ETH zürich

Network Security Group
Department of Computer Science

ALBUS: a Probabilistic Monitoring Algorithm to Counter Burst-Flood Attacks

Simon Scherrer, Adrian Perrig
ETH Zürich

Jo Vliegen, Arish Sateesan, Nele Mentens*
*KU Leuven, *Leiden University*

Hsu-Chun Hsiao
National Taiwan University

SRDS 2023, Marrakech

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon	Ripple	Jaqen	COLIBRI	ACC-Turbo
(Zhang et al. 2020)	(Xing et al. 2021)	(Liu et al. 2021)	(Giuliari et al. 2021)	(Alcoz et al. 2022)

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon	Ripple	Jaqen	COLIBRI	ACC-Turbo
(Zhang et al. 2020)	(Xing et al. 2021)	(Liu et al. 2021)	(Giuliari et al. 2021)	(Alcoz et al. 2022)

DDoS Defense System

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon	Ripple	Jaqen	COLIBRI	ACC-Turbo
(Zhang et al. 2020)	(Xing et al. 2021)	(Liu et al. 2021)	(Giuliari et al. 2021)	(Alcoz et al. 2022)

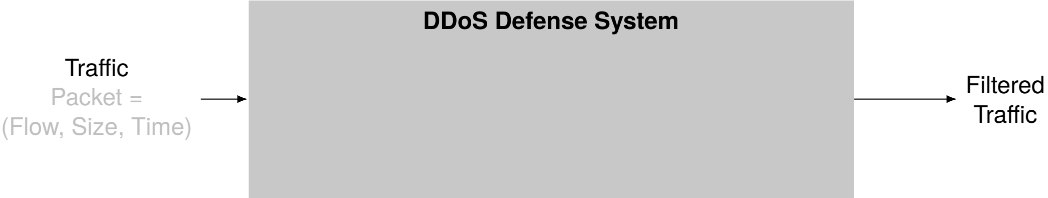
Traffic
Packet =
(Flow, Size, Time)
→

DDoS Defense System

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon	Ripple	Jaqen	COLIBRI	ACC-Turbo
(Zhang et al. 2020)	(Xing et al. 2021)	(Liu et al. 2021)	(Giuliari et al. 2021)	(Alcoz et al. 2022)



How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon	Ripple	Jaqen	COLIBRI	ACC-Turbo
(Zhang et al. 2020)	(Xing et al. 2021)	(Liu et al. 2021)	(Giuliari et al. 2021)	(Alcoz et al. 2022)

Traffic
Packet =
(Flow, Size, Time)

DDoS Defense System

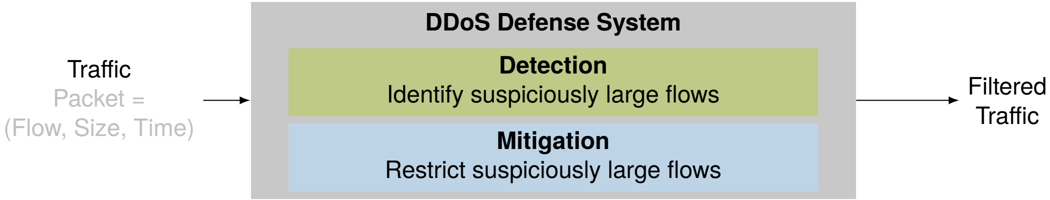
Detection
Identify suspiciously large flows

Filtered
Traffic

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon (Zhang et al. 2020) | Ripple (Xing et al. 2021) | Jaqen (Liu et al. 2021) | COLIBRI (Giuliari et al. 2021) | ACC-Turbo (Alcoz et al. 2022)



How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon (Zhang et al. 2020) | Ripple (Xing et al. 2021) | Jaqen (Liu et al. 2021) | COLIBRI (Giuliari et al. 2021) | ACC-Turbo (Alcoz et al. 2022)

Traffic
Packet =
(Flow, Size, Time)

DDoS Defense System

Detection
Identify suspiciously large flows

Mitigation
Restrict suspiciously large flows

Filtered
Traffic

Configuration,
Programming

Network operator

How Do DDoS Defense Systems Work?

Goal: Guarantee availability of network resources under malicious traffic from many sources

Poseidon (Zhang et al. 2020)

Ripple (Xing et al. 2021)

Jaqen (Liu et al. 2021)

COLIBRI (Giuliari et al. 2021)

ACC-Turbo (Alcoz et al. 2022)

Traffic

Packet = (Flow, Size, Time)

DDoS Defense System

Detection
Identify suspiciously large flows

Mitigation
Restrict suspiciously large flows

Filtered Traffic

Configuration, Programming

Network operator

How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

+ Limited memory
+ Efficient processing
- Limited accuracy

How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

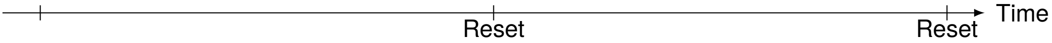
+ Limited memory + Efficient processing - Limited accuracy

Time

How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

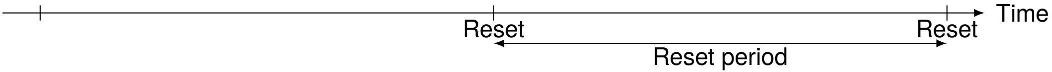
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

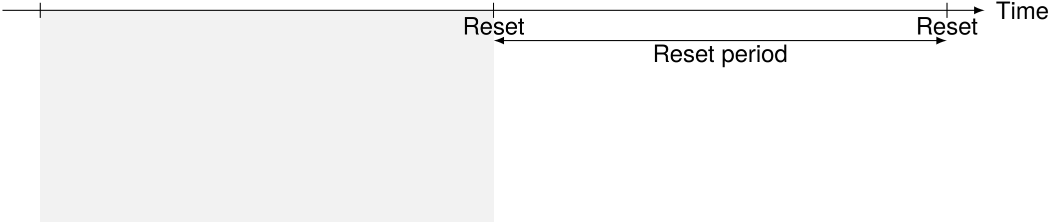
\+ Limited memory   + Efficient processing   - Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

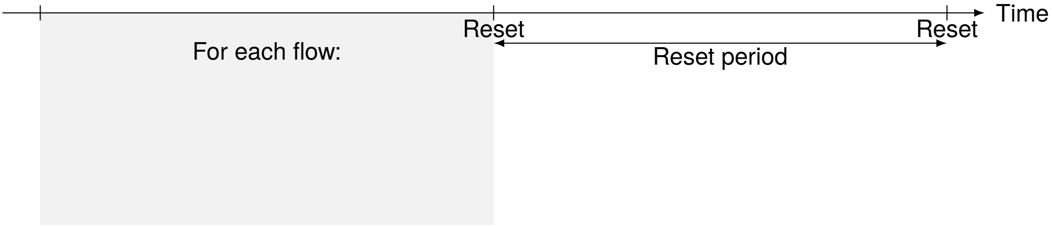
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

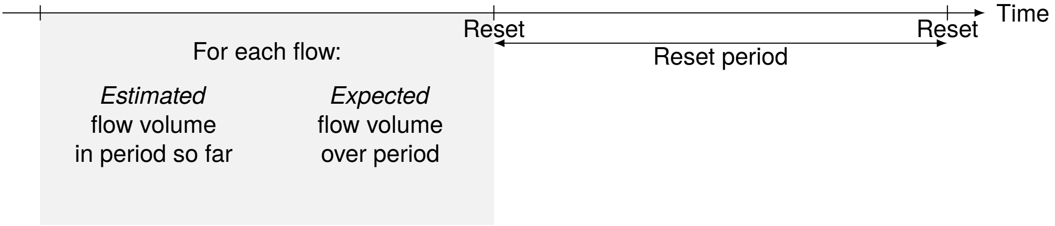
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

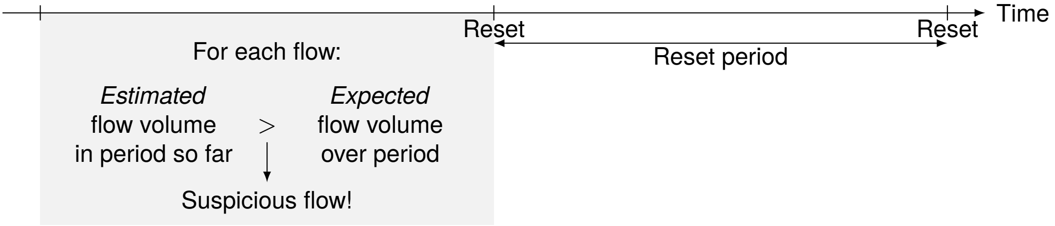
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

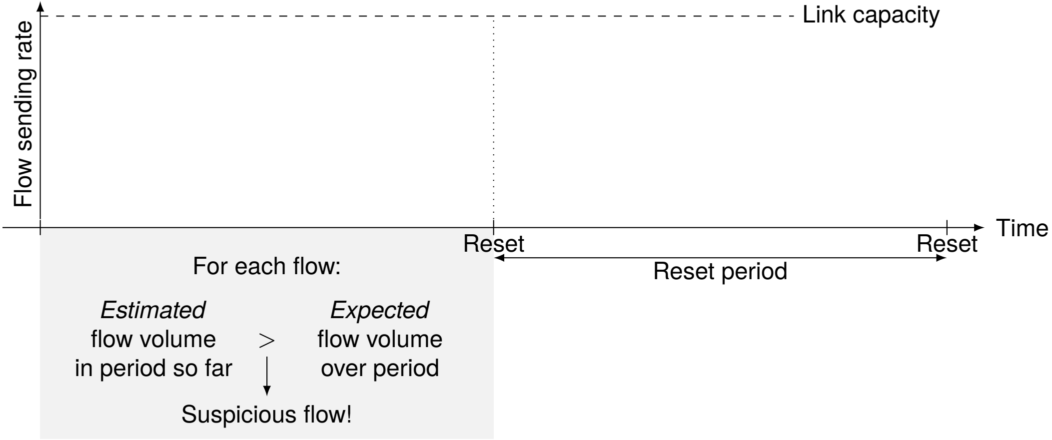
+ Limited memory + Efficient processing - Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

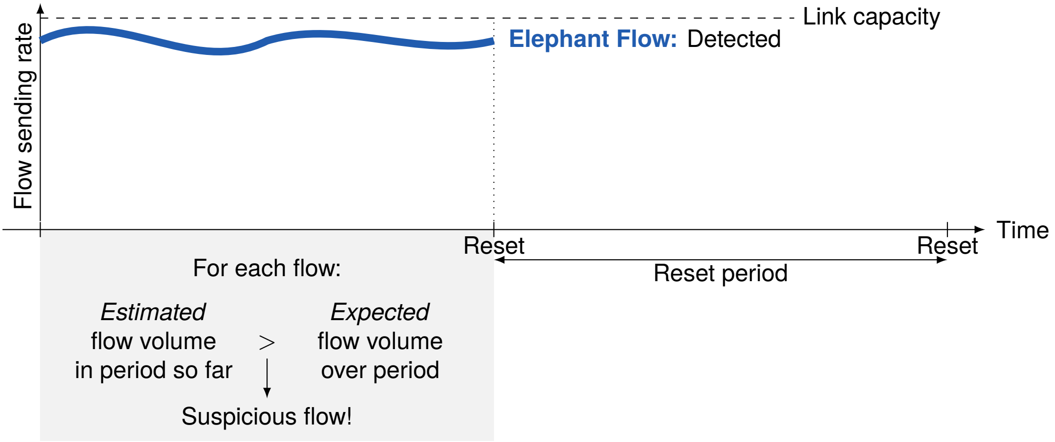
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

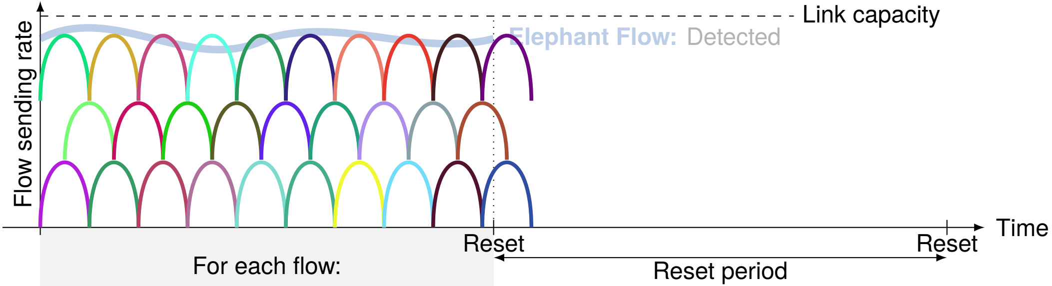
+ Limited memory
+ Efficient processing
- Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

+ Limited memory + Efficient processing - Limited accuracy



For each flow:

Estimated flow volume in period so far > *Expected* flow volume over period

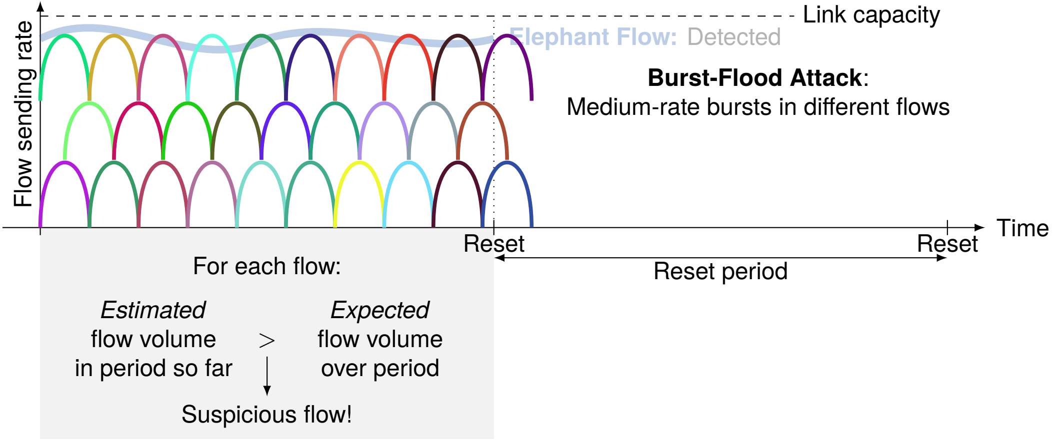
↓

Suspicious flow!

How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

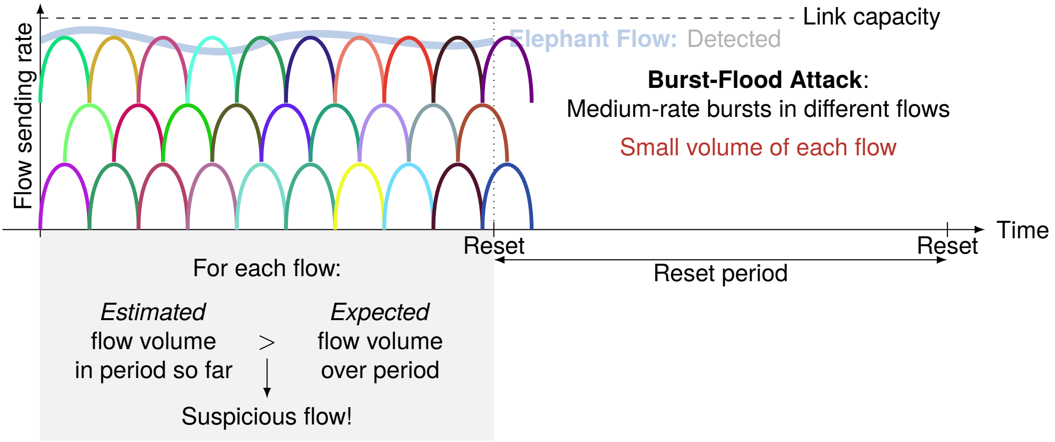
\+ Limited memory   + Efficient processing   - Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

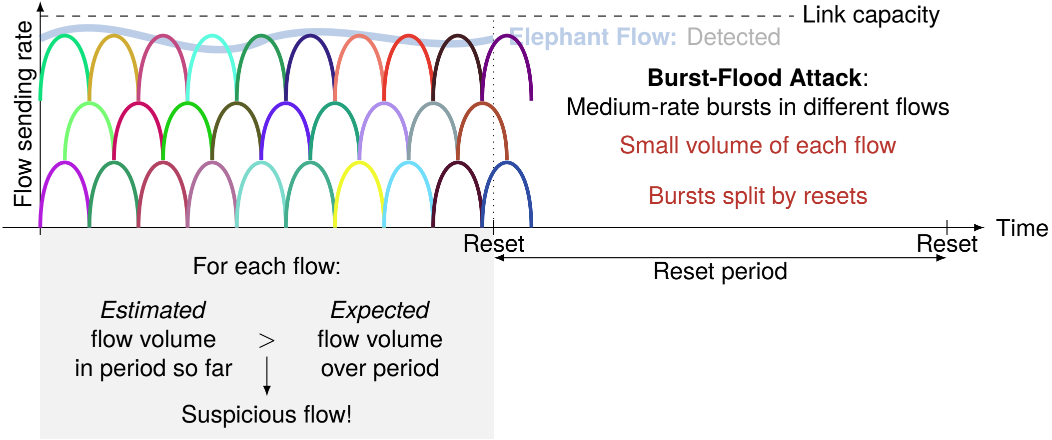
+ Limited memory   + Efficient processing   - Limited accuracy



How Do DDoS Defense Systems Detect Suspicious Flows?

DDoS defense systems mostly use the algorithms **CountMin-Sketch** and **Count-Sketch** for detection

+ Limited memory + Efficient processing - Limited accuracy



How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

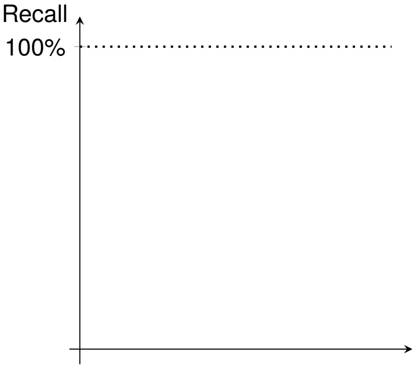
$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}} \qquad \text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$

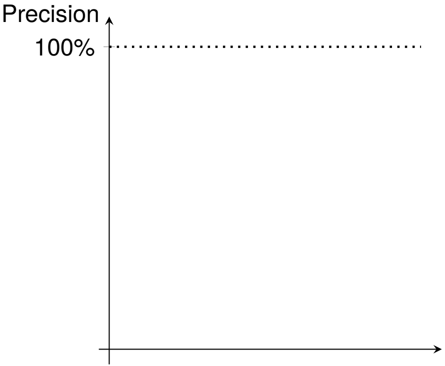
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



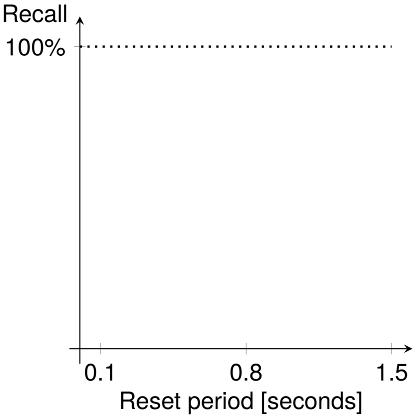


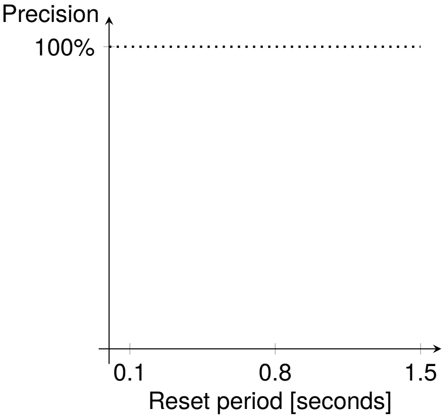
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



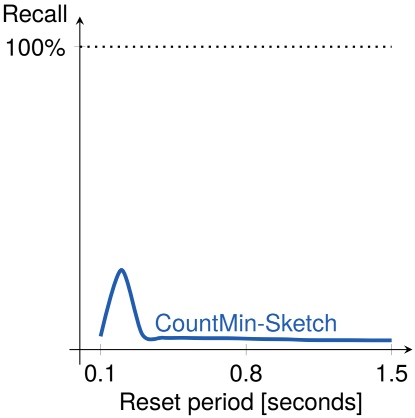


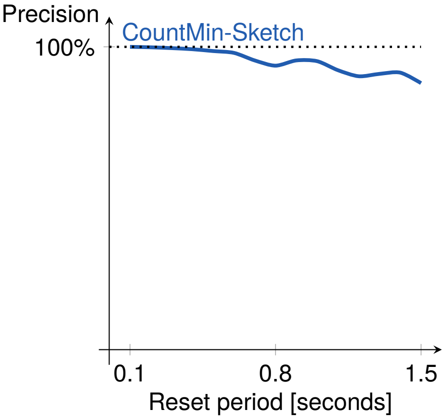
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$

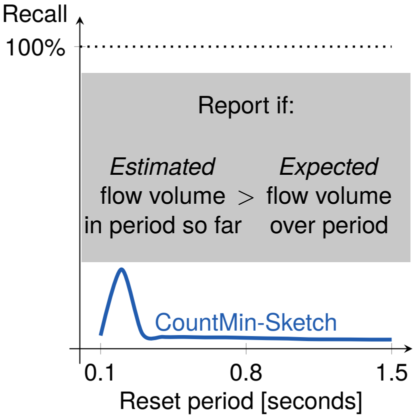




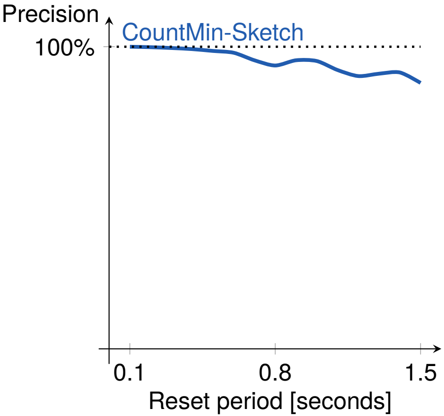
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$



$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$

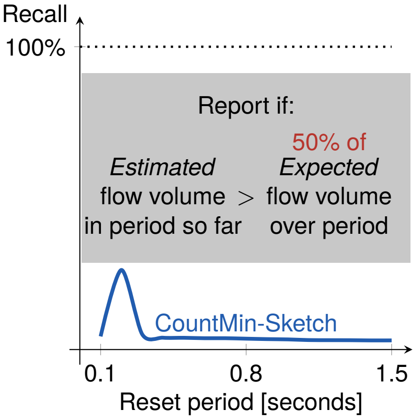


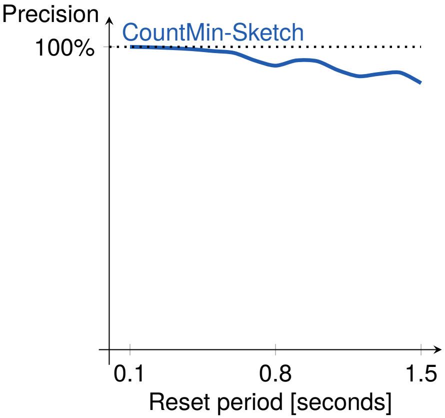
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



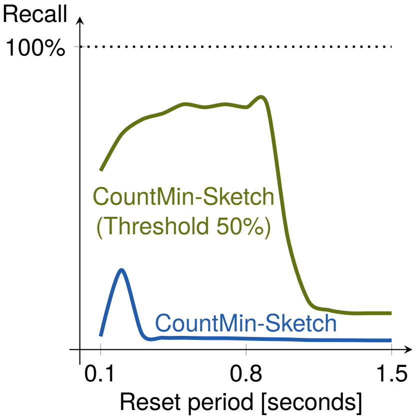


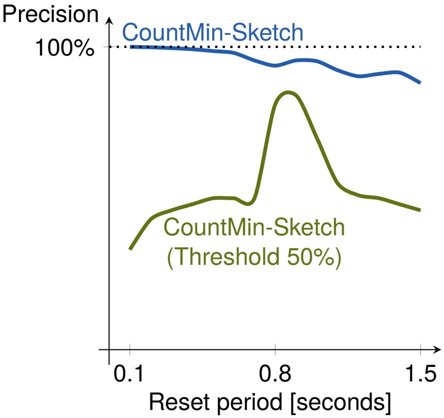
How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$

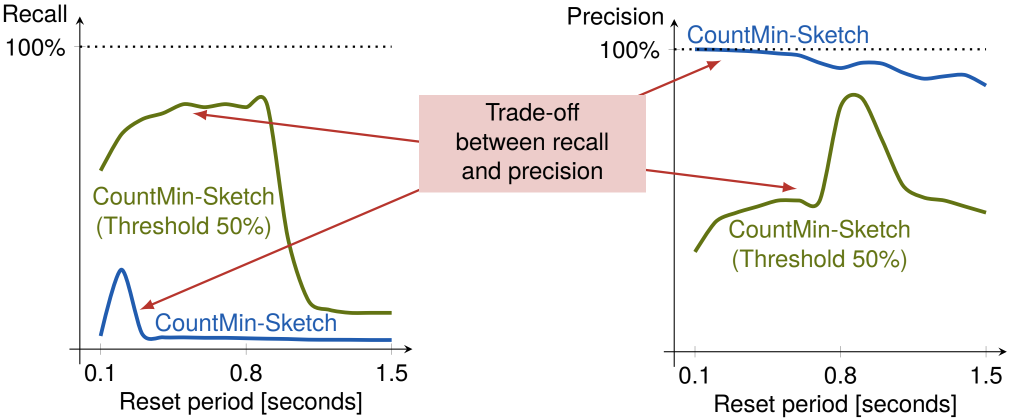




How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

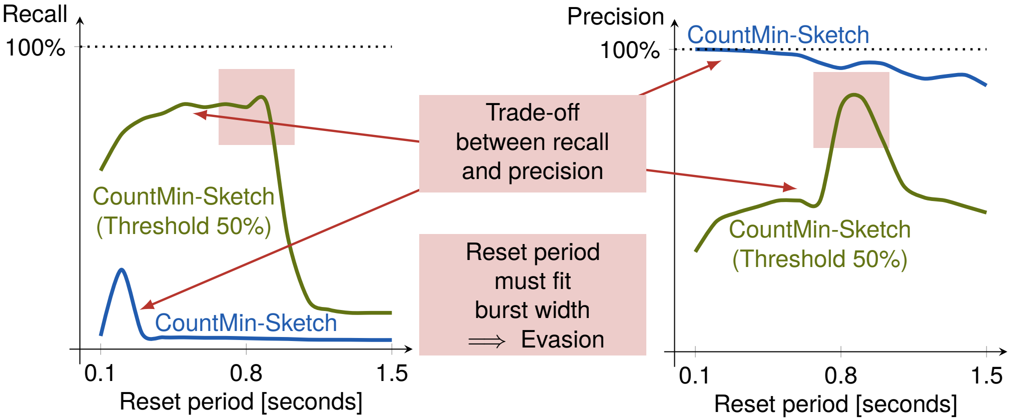
$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}} \qquad \text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



How Do Sketch Algorithms Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}} \qquad \text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

Time-window flexibility

How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

Time-window flexibility

Memory efficiency

How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

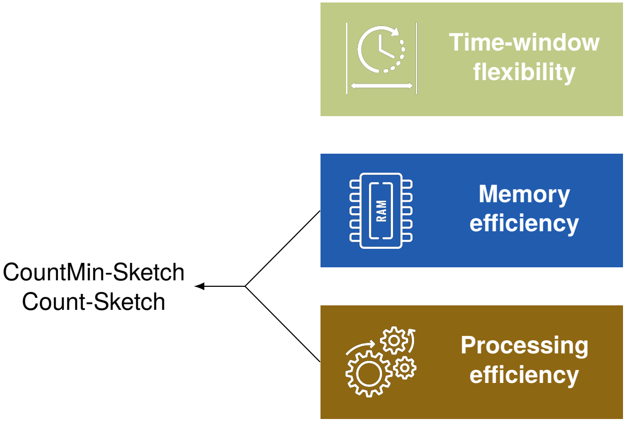
Time-window flexibility

Memory efficiency

Processing efficiency

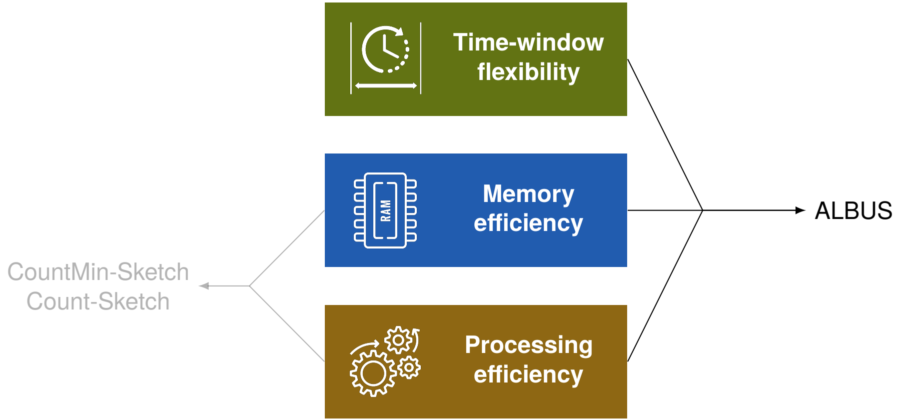
How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:



How Can We Better Detect Malicious Bursts?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:



How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

How Does ALBUS Work?

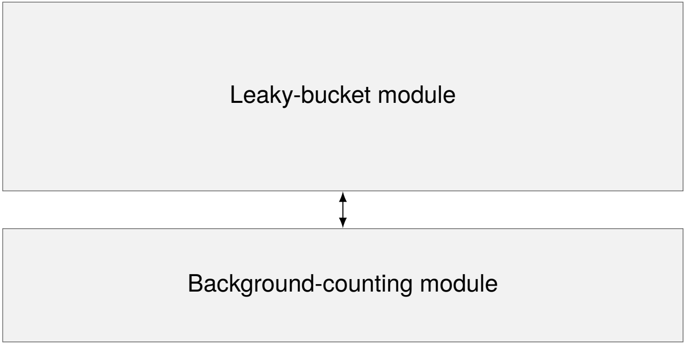
ALBUS: Adaptive Leaky-Bucket Undulation Sensor

Flows: f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 ⋯ $f_{1475286}$

How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

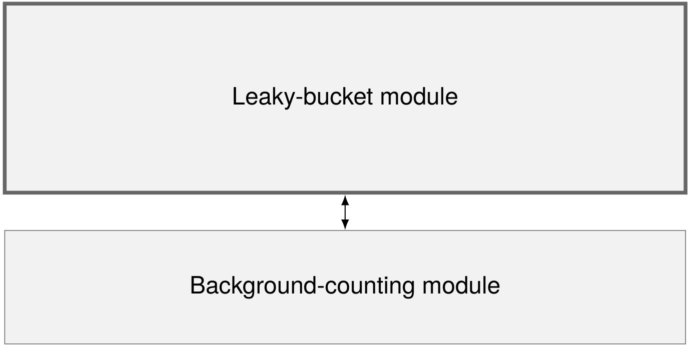
Flows: f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 $\cdots$ $f_{1475286}$



How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

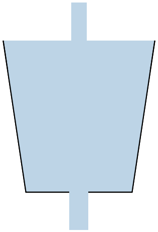
Flows: f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 $\cdots$ $f_{1475286}$



How Does the Leaky-Bucket Algorithm Work?

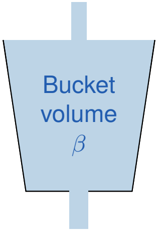
How Does the Leaky-Bucket Algorithm Work?

General idea:



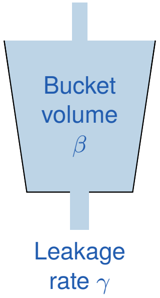
How Does the Leaky-Bucket Algorithm Work?

General idea:



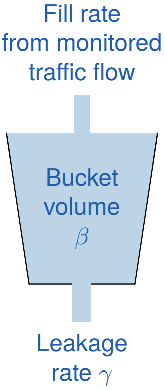
How Does the Leaky-Bucket Algorithm Work?

General idea:



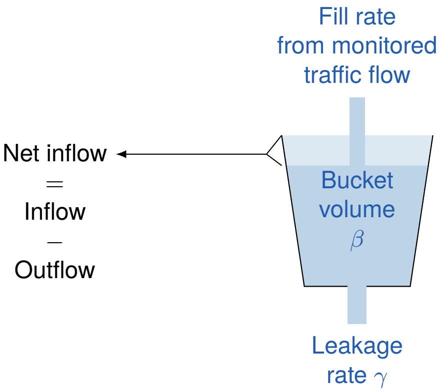
How Does the Leaky-Bucket Algorithm Work?

General idea:



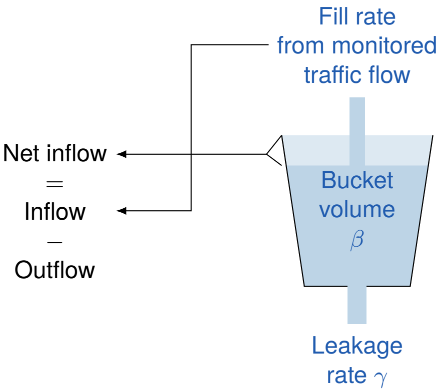
How Does the Leaky-Bucket Algorithm Work?

General idea:



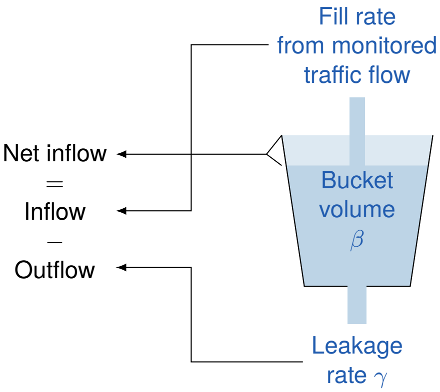
How Does the Leaky-Bucket Algorithm Work?

General idea:



How Does the Leaky-Bucket Algorithm Work?

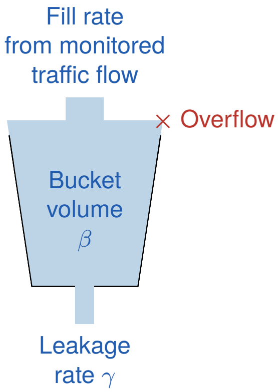
General idea:



How Does the Leaky-Bucket Algorithm Work?

General idea:

Net inflow
$=$
Inflow
$-$
Outflow



What Properties Are Offered by the Leaky-Bucket Algorithm?

What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$
during a time window with arbitrary width w

What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \ \beta$
during a time window with arbitrary width w

γ $=$ Leakage rate $=$ Flow base rate

What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \boxed{\beta}$ during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

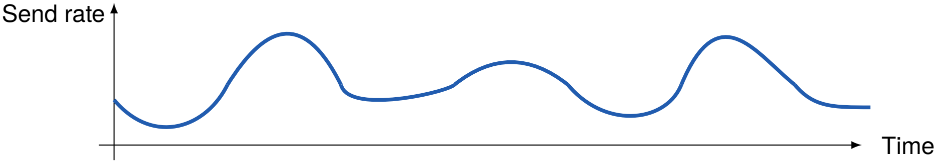
β = Bucket volume = Burstiness allowance

What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$
during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance

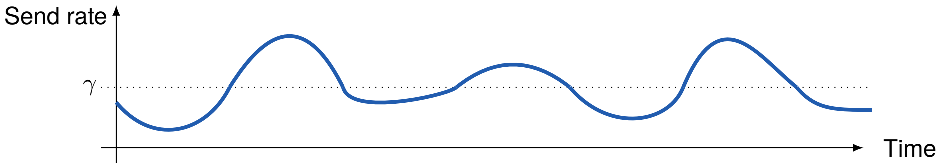


What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$
during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance

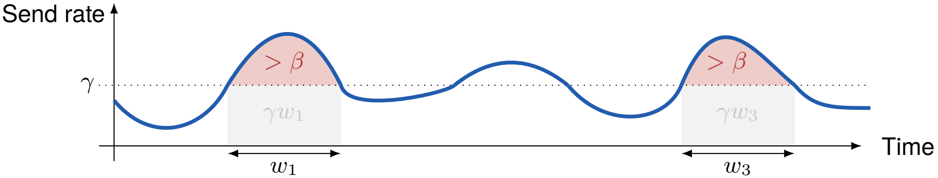


What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$ during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance

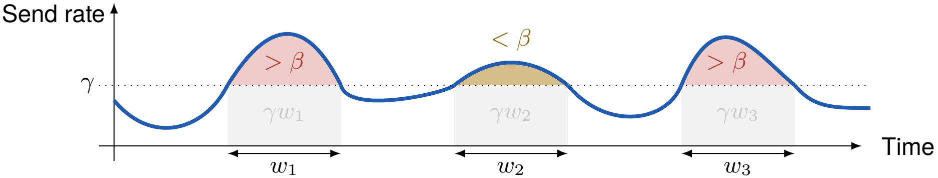


What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$
during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance



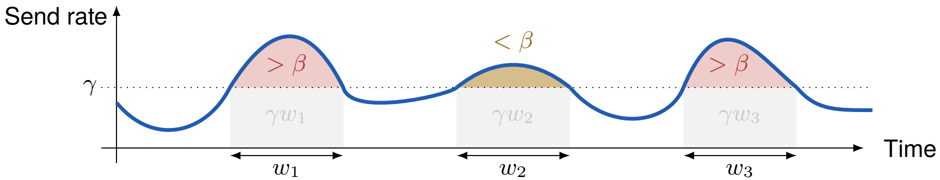
What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$
during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance

Time-window flexibility

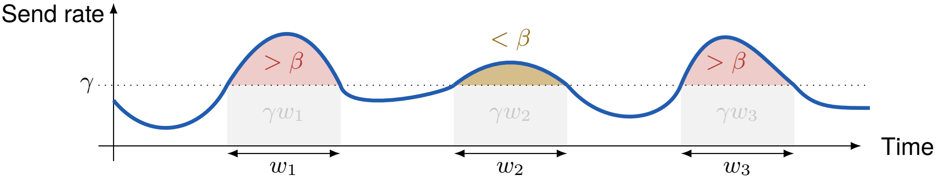


What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$ during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance



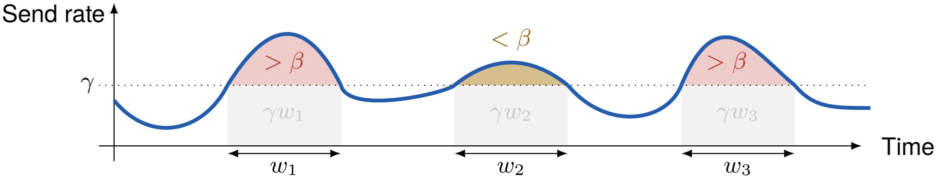
But: A leaky bucket can only monitor a single flow at a time!

What Properties Are Offered by the Leaky-Bucket Algorithm?

A leaky bucket reports a flow that sends more than $\gamma\, w + \beta$ during a time window with arbitrary width w

γ = Leakage rate = Flow base rate

β = Bucket volume = Burstiness allowance

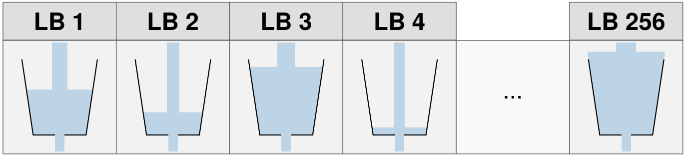


But: A leaky bucket can only monitor a single flow at a time!

How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?

How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?

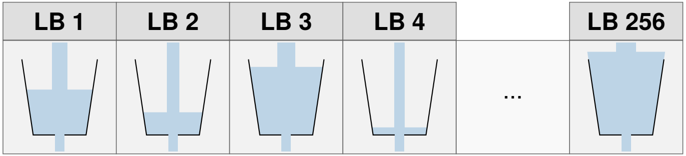
Leaky-bucket counters:



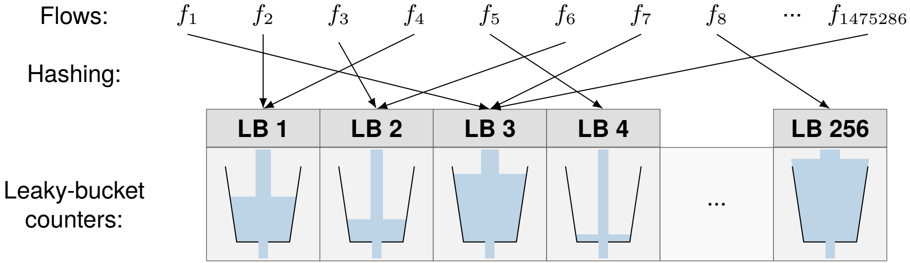
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?

Flows: f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 ... $f_{1475286}$

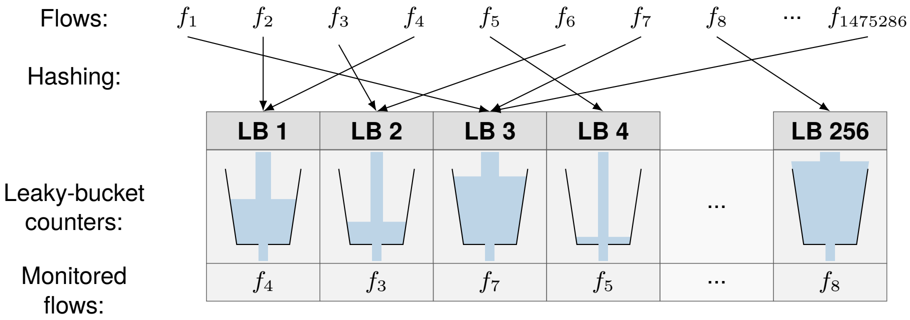
Leaky-bucket counters:



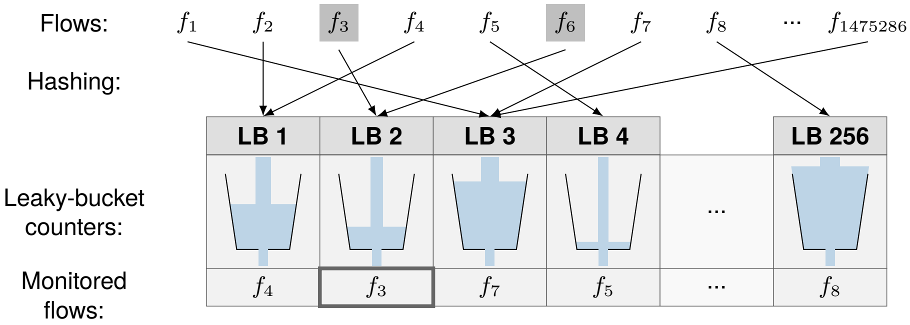
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



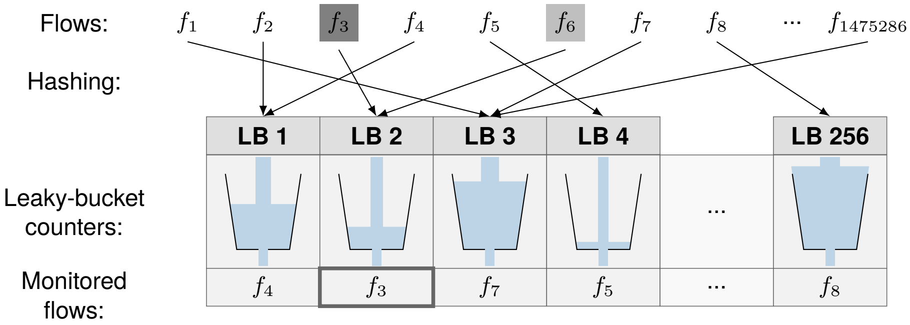
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



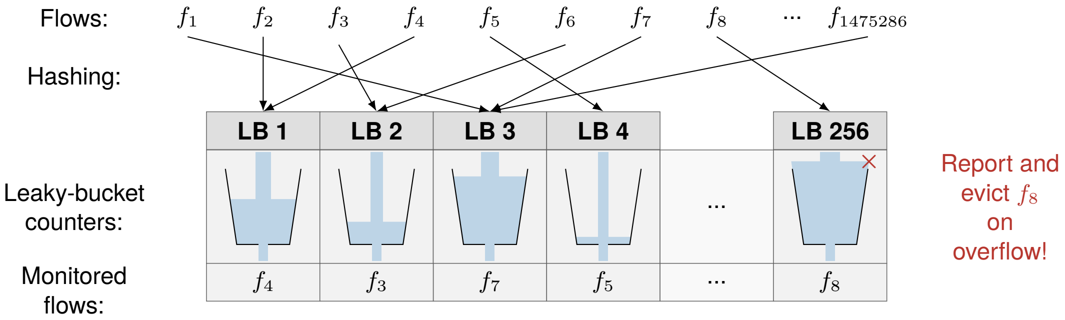
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



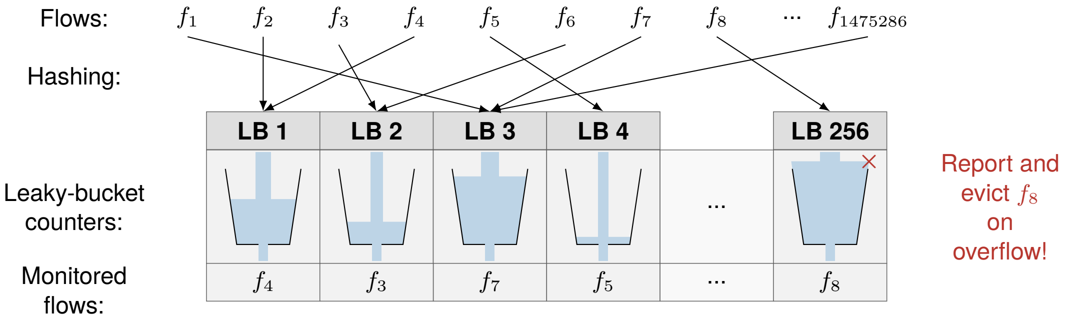
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?

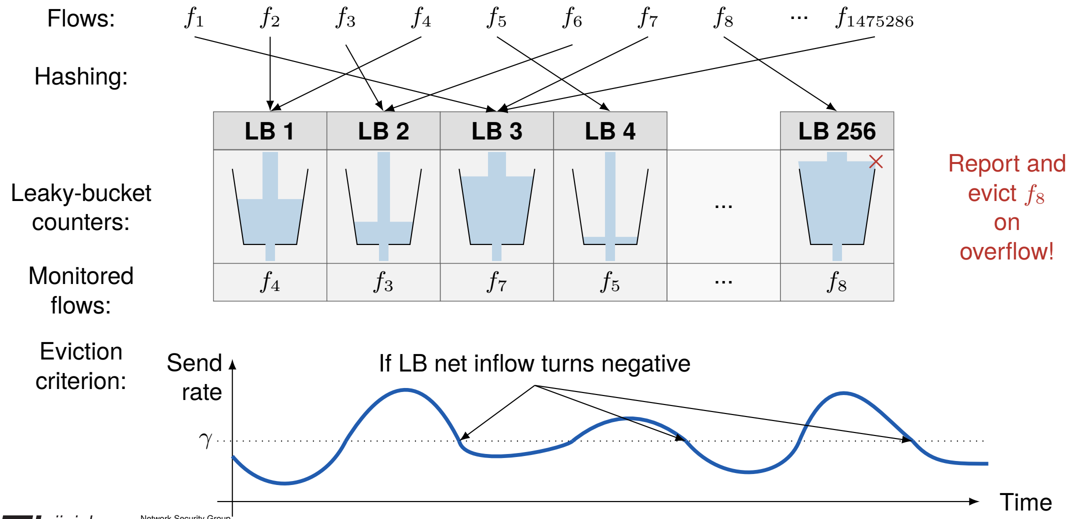


How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



Eviction criterion: If LB net inflow turns negative

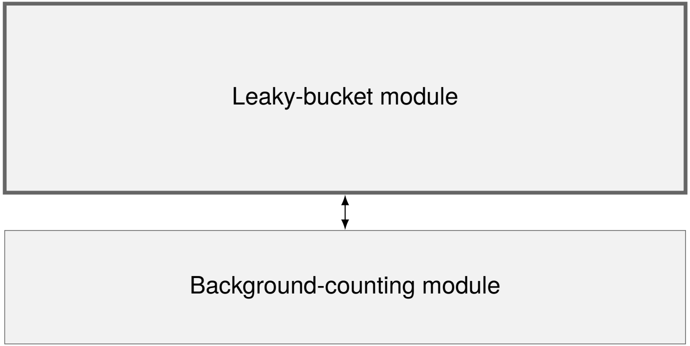
How Can We Apply the LB Algorithm in a Memory-Efficient Fashion?



How Does ALBUS Work?

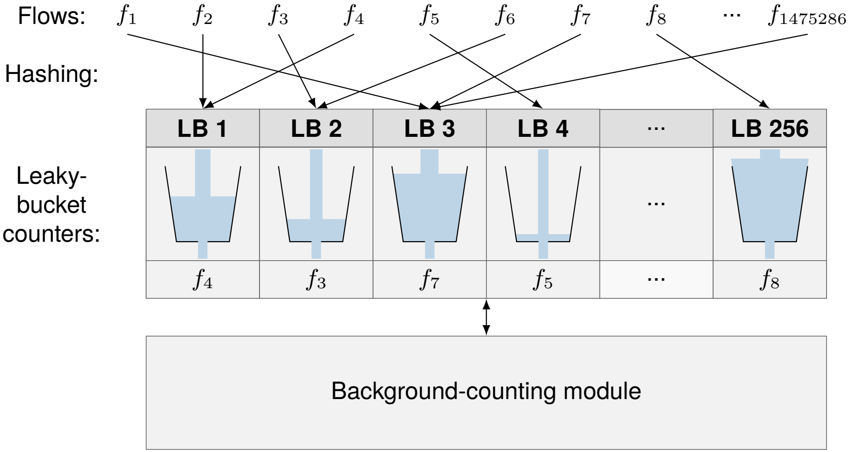
ALBUS: Adaptive Leaky-Bucket Undulation Sensor

Flows: f_1 f_2 f_3 f_4 f_5 f_6 f_7 f_8 $\cdots$ $f_{1475286}$

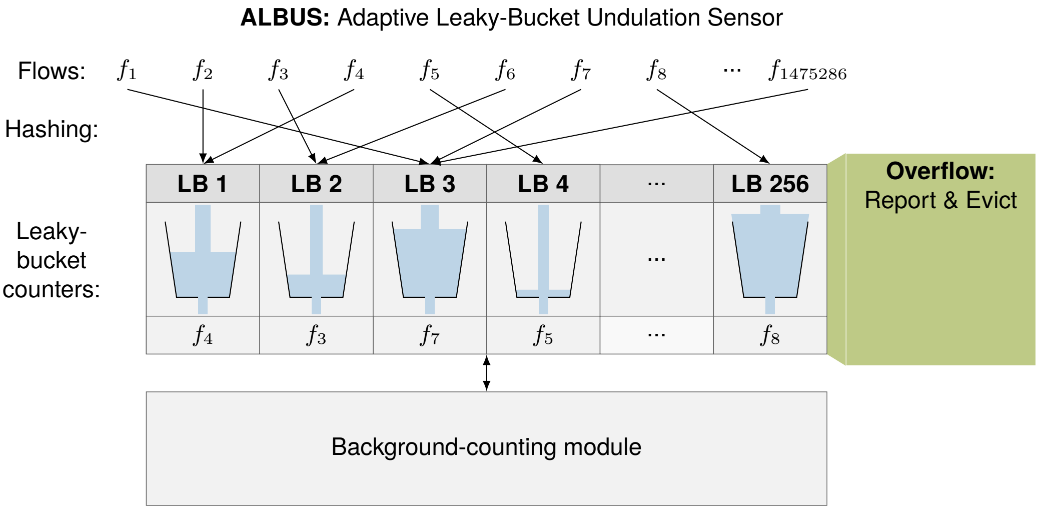


How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

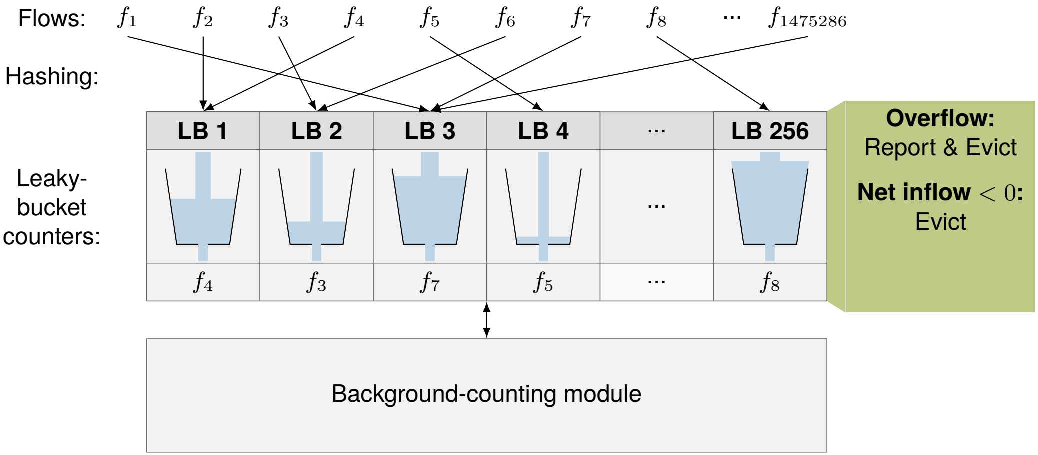


How Does ALBUS Work?

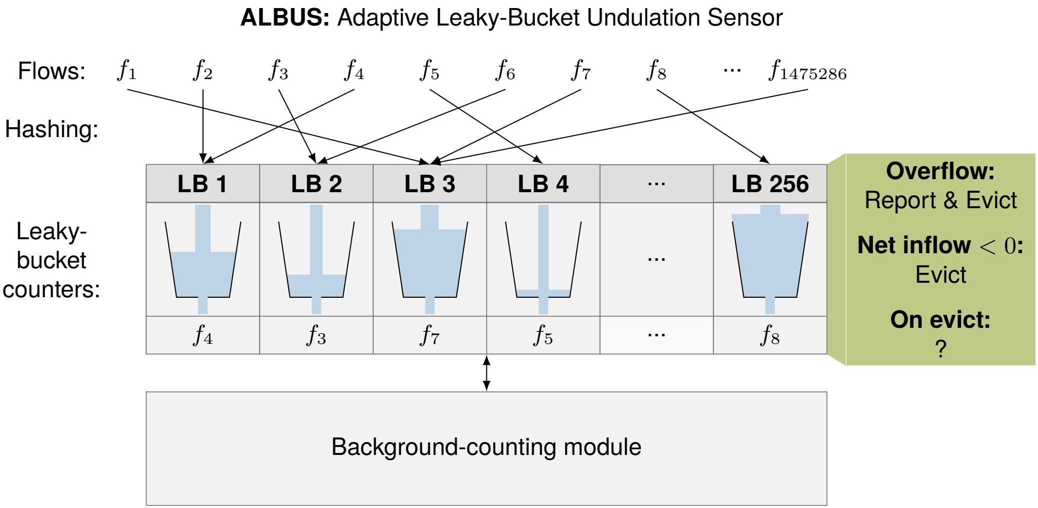


How Does ALBUS Work?

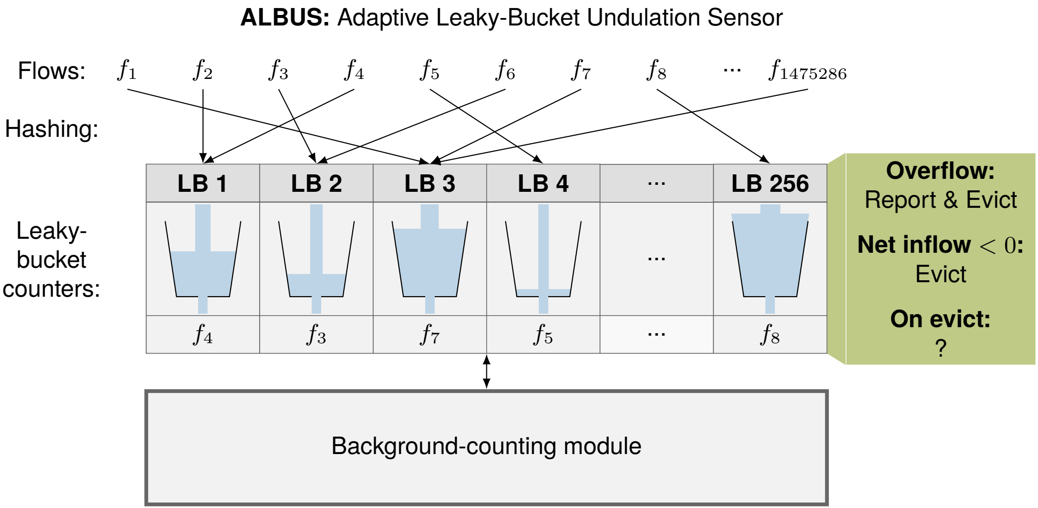
ALBUS: Adaptive Leaky-Bucket Undulation Sensor



How Does ALBUS Work?



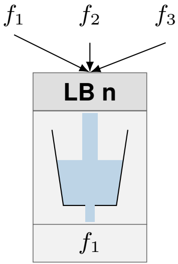
How Does ALBUS Work?



What to Do When a Flow Is Evicted?

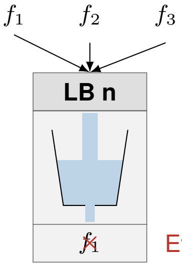
What to Do When a Flow Is Evicted?

Leaky-bucket counter



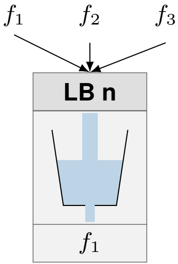
What to Do When a Flow Is Evicted?

Leaky-bucket counter



Evicted

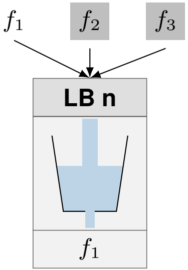
What to Do When a Flow Is Evicted?



Leaky-bucket counter

Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

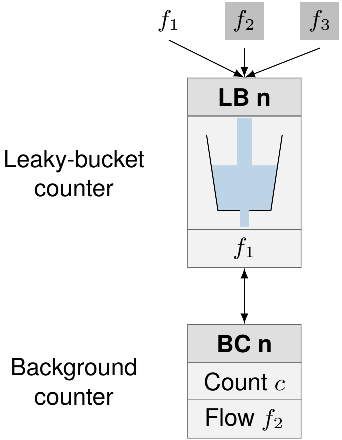
What to Do When a Flow Is Evicted?



Leaky-bucket counter

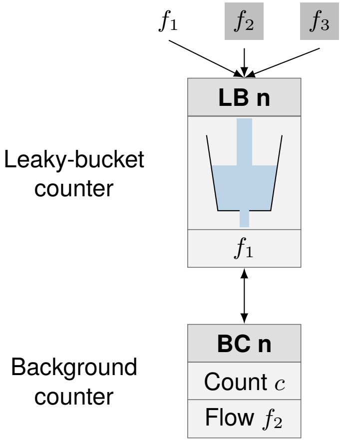
Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

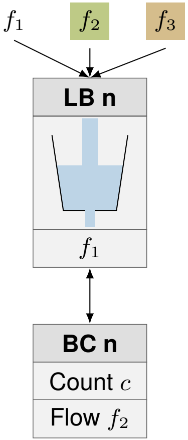
What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

What to Do When a Flow Is Evicted?



Leaky-bucket counter

Background counter

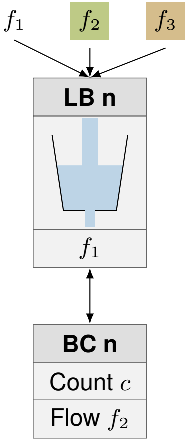
Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

Packet: Flow f_2, Size s_2

What to Do When a Flow Is Evicted?

Leaky-bucket counter



Background counter

Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

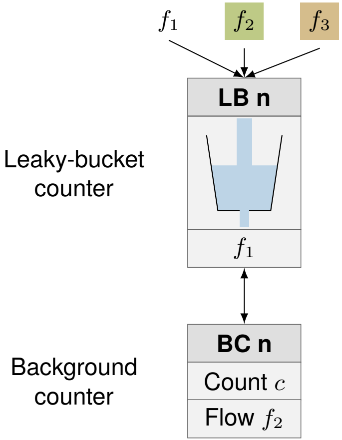
Packet: Flow f_2, Size s_2

↓

Count: $c + s_2$

Flow: f_2

What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

Packet: Flow f_2, Size s_2

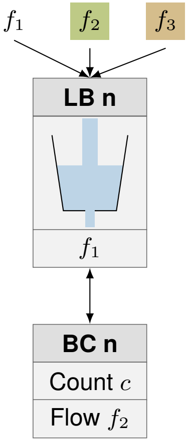
Packet: Flow f_3, Size s_3

Count: $c + s_2$

Flow: f_2

What to Do When a Flow Is Evicted?

Leaky-bucket counter



Background counter

Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

Packet: Flow f_2, Size s_2

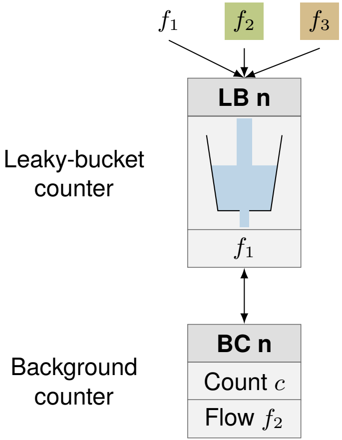
Count: $c + s_2$
Flow: f_2

Packet: Flow f_3, Size s_3

Probability p

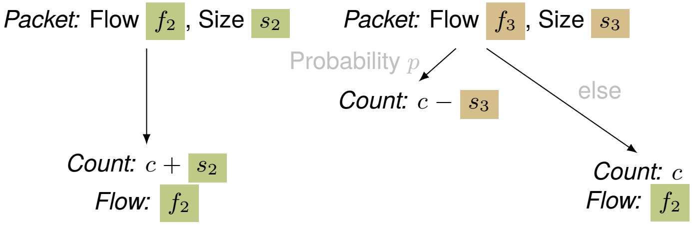
Count: $c - s_3$

What to Do When a Flow Is Evicted?

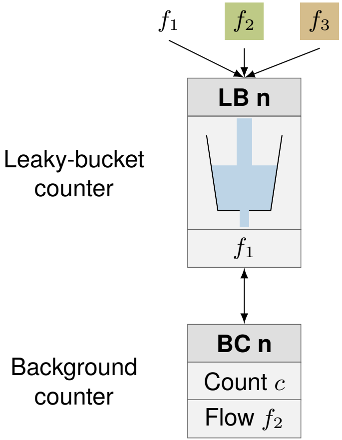


Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)



What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

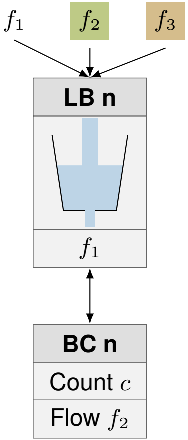
Packet: Flow f_2, Size s_2 → *Count:* $c + s_2$, *Flow:* f_2

Packet: Flow f_3, Size s_3

- Probability p → *Count:* $c - s_3$
  - $c - s_3 \leq 0$ → *Count:* s_3, *Flow:* f_3
- else → *Count:* c, *Flow:* f_2

What to Do When a Flow Is Evicted?

Leaky-bucket counter



Background counter

Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Probabilistic decay (Yang et al. 2019)

Packet: Flow f_2, Size s_2

Count: $c + s_2$
Flow: f_2

Packet: Flow f_3, Size s_3

Probability p

Count: $c - s_3$

$c - s_3 \leq 0$

Count: s_3
Flow: f_3

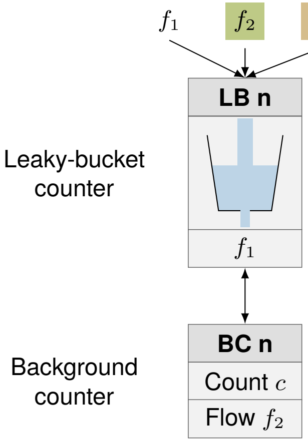
else

Count: $c - s_3$
Flow: f_2

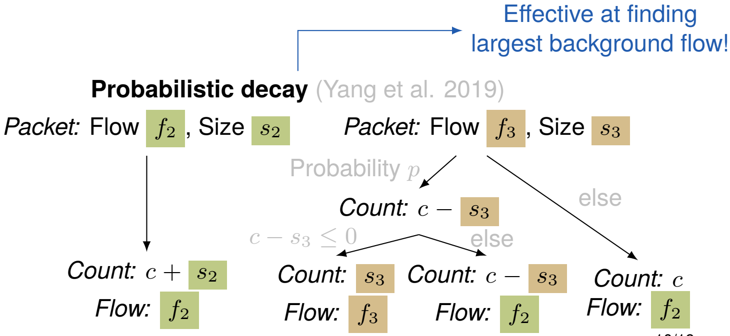
else

Count: c
Flow: f_2

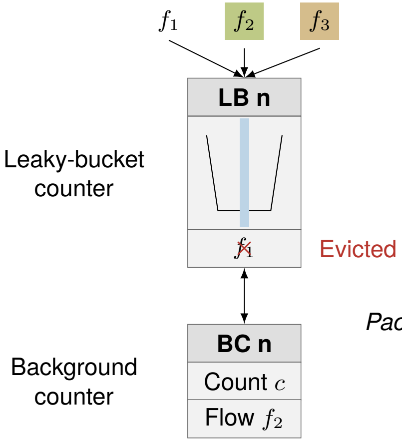
What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?



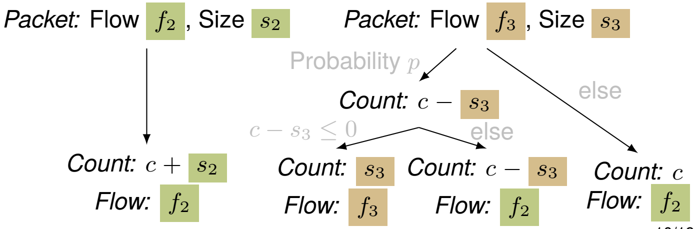
What to Do When a Flow Is Evicted?



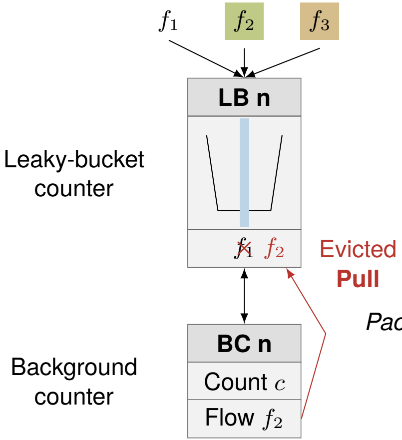
Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Effective at finding largest background flow!

Probabilistic decay (Yang et al. 2019)



What to Do When a Flow Is Evicted?



Question:
How to find the most bursty flow among the background flows of a leaky-bucket counter?

Effective at finding largest background flow!

Probabilistic decay (Yang et al. 2019)

Packet: Flow f_2, Size s_2

Count: $c + s_2$
Flow: f_2

Packet: Flow f_3, Size s_3

Probability p

Count: $c - s_3$

$c - s_3 \leq 0$

Count: s_3
Flow: f_3

else

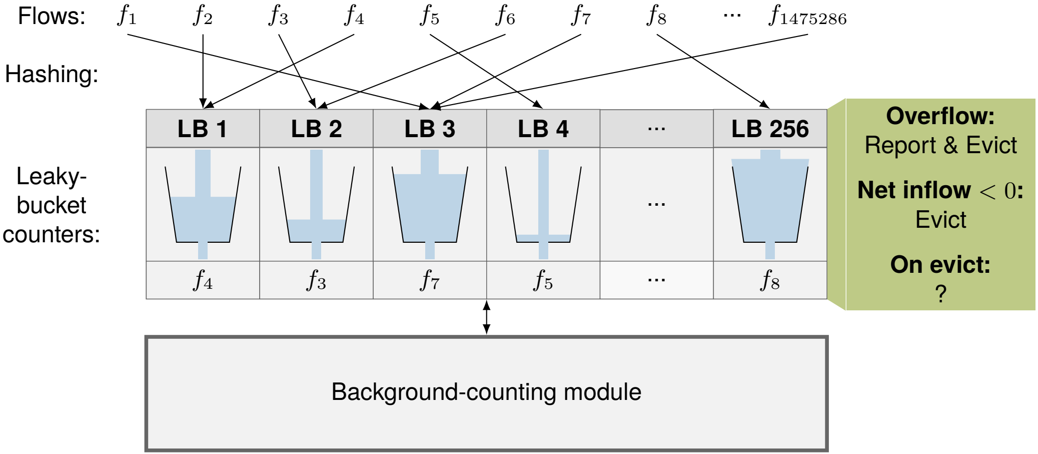
Count: $c - s_3$
Flow: f_2

else

Count: c
Flow: f_2

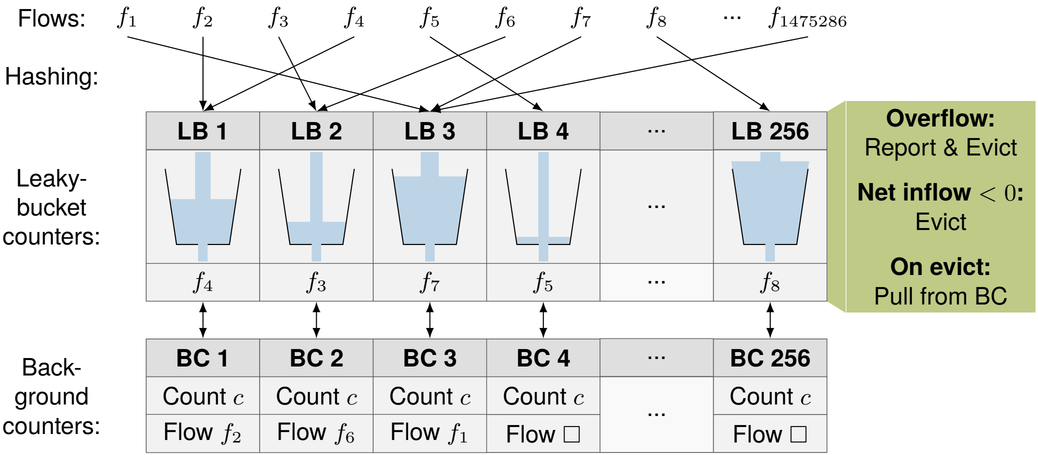
How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor



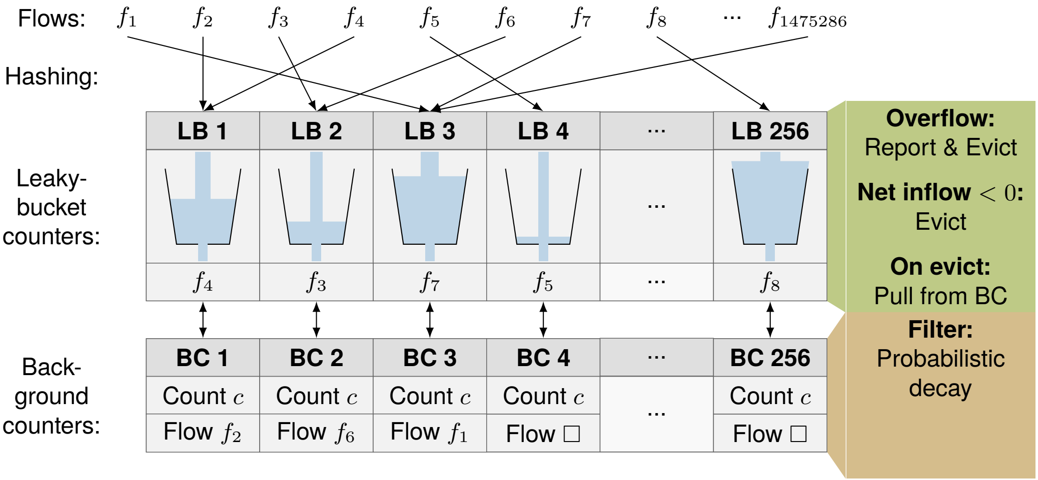
How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor



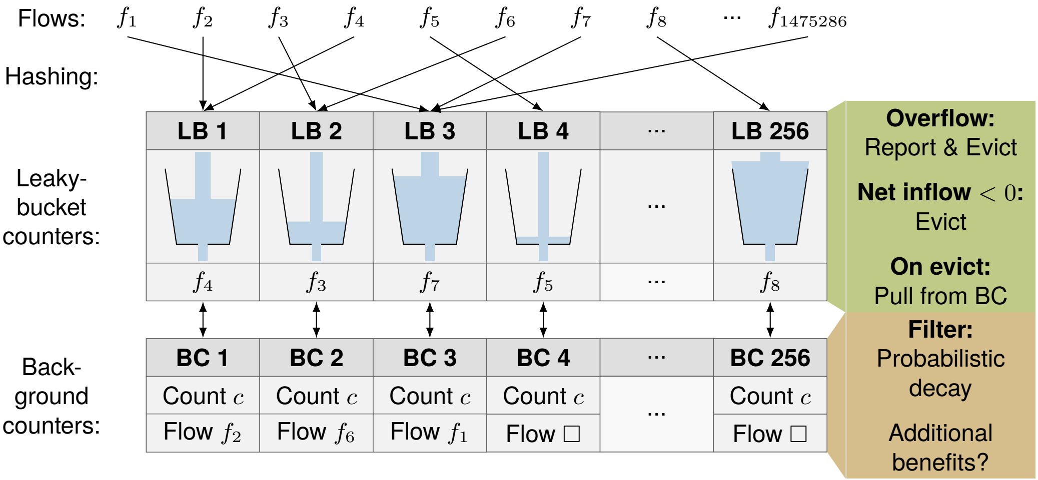
How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor



How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

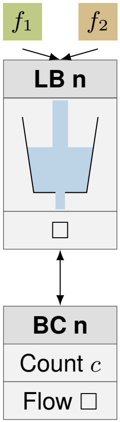


Do Background Counters Have Additional Benefits?

Background counters prevent **flow masking**

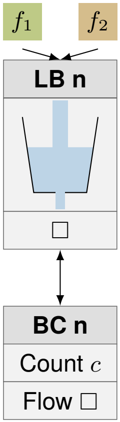
Do Background Counters Have Additional Benefits?

Background counters prevent **flow masking**



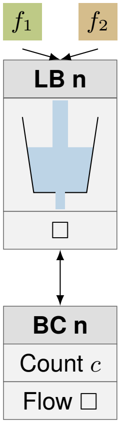
Do Background Counters Have Additional Benefits?

Background counters prevent **flow masking**



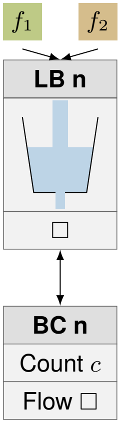
Do Background Counters Have Additional Benefits?

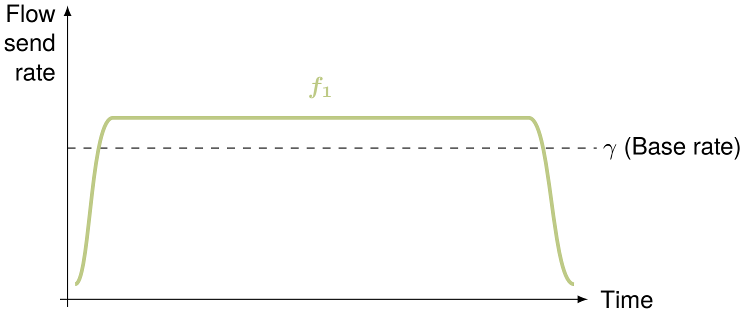
Background counters prevent **flow masking**



Do Background Counters Have Additional Benefits?

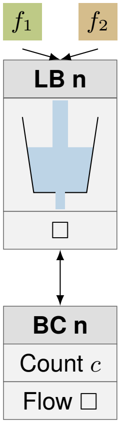
Background counters prevent **flow masking**

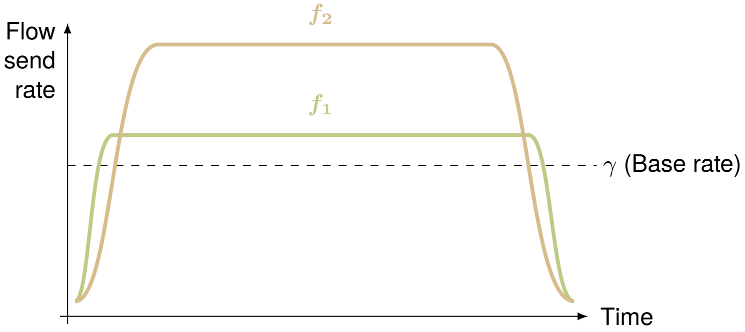




Do Background Counters Have Additional Benefits?

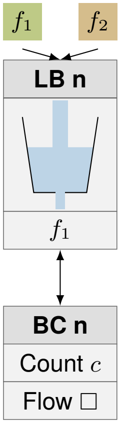
Background counters prevent **flow masking**

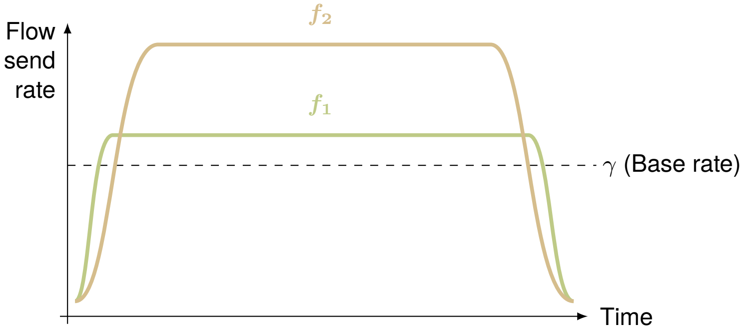




Do Background Counters Have Additional Benefits?

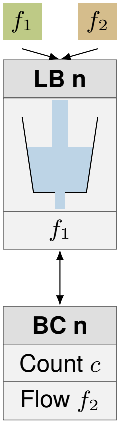
Background counters prevent **flow masking**

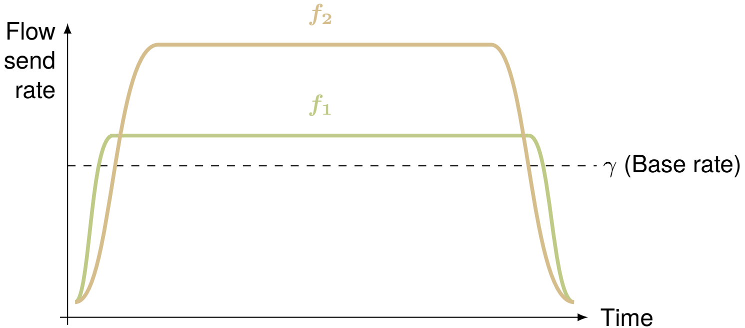




Do Background Counters Have Additional Benefits?

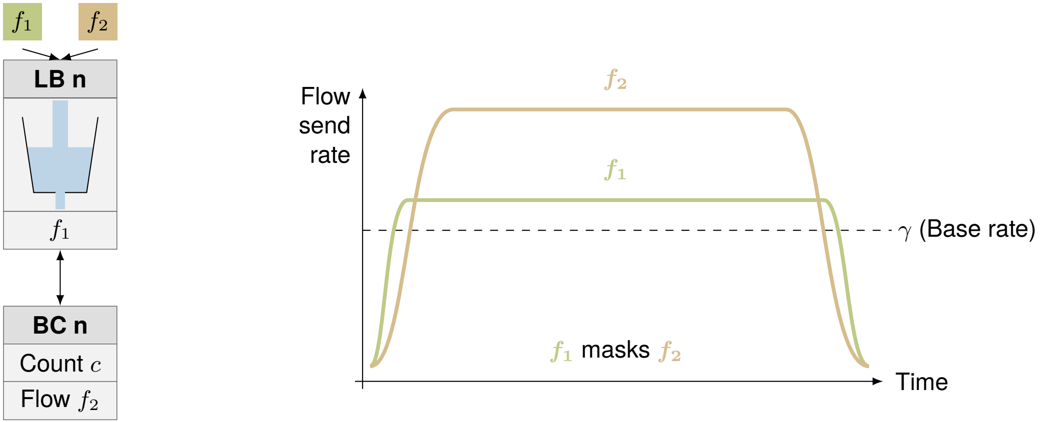
Background counters prevent **flow masking**





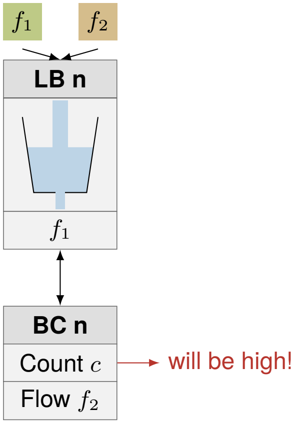
Do Background Counters Have Additional Benefits?

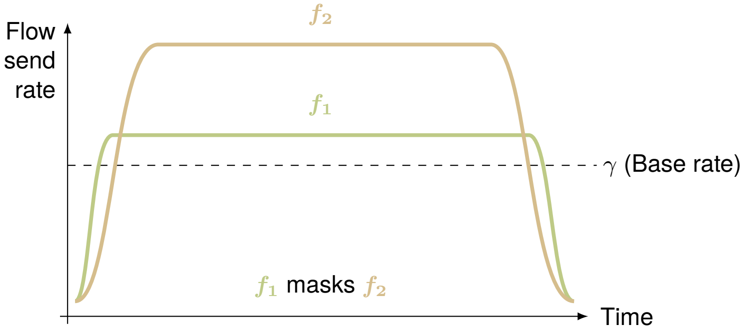
Background counters prevent **flow masking**



Do Background Counters Have Additional Benefits?

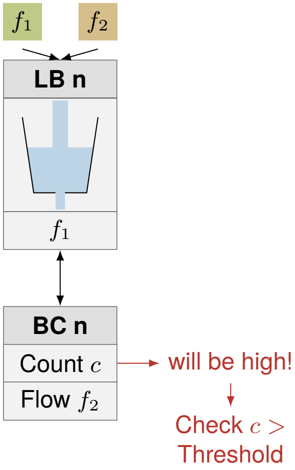
Background counters prevent **flow masking**

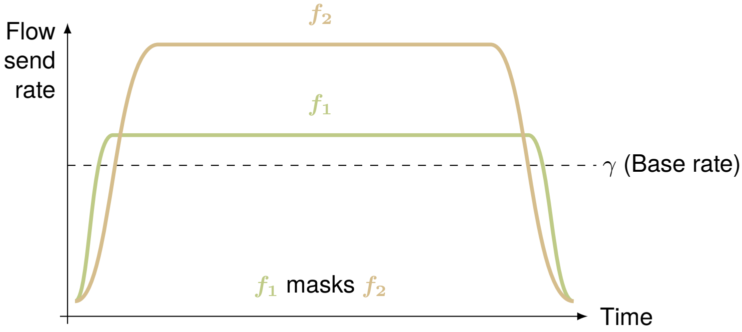




Do Background Counters Have Additional Benefits?

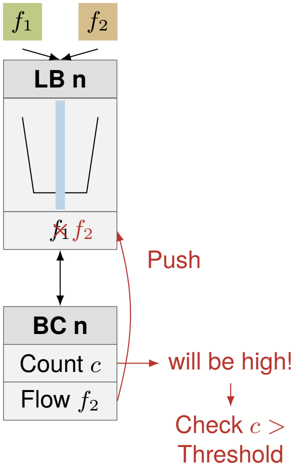
Background counters prevent **flow masking**

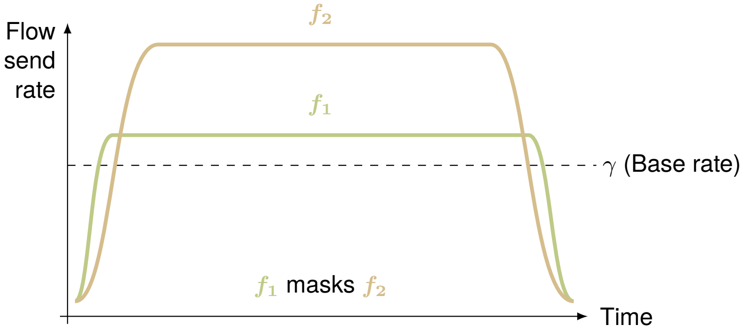




Do Background Counters Have Additional Benefits?

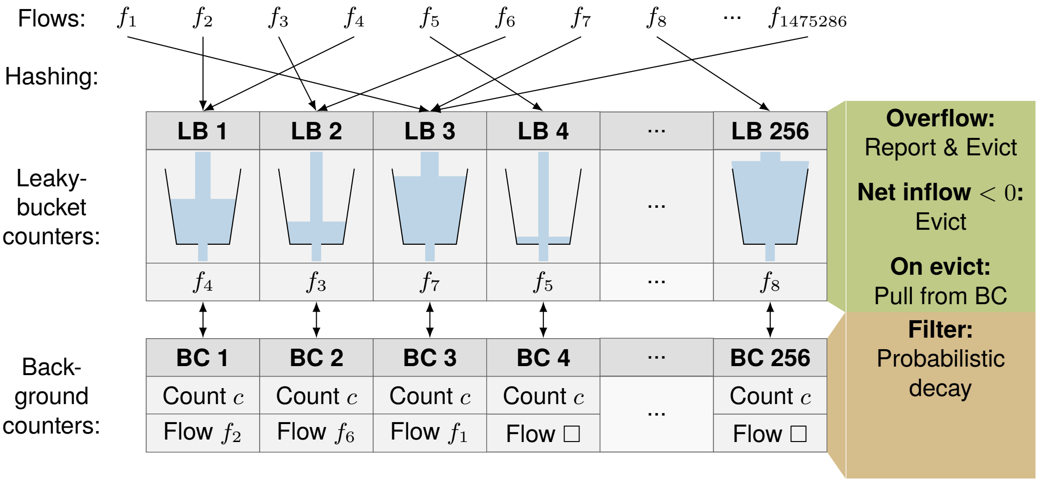
Background counters prevent **flow masking**





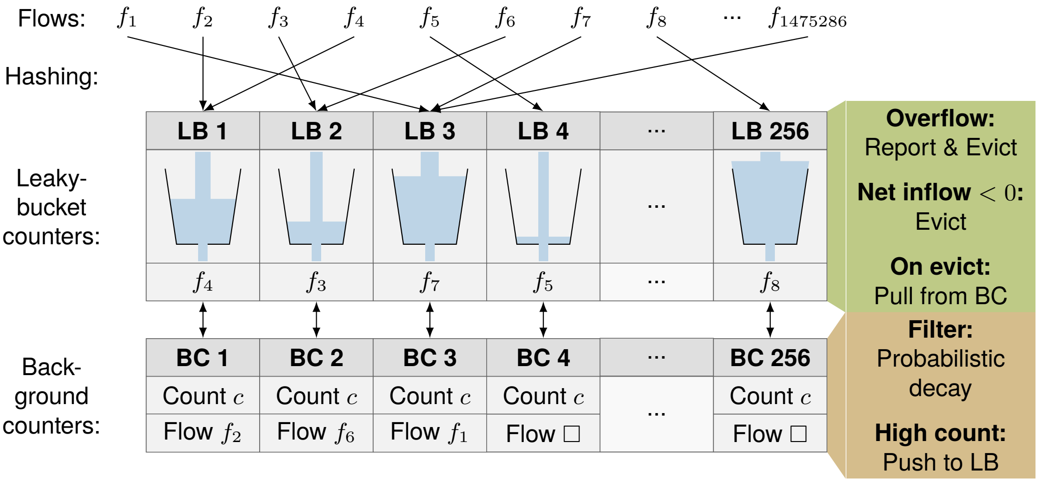
How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor



How Does ALBUS Work?

ALBUS: Adaptive Leaky-Bucket Undulation Sensor

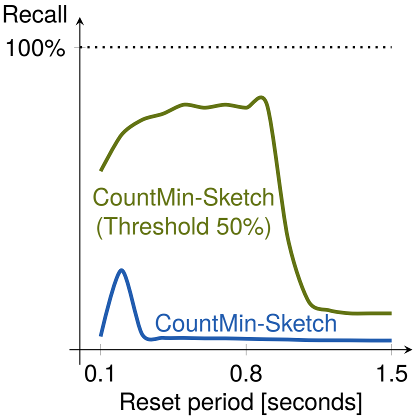


How Does ALBUS Perform Under Burst-Flood Attacks?

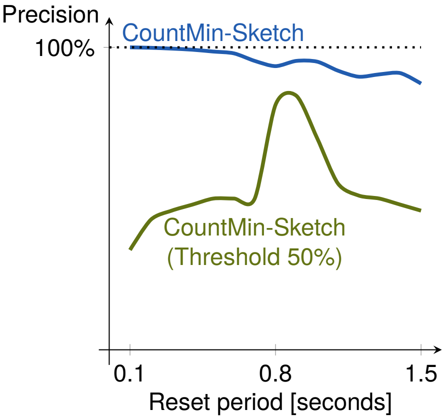
How Does ALBUS Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$



$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$

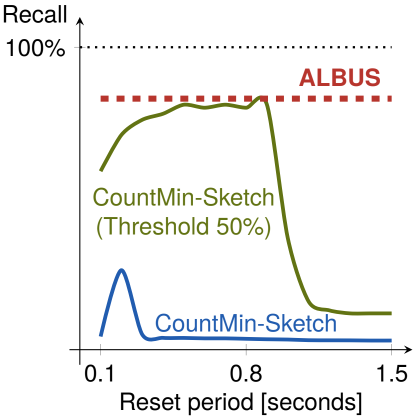


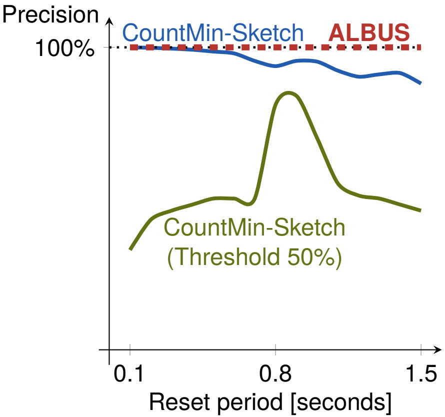
How Does ALBUS Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



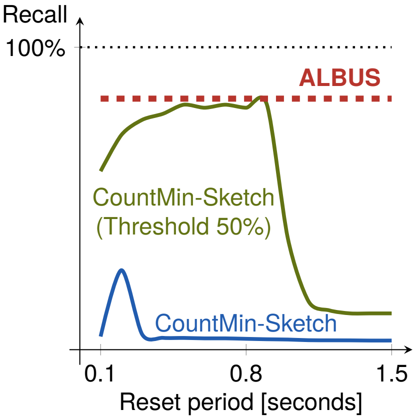


How Does ALBUS Perform Under Burst-Flood Attacks?

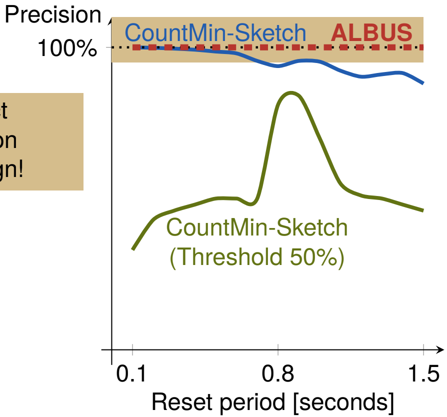
Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



Perfect precision by design!

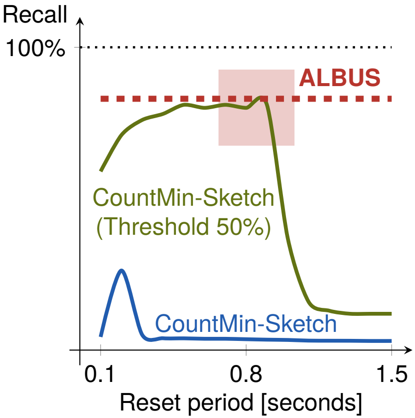


How Does ALBUS Perform Under Burst-Flood Attacks?

Scenario: Malicious bursts last for 200 milliseconds and are 20% larger than allowed

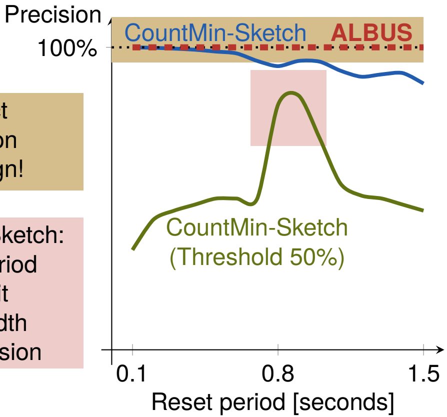
$$\text{Recall} = \frac{\text{Reported malicious bursts}}{\text{Malicious bursts}}$$

$$\text{Precision} = \frac{\text{Reported malicious bursts}}{\text{Reported bursts}}$$



Perfect precision by design!

CountMin-Sketch: Reset period must fit burst width $\implies$ Evasion



Does ALBUS Satisfy All Requirements of Effective Burst Detection?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

Time-window flexibility

Memory efficiency

Processing efficiency

Does ALBUS Satisfy All Requirements of Effective Burst Detection?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:

Time-window flexibility

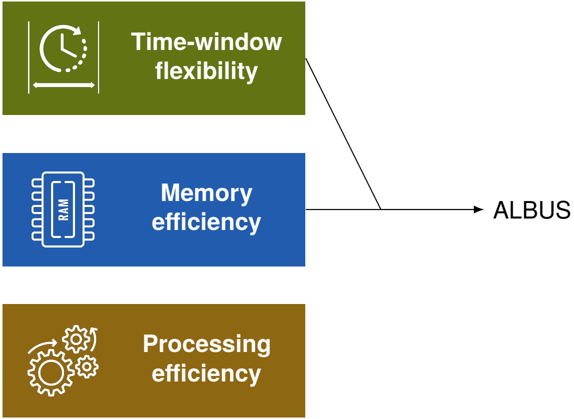
Memory efficiency

Processing efficiency

ALBUS

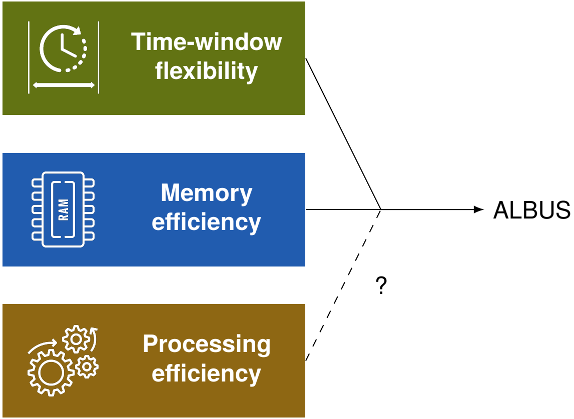
Does ALBUS Satisfy All Requirements of Effective Burst Detection?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:



Does ALBUS Satisfy All Requirements of Effective Burst Detection?

To withstand burst-flood attacks, a monitoring algorithm must satisfy the following requirements:



Does ALBUS Allow Efficient Processing?

Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation

Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation

No counter-array iterations

Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation

No counter-array iterations

No associative arrays

Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation | No counter-array iterations | No associative arrays

$\Longrightarrow$ ALBUS is hardware-friendly

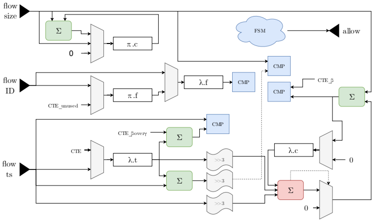
Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation | No counter-array iterations | No associative arrays

$\Longrightarrow$ ALBUS is hardware-friendly

FPGA implementation for Xilinx Virtex UltraScale+ FPGA:



Hardware design of a LB-BC combination

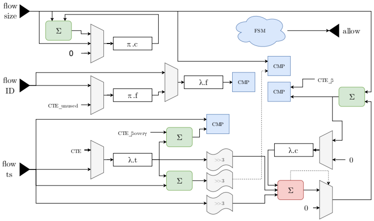
Does ALBUS Allow Efficient Processing?

Yes. ALBUS has low computational complexity:

Single hash computation | No counter-array iterations | No associative arrays

$\Longrightarrow$ ALBUS is hardware-friendly

FPGA implementation for Xilinx Virtex UltraScale+ FPGA:



Hardware design of a LB-BC combination

Throughput:
200 million packets per second
$\sim$ 560 Gbps!

Summary: What Is the Contribution of ALBUS?

Summary: What Is the Contribution of ALBUS?

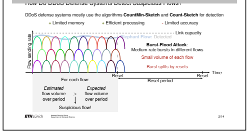
ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

Summary: What Is the Contribution of ALBUS?

ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

CountMin-Sketch
Count-Sketch

Fixed time-windows

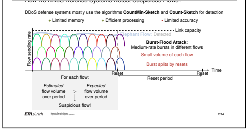


Summary: What Is the Contribution of ALBUS?

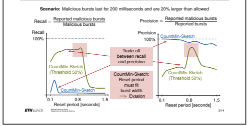
ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

CountMin-Sketch
Count-Sketch

Fixed time-windows



Accuracy trade-off

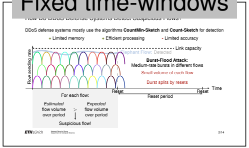


Summary: What Is the Contribution of ALBUS?

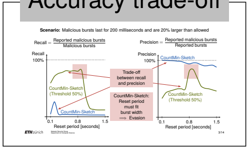
ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

CountMin-Sketch
Count-Sketch

Fixed time-windows



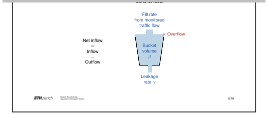
Accuracy trade-off



ALBUS

Time-window flexibility

Leaky-bucket algorithm

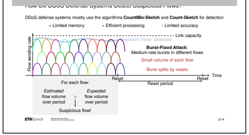


Summary: What Is the Contribution of ALBUS?

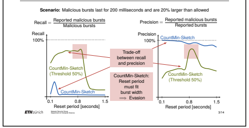
ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

CountMin-Sketch
Count-Sketch

Fixed time-windows

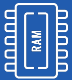


Accuracy trade-off



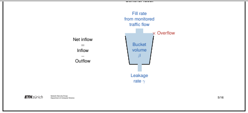
ALBUS

Time-window flexibility

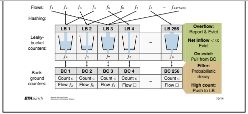


Memory efficiency

Leaky-bucket algorithm



ALBUS data-structure

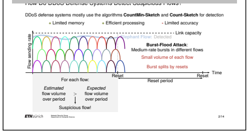


Summary: What Is the Contribution of ALBUS?

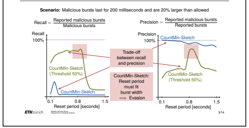
ALBUS is considerably more effective under burst-flood attacks than previous monitoring algorithms.

CountMin-Sketch
Count-Sketch

Fixed time-windows

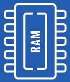


Accuracy trade-off



ALBUS

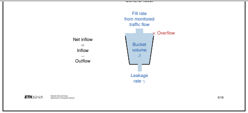
Time-window flexibility



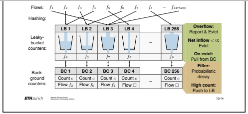
Memory efficiency

Processing efficiency

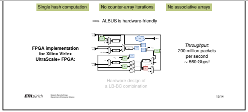
Leaky-bucket algorithm



ALBUS data-structure



FPGA implementation



Additional

Material

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g

How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

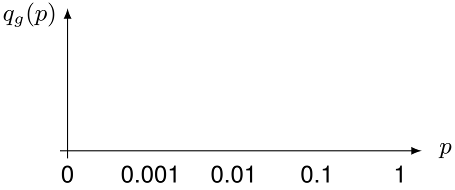
$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g



How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

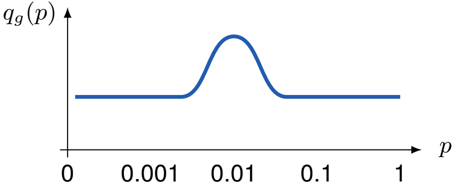
$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g



How Do Background Counters Help Detection Accuracy?

Probabilistic decay identifies large background flows

$p =$ Count-modification probability

$p = 1$: Majority algorithm (Boyer et al. 1981)

BC contains flow f with probability

$$q_f(1) = \min\left(1, \frac{\text{Volume of flow } f}{\text{Volume of all flows} \neq f}\right)$$

$p < 1$: Bias BC flow towards *largest* flow g