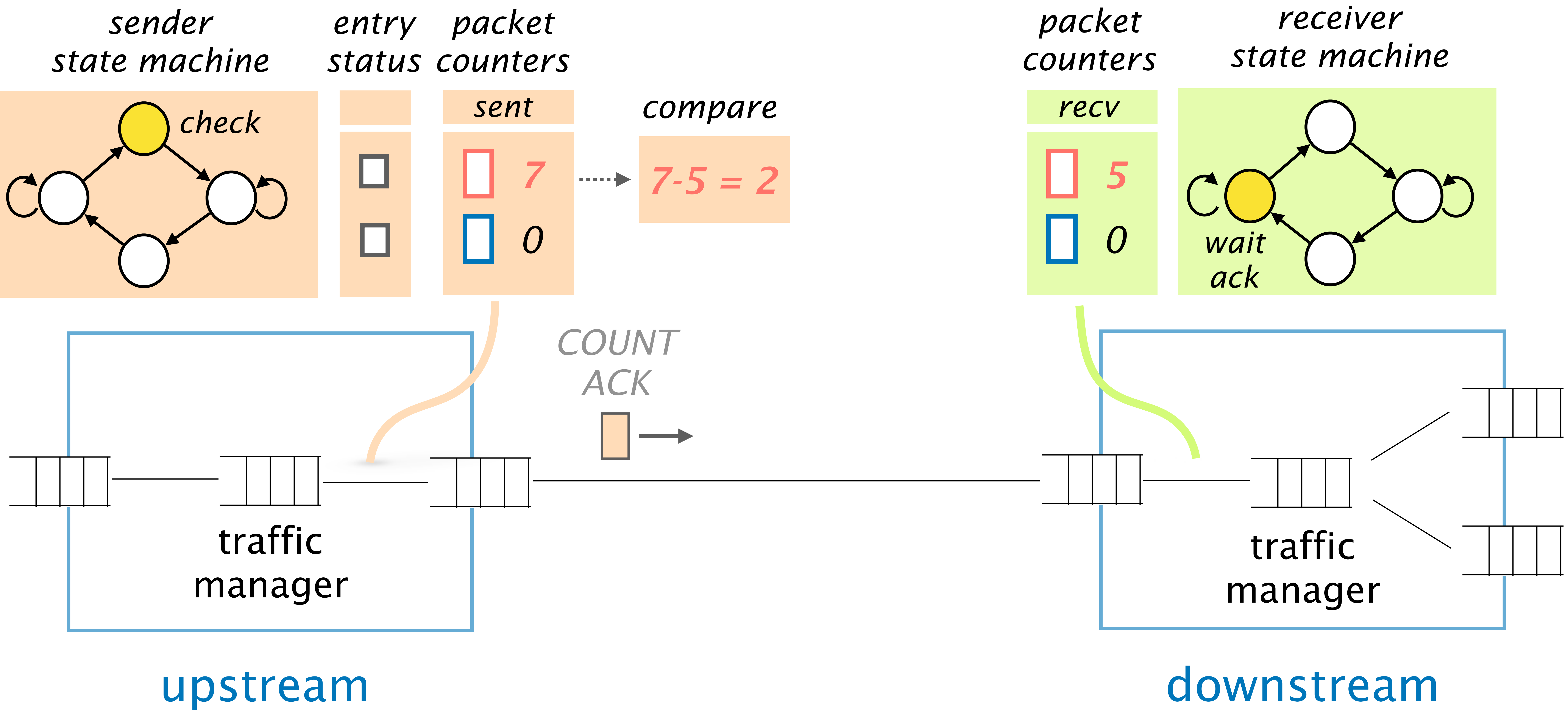
FANcY uses **state machines** for each counting session



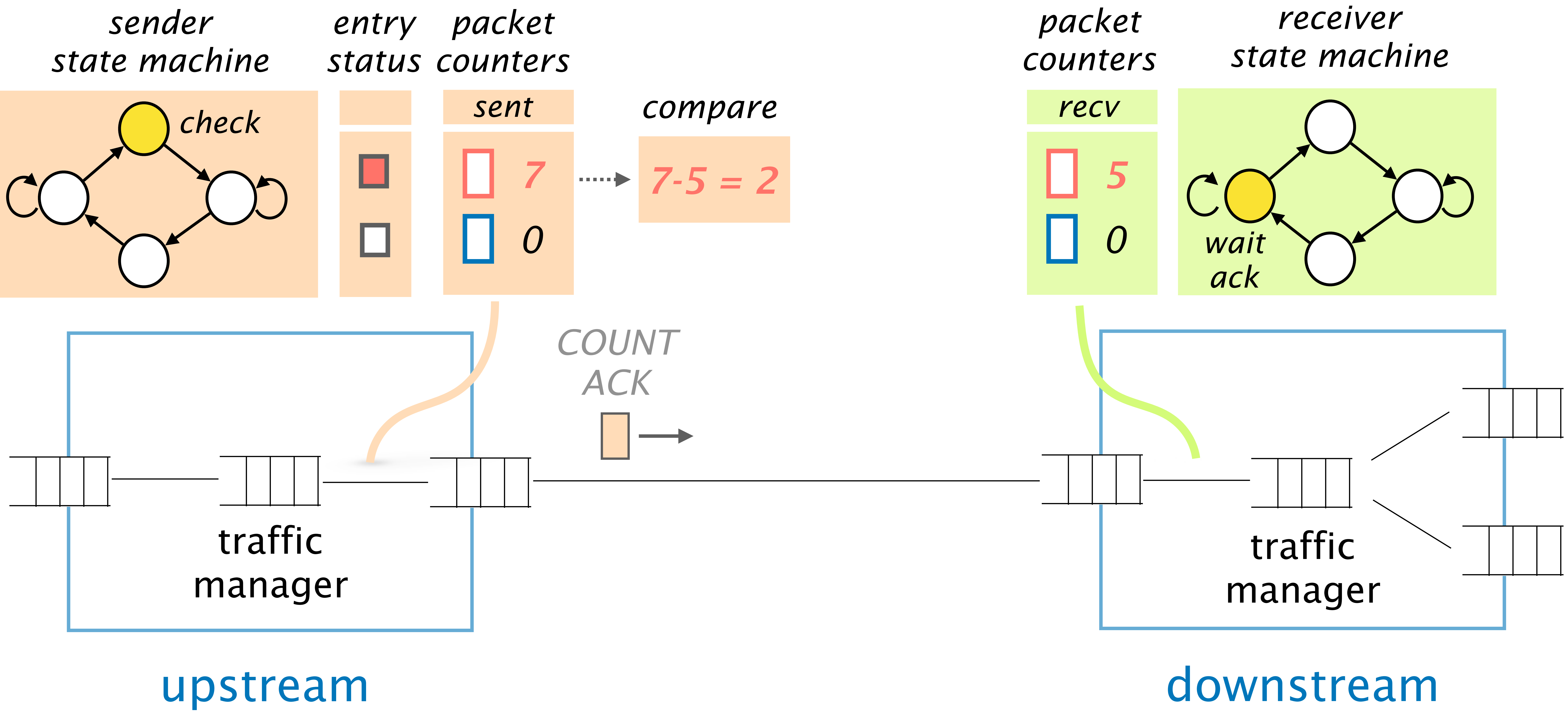
To achieve perfect **synchronization** and **reliability**
FANcY uses **state machines** for each counting session



The upstream checks if there is any ***discrepancy*** between counters



The upstream checks if there is any ***discrepancy*** between counters
If so, it ***flags*** the entry as ***faulty***



Our design has ***two*** main ***challenges***

#1 Synchronizing *our packet counts and make them reliable*

FANcY establishes counting sessions for each counter pair

#2 Scaling *to many traffic entries*

FANcY uses a hybrid approach to support a big number of entries

Having a ***pair of counters*** and **state machines** per traffic entry ***does not scale***

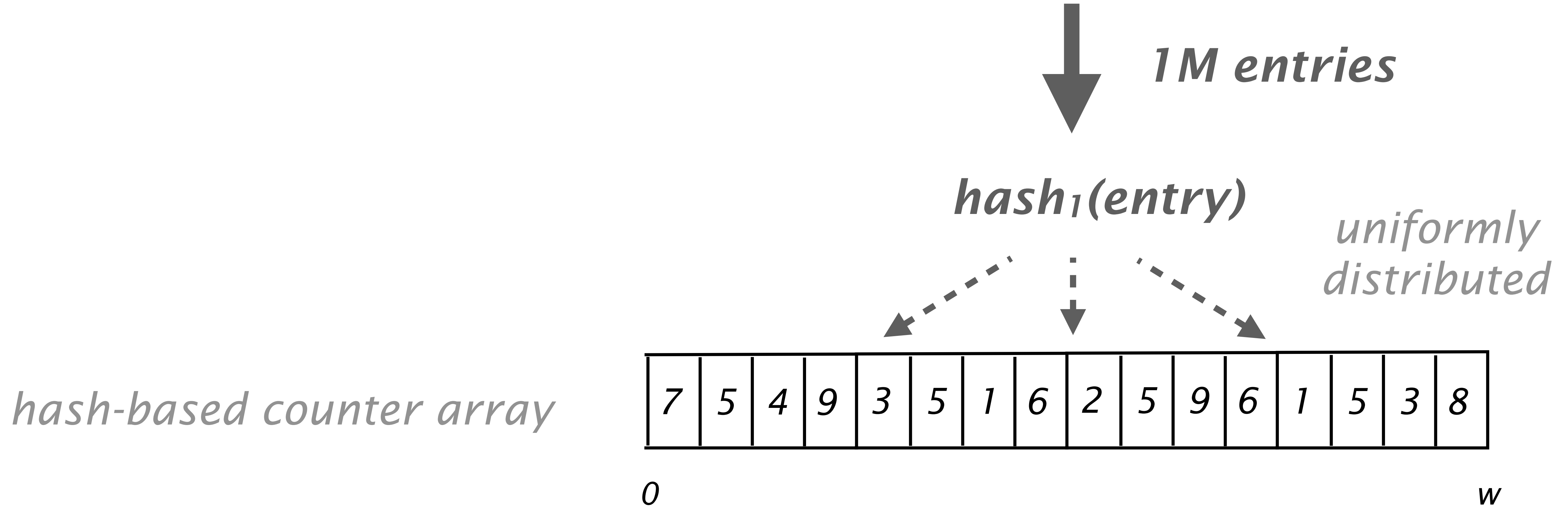
Each pair of counters and state machines requires 160 bits

If we want to track ***1M entries*** (i.e all prefixes in the internet) we need:

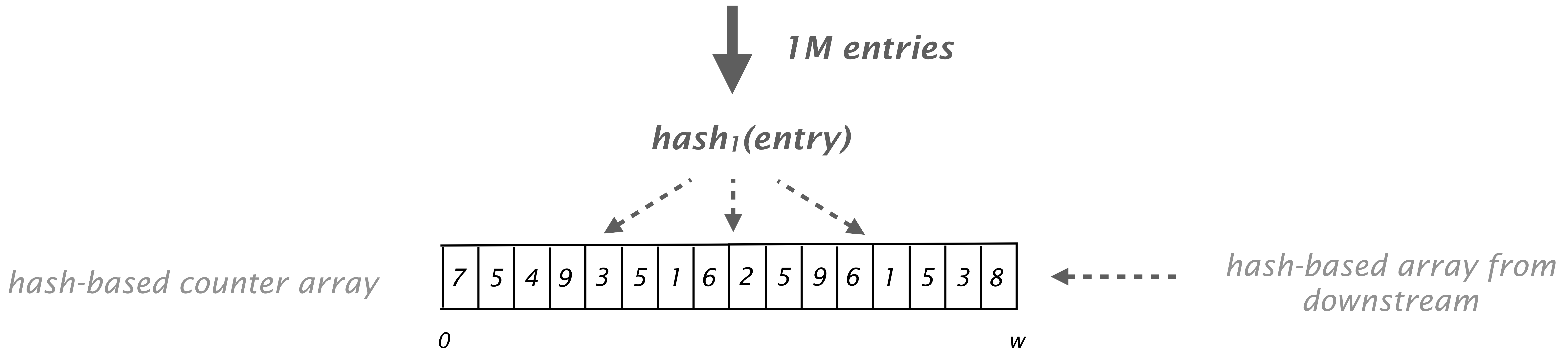
~1.25 GB for a 64 port switch!

We can leverage the fact that ***gray*** failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***

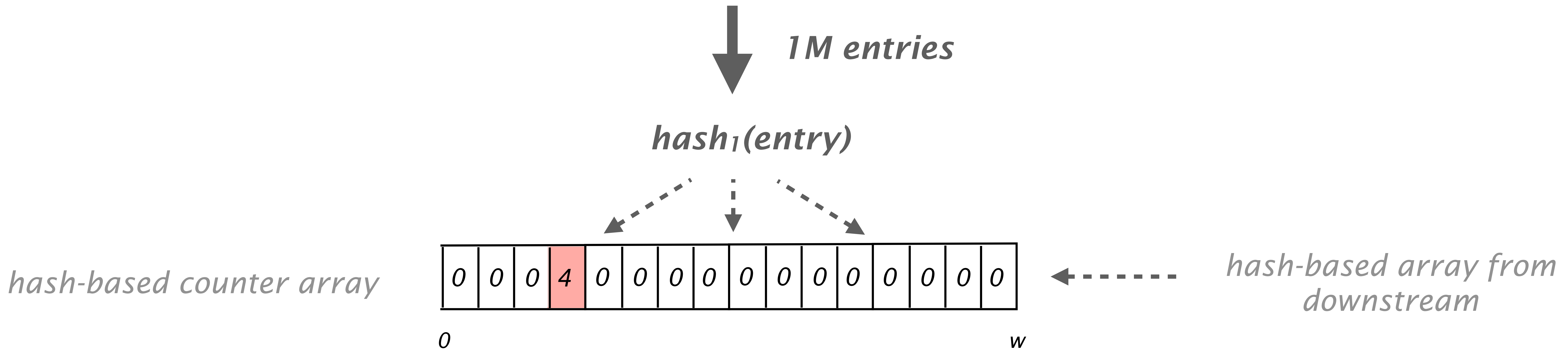
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



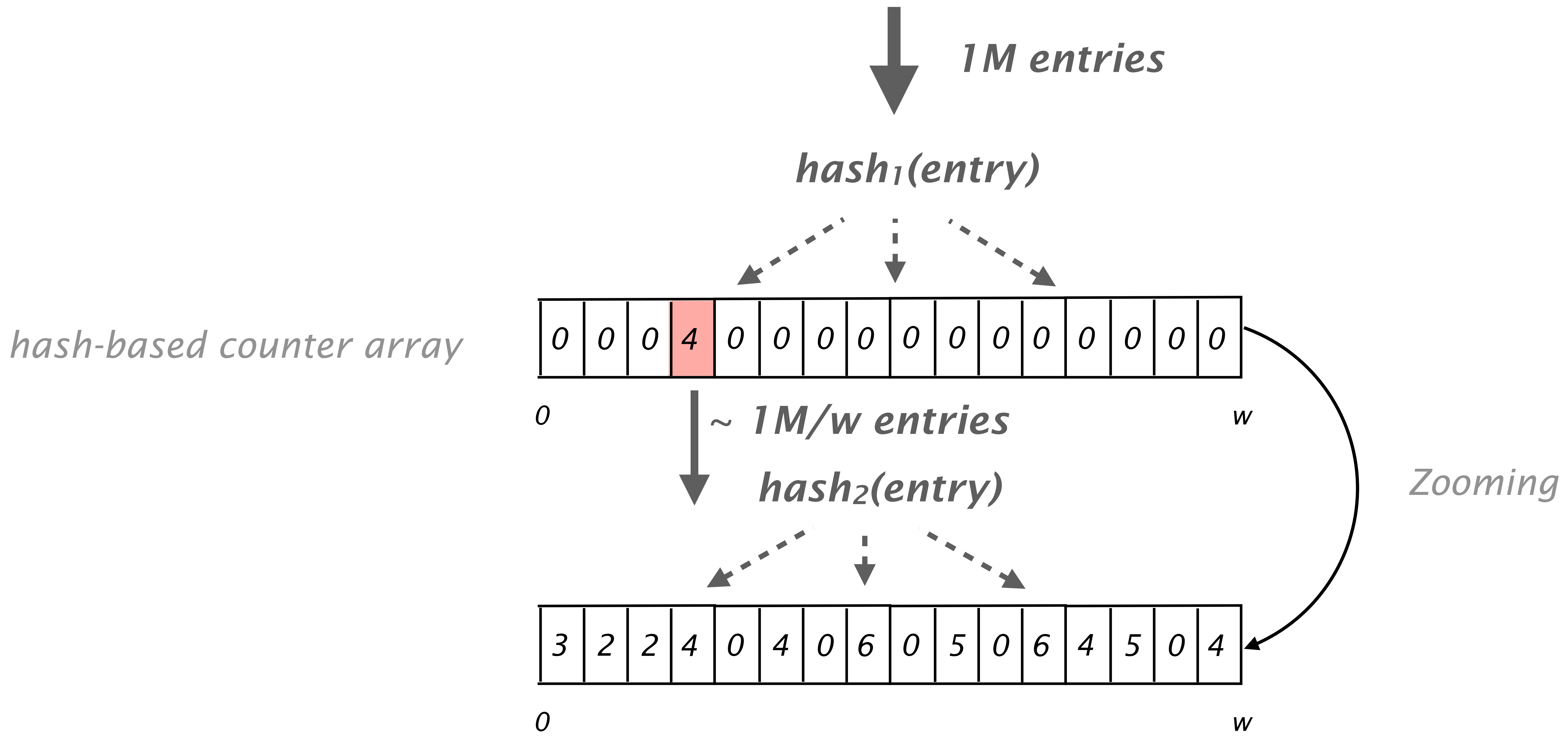
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



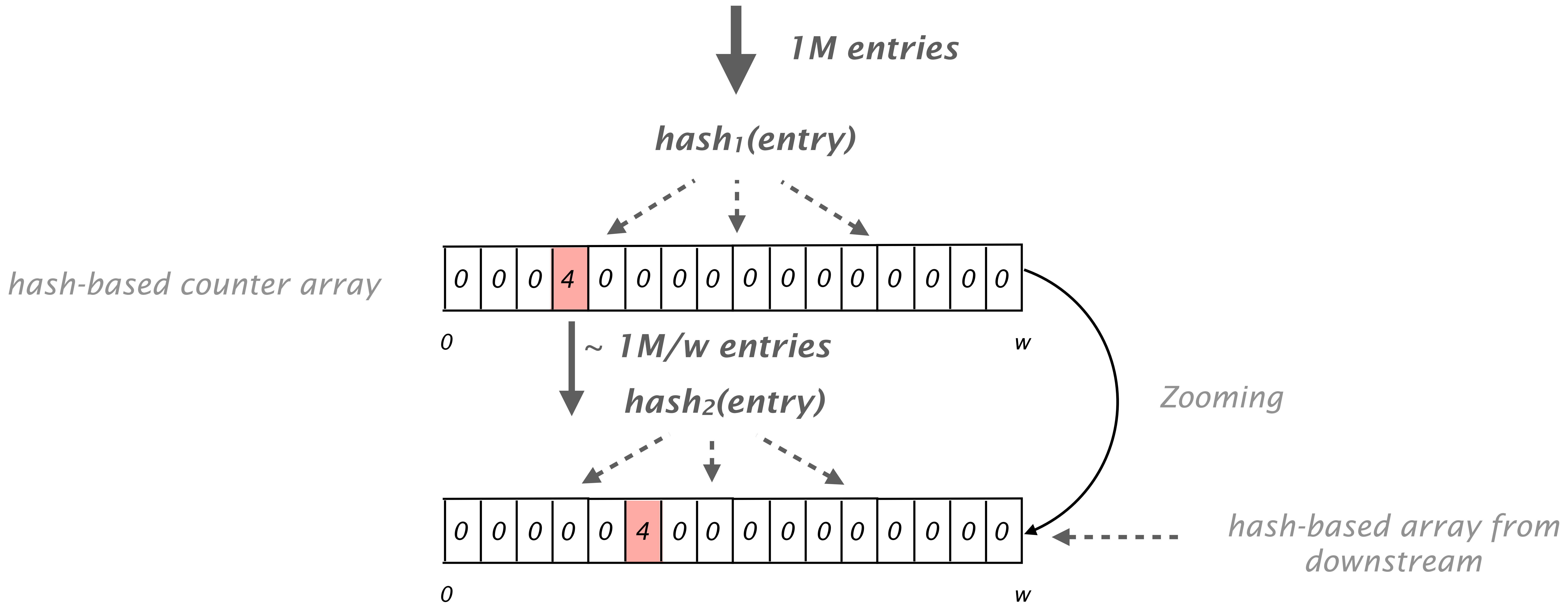
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



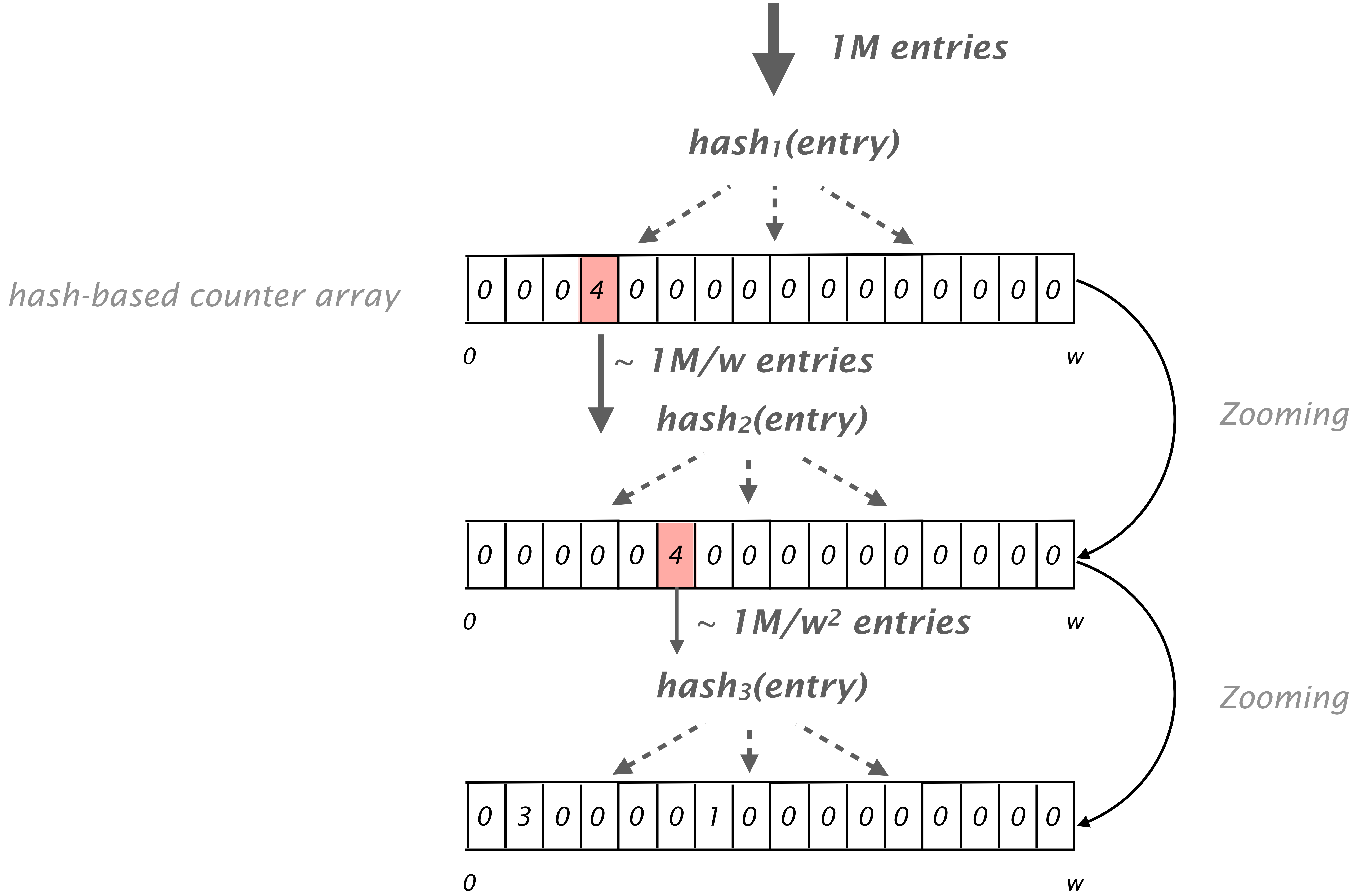
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



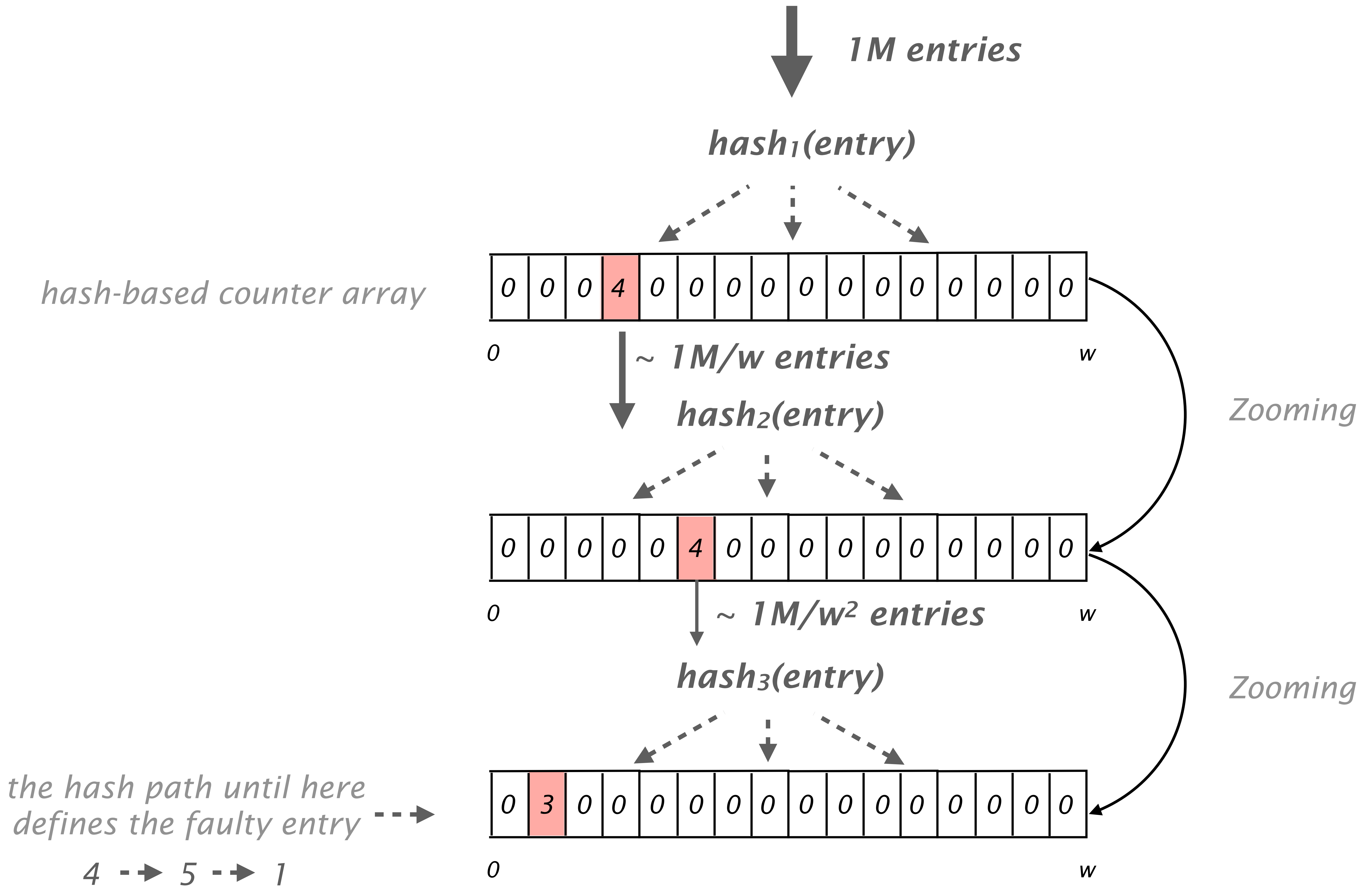
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



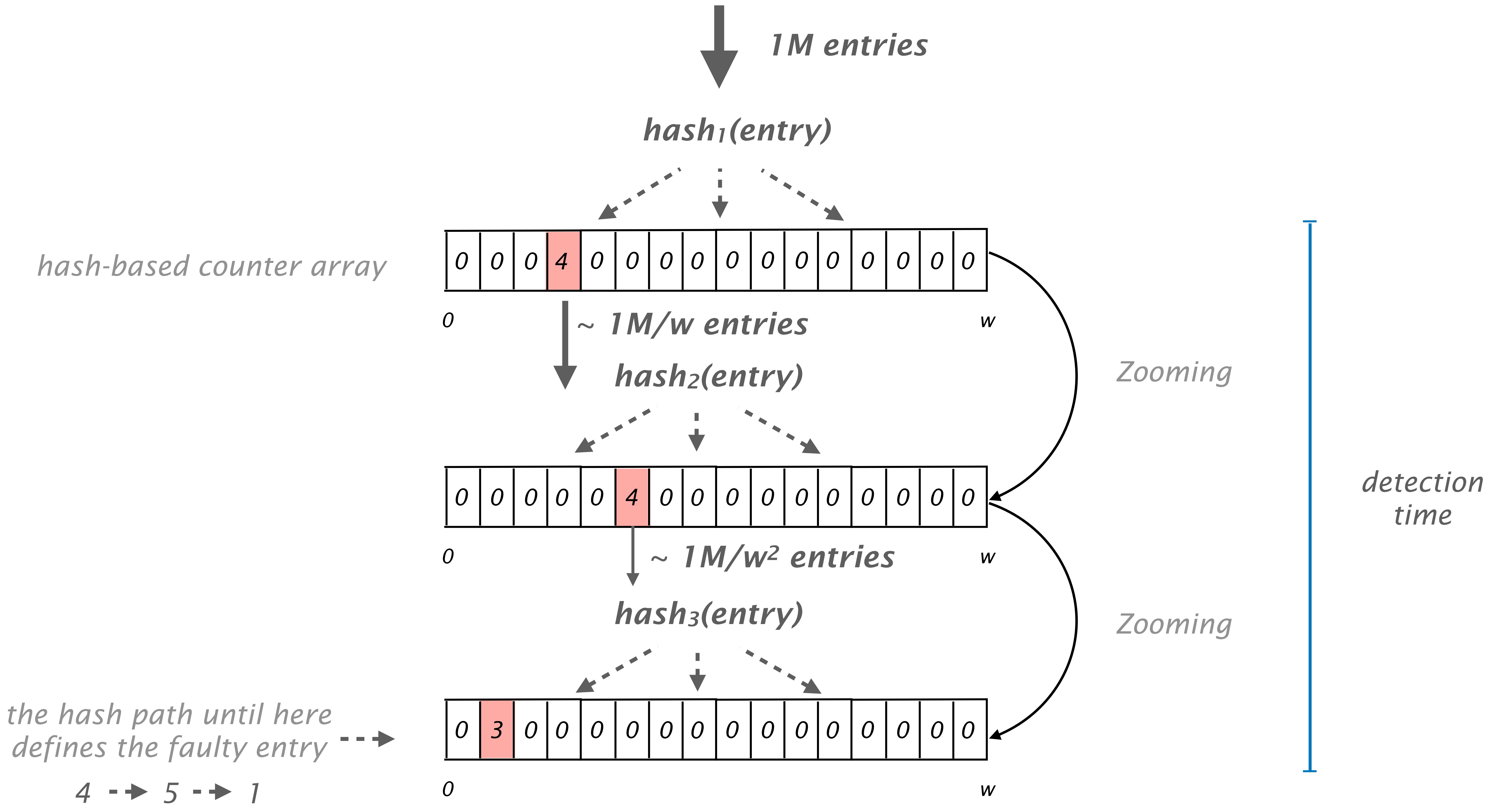
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



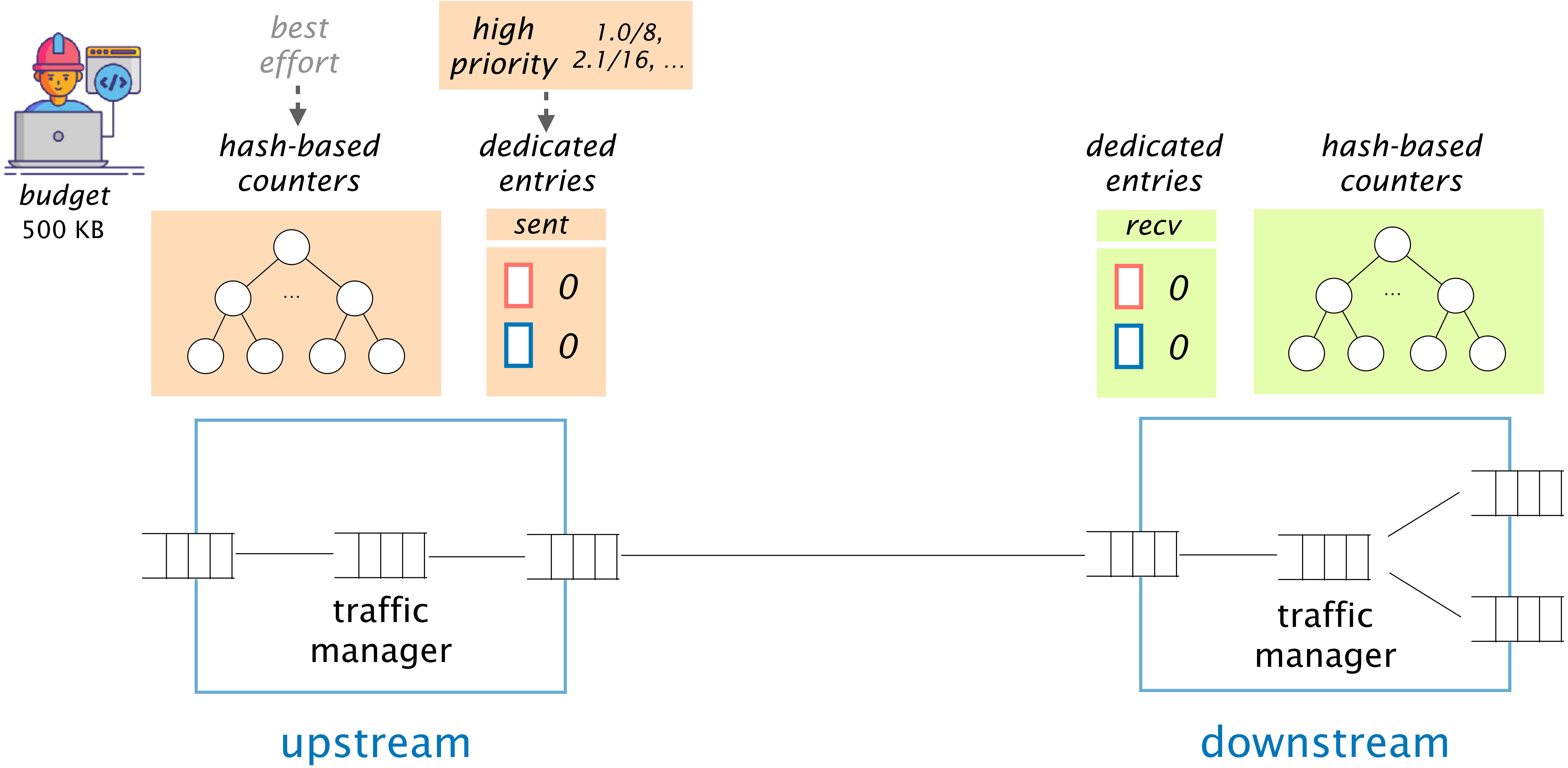
We can leverage the fact that *gray* failures tend to be ***sparse*** and ***aggregate*** multiple traffic entries into the ***same counter***



Hash-based counters allow ***FANcY*** to scale at the cost of reducing the ***detection speed*** and ***accuracy***



FANcY can combine ***dedicated counter entries*** with the ***hash-based counters***



We evaluated ***FANcY*** accuracy and speed

- Software simulations: ~9000 lines of C++ code extending ns-3

  #1 How does FANcY perform depending on the gray failure type and the volume of traffic affected

- Hardware implementation: ~3000 lines of P4 code

  #2 Does FANcY work on Intel Tofino programmable switches?

We evaluated ***FANcY*** accuracy and speed

Multi-entry failures, uniform random drops, eval with CAIDA traces

more in the paper!

#1 How does FANcY perform depending on the gray failure type and the volume of traffic affected?

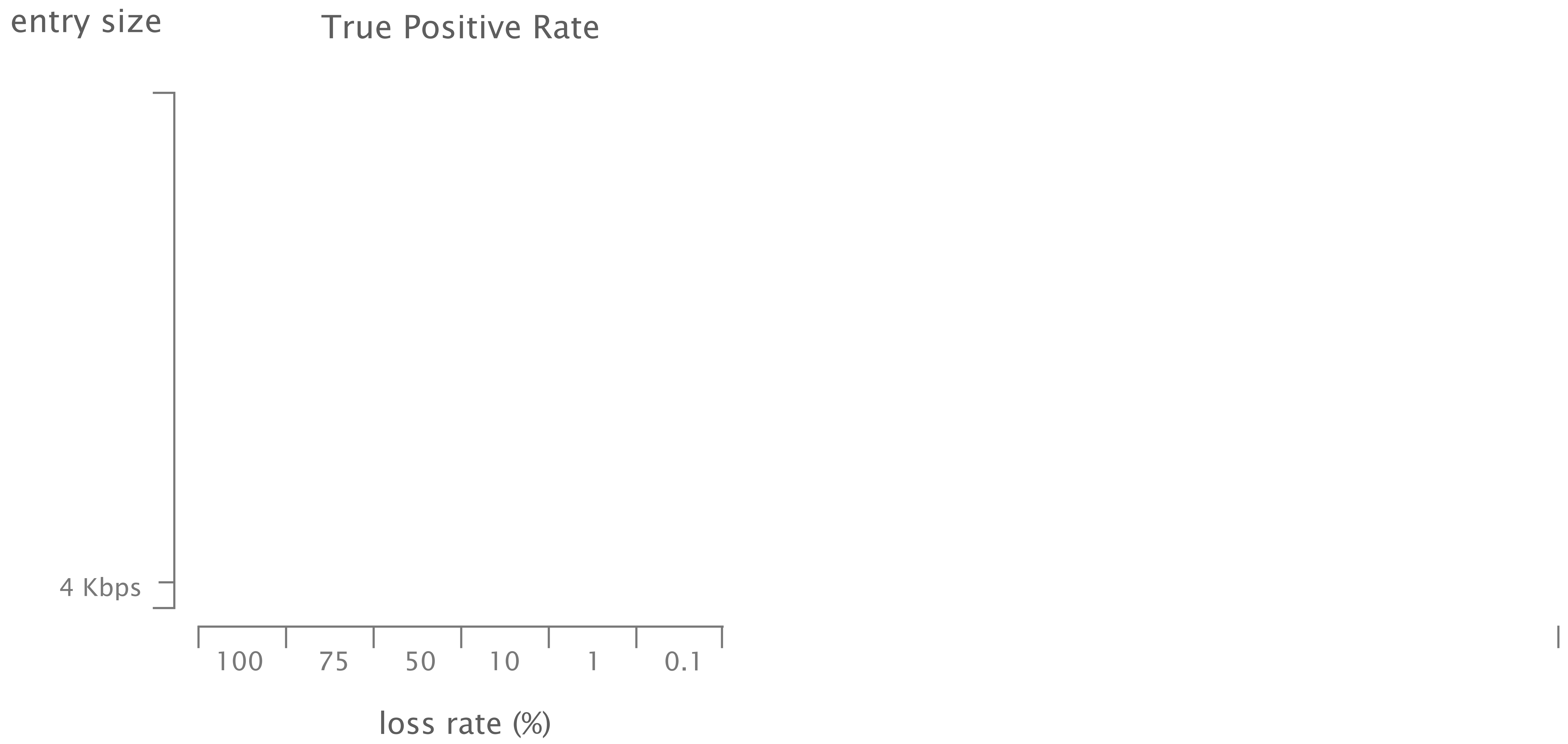
Methodology

We evaluate ***dedicated*** and ***hash-based*** counters on ***single-entry*** gray failures

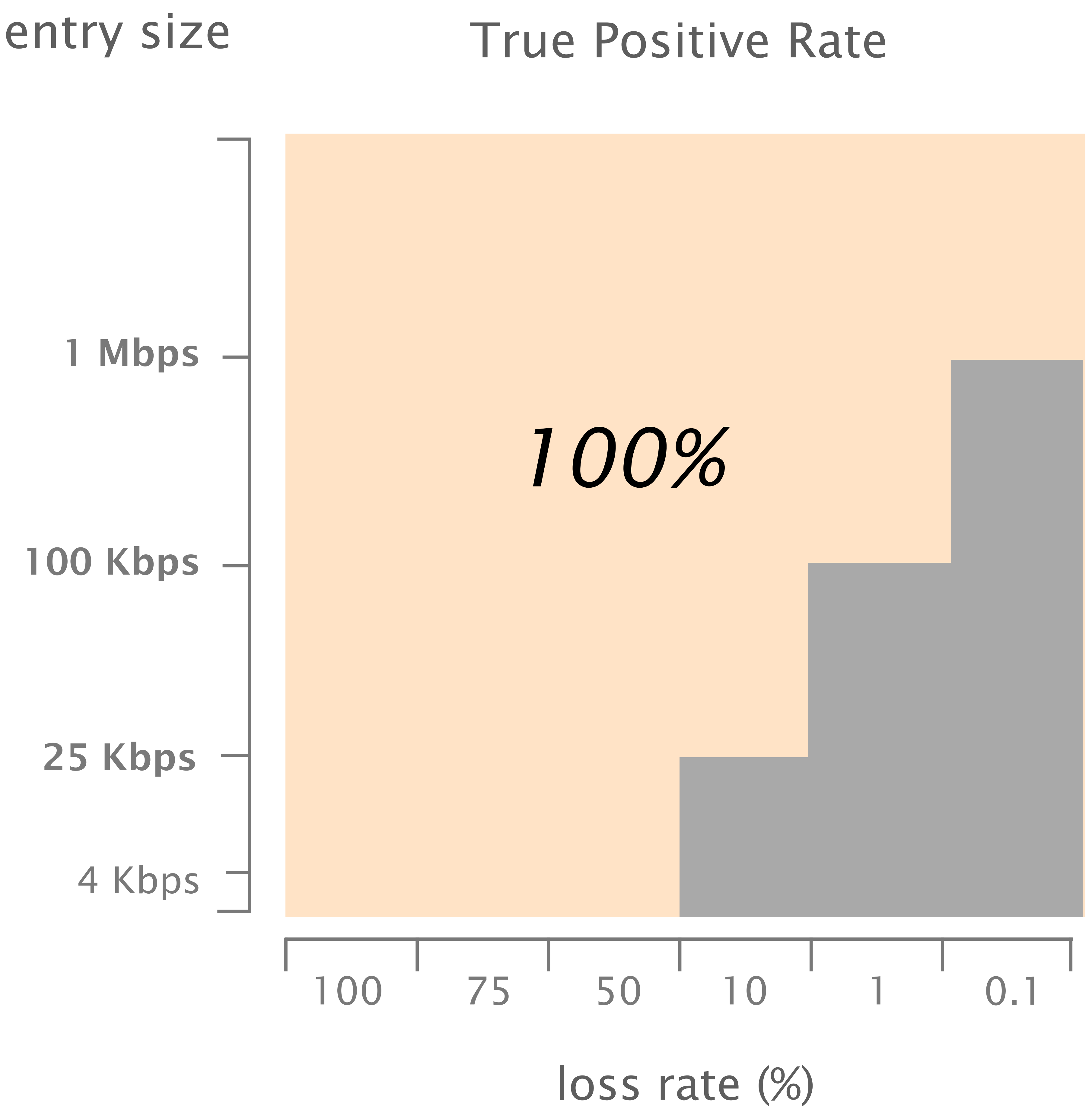
We set the inter-switch delay to ***10 ms***

We run each experiment for ***30 seconds***

FANcY's ***hash-based counters*** performance with ***3 layers*** and ***counting time of 200ms***

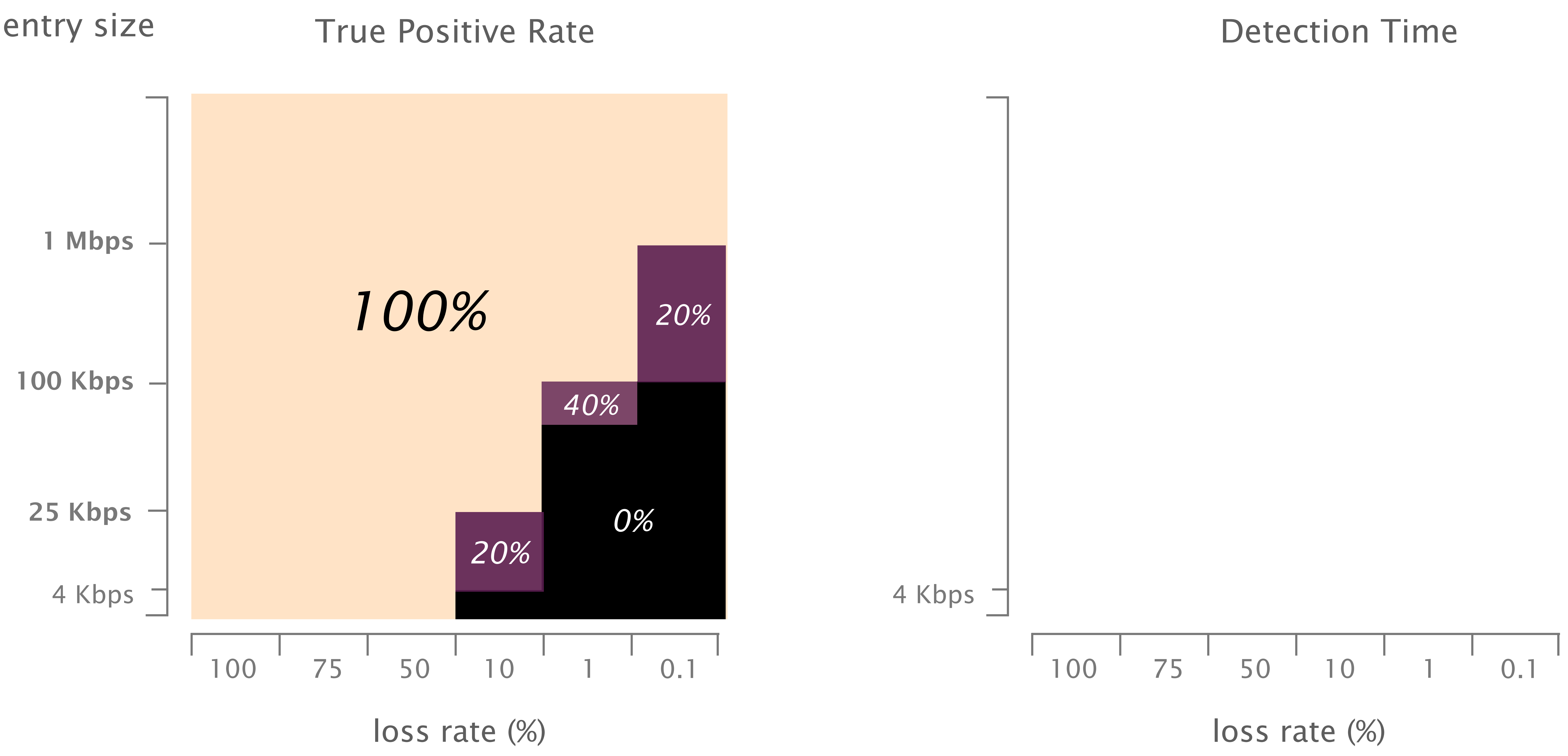


FANcY's ***hash-based counters*** performance with ***3 layers*** and ***counting time of 200ms***

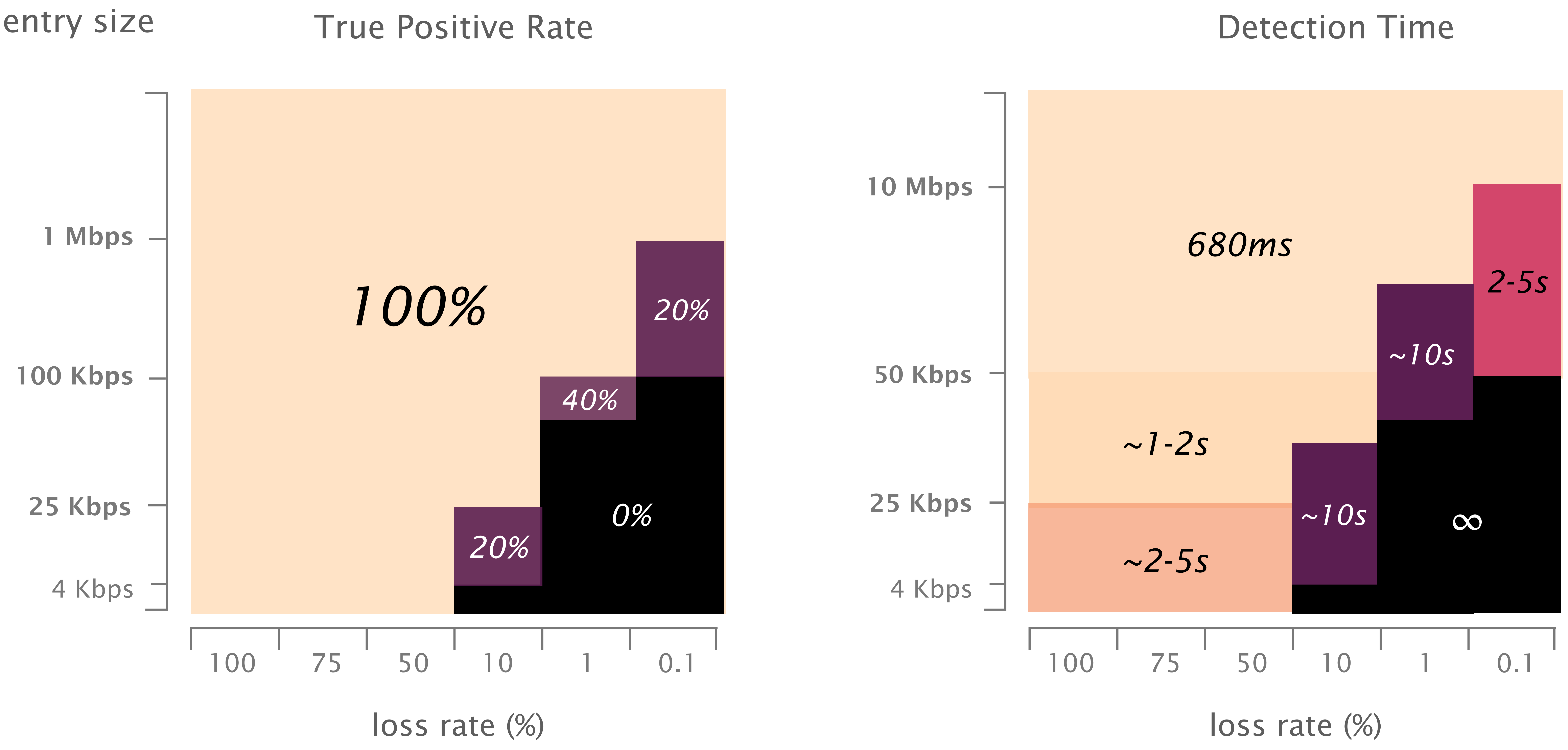


FANcY's ***hash-based counters*** performance with ***3 layers*** and ***counting time of 200ms***

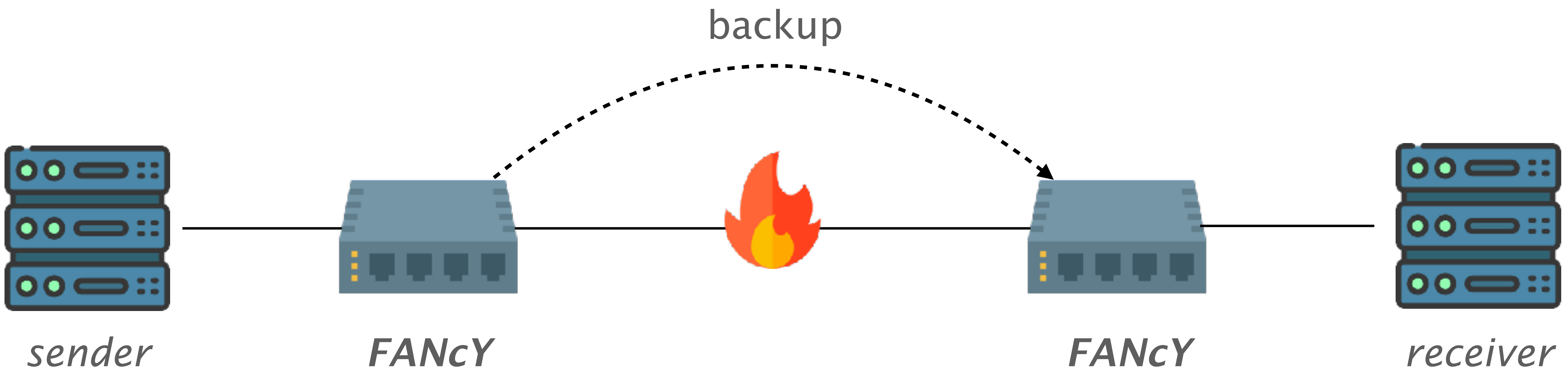
FANcY's ***hash-based counters*** performance with ***3 layers*** and ***counting time of 200ms***



FANcY's ***hash-based counters*** performance with ***3 layers*** and ***counting time of 200ms***



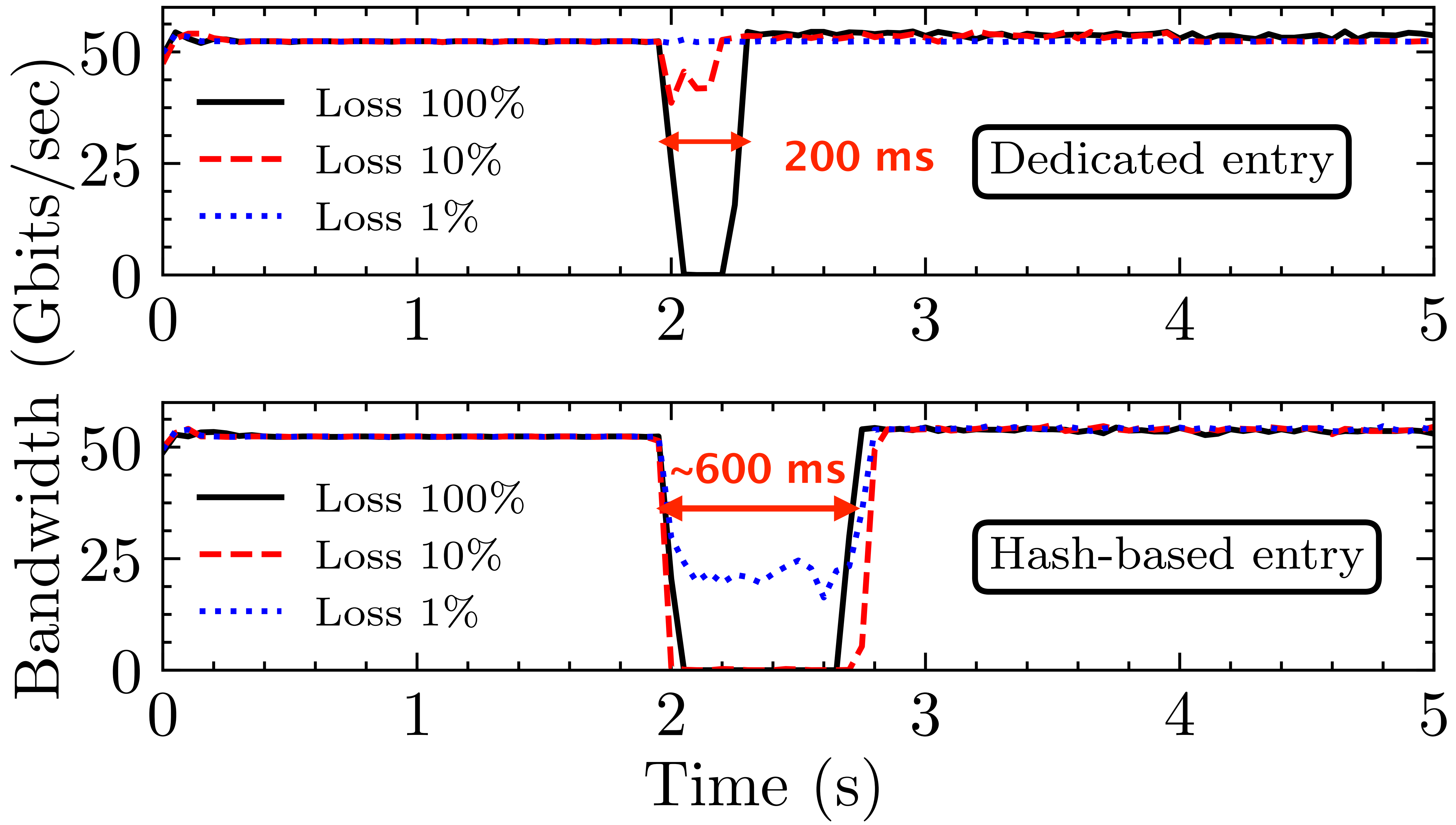
#2 Does ***FANcY*** work on ***Intel Tofino*** programmable switches?



Dedicated counters are exchanged every ***200 ms***

Hash-based counters have a depth of ***3*** and are zoomed every ***200 ms***

Dedicated counters can detect *gray* failures after the first counting session whereas ***hash-based counters*** need to zoom three times



FANcY: FAst In-Network *Gray* Failure Detection for ISPs

detects gray failures by doing counter comparisons
reliable counter synch protocol directly in data plane

scales by using two types of counting data structures
uses dedicated counters and hash-based counters

runs in today's hardware
implemented and tested on Intel Tofino Switches

FAst In-Network *Gray* Failure Detection for ISPs

github.com/nsg-ethz/FANcY

Thank you!

SIGCOMM'22
August, 25 2022

Edgar Costa Molero[1],
Stefano Vissicchio[2],
Laurent Vanbever[1]

[1] ETH zürich

[2] UCL