Programmable Networks
Lecture 3 – Stateful applications

Sándor Laki, PhD

Communication Networks Laboratory

Dept. of Information Systems, Faculty of Informatics

ELTE Eötvös Loránd University

lakis@elte.hu

http://lakis.web.elte.hu

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

this week

- Stateful programming
  - How to store state information?
- Fast reroute – an example application
- Probabilistic data structures I
  - Bloom filters

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

Stateful programming

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

stateless vs stateful

stateless objects	reinitialized for each packet variables headers
stateful objects	keep state between packets tables registers counters meters …

stateful objects

tables	managed by the control plane
register (extern in v1model)	store arbitrary data can be managed by both data and control planes
counter (extern in v1model)	count events like number of table entry matches
meter (extern in v1model)	assign „colors” to packets rate-limiting

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

register

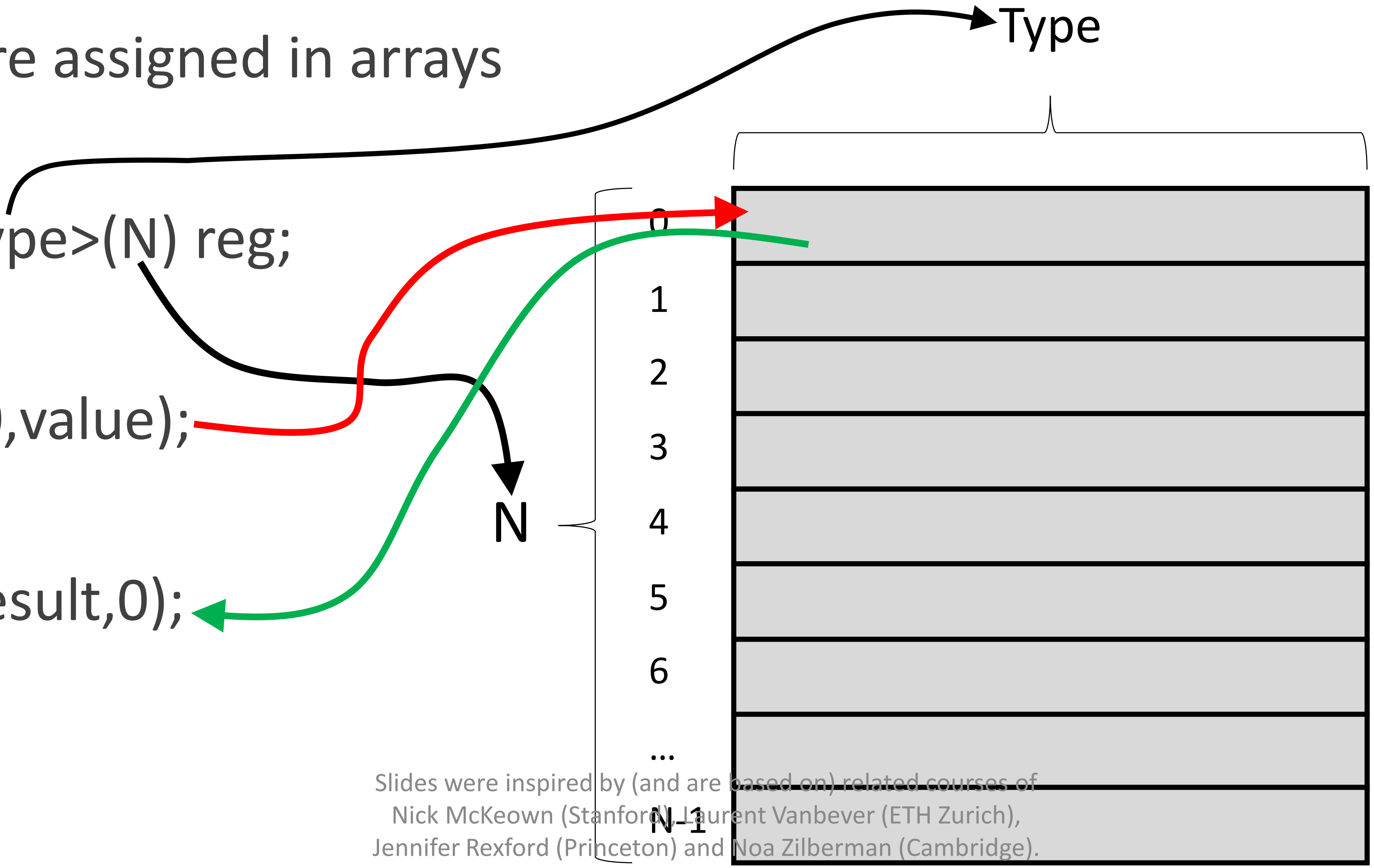
stores small amount of arbitrary data

registers are assigned in arrays

register<Type>(N) reg;

reg.write(0,value);

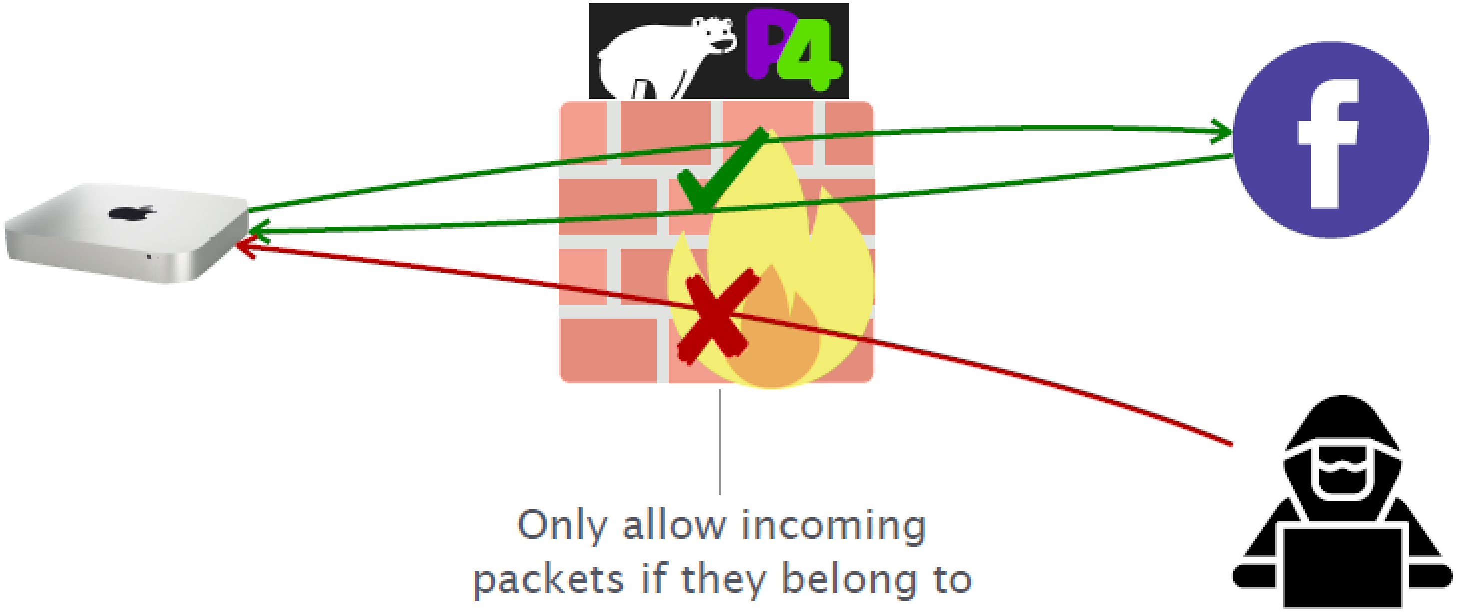
reg.read(result,0);



register – calculating inter packet gap

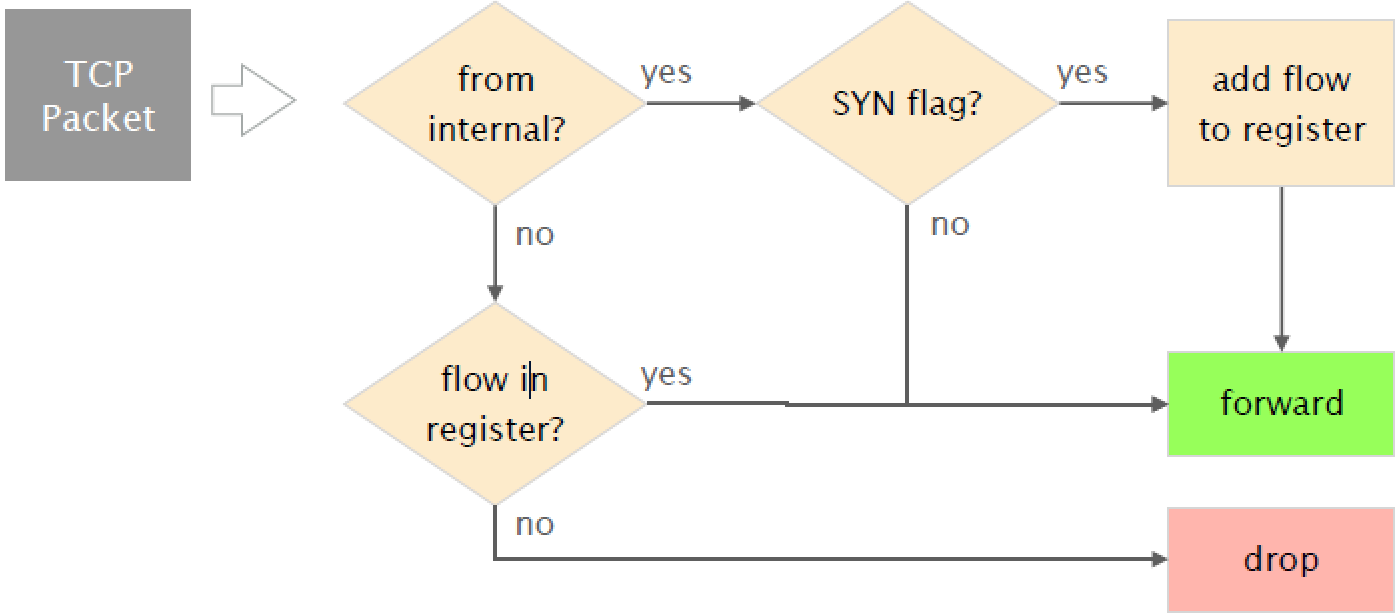
```
register<bit<48>>(16384) last_seen;

action get_inter_packet_gap(out bit<48> interval, bit<32> flow_id)
{
    bit<48> last_pkt_ts;
    /* Get the time the previous packet was seen */
    last_seen.read(last_pkt_ts, flow_id);
    /* Calculate the time interval */
    interval = standard_metadata.ingress_global_timestamp – last_pkt_ts;
    /* Update the register with the new timestamp */
    last_seen.write(flow_id, standard_metadata.ingress_global_timestamp);
    …
}
```

example: stateful firewall



Only allow incoming packets if they belong to an established connection

stateful firewall



Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

stateful firewall

```
control MyIngress(...) {
    register<bit<1>>(4096) known_flows;
    ...
    apply {
        meta.flow_id = ... // hash(5-tuple)
        if (hdr.ipv4.isValid()){
                if (hdr.tcp.isValid()){
                        if (standard_metadata.ingress_port == 1){
                                if (hdr.tcp.syn == 1){
                                        known_flows.write(meta.flow_id, 1);
                                }
                        }
                        if (standard_metadata.ingress_port == 2){
                                known_flows.read(meta.flow_is_known, meta.flow_id);
                                if (meta.flow_is_known != 1){
                                        drop(); return;
                                }
                        }
                }
                ipv4_lpm.apply();
        }
    }
}
```

Registers for maintaining established connections

Add to register if a new packet with syn flag arrives from internal network.

Drop all packets comming from outside that do not belong to existing connections.

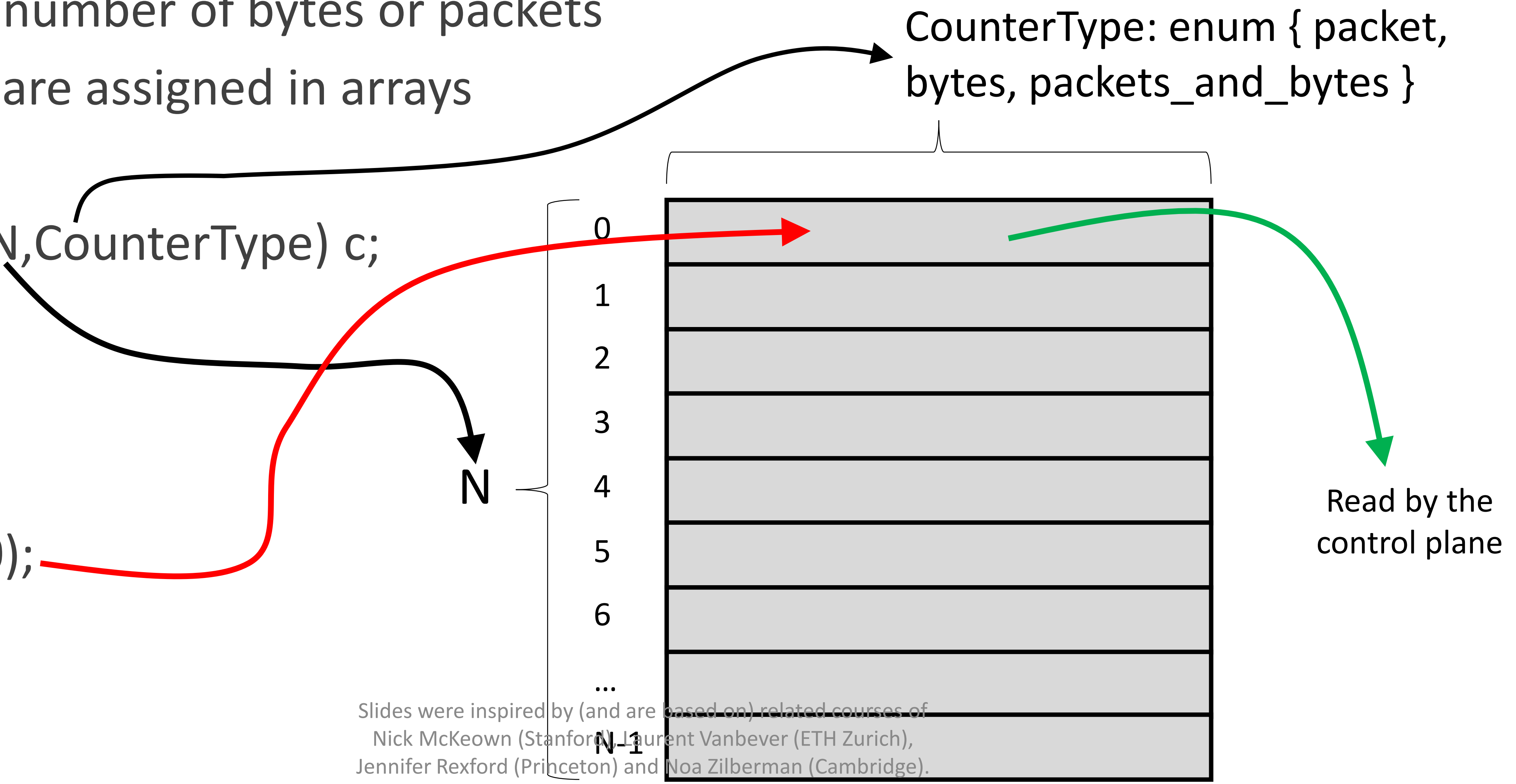
counter

counting number of bytes or packets

counters are assigned in arrays

counter(N,CounterType) c;

c.count(0);



example - port statistics

ingress port is used as counter idx

```
control MyIngress(...) {
        counter(512, CounterType.packets_and_bytes) port_counter;

        apply {
                port_counter.count((bit<32>) standard_metadata.ingress_port);
        }
}
```

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

reading counter values from the control plane

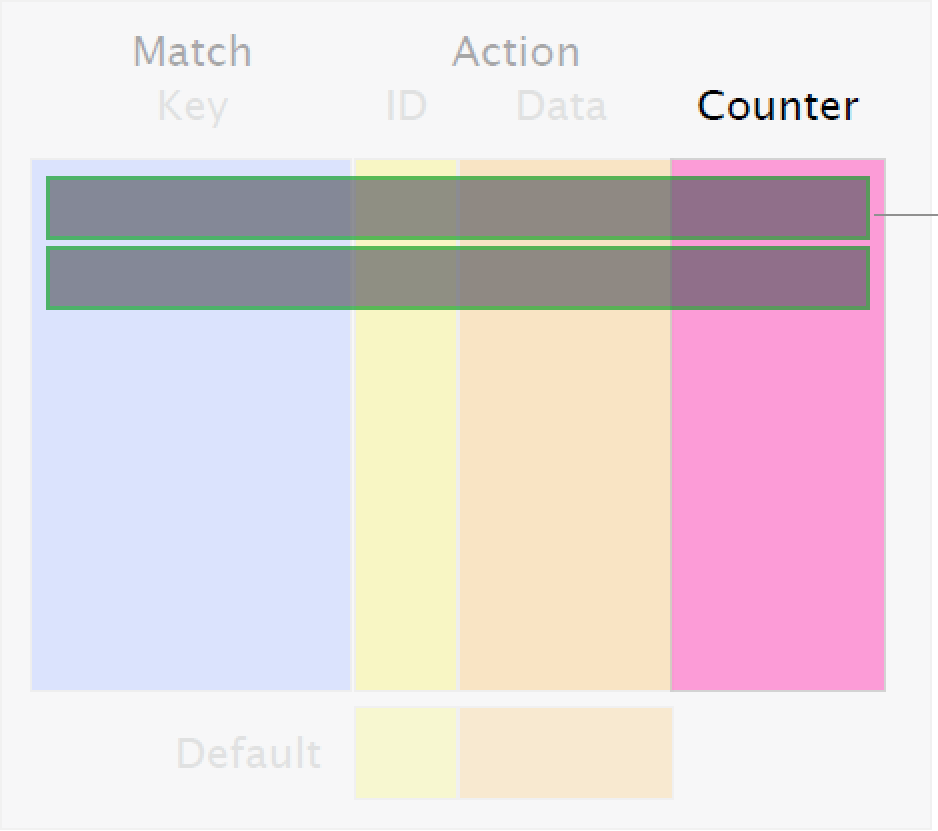
```
RuntimeCmd: counter_read MyIngress.port_counter 1
MyIngress.port_counter[1]= BmCounterValue(packets=13, bytes=1150)
```

```
control MyIngress(...) {
	counter(512, CounterType.packets_and_bytes) port_counter;

	apply {
		port_counter.count((bit<32>) standard_metadata.ingress_port);
	}
}
```

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

direct counters

special counters attached to tables



Each entry has a counter cell that counts when the entry matches

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

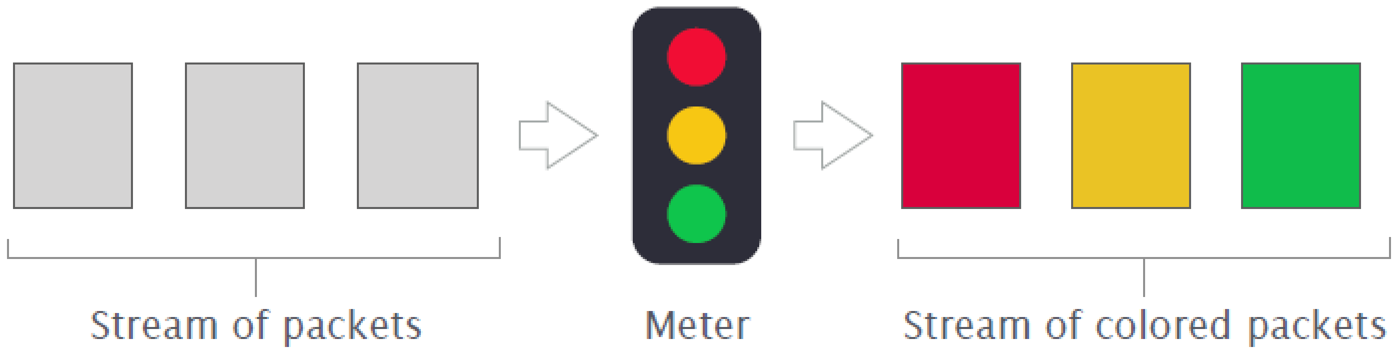
port statistics in a bit different way

```
control MyIngress(...) {
        direct_counter(CounterType.packets_and_bytes) direct_port_counter;

        table count_table {
                key = {
                        standard_metadata.ingress_port: exact;
                }
                actions = {
                        NoAction;
                }
                default_action = NoAction;
                counters = direct_port_counter;
                size = 512;
        }

        apply {
                count_table.apply();
        }
}
```

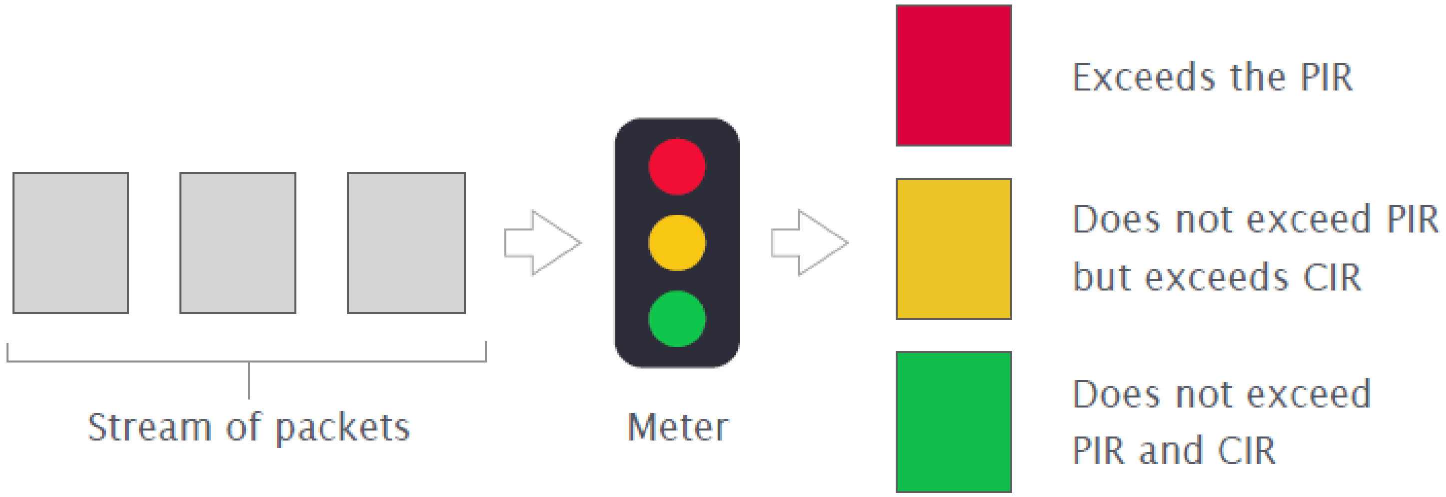
Attach counter to the table

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

meters



meters



Parameters:	PIR	Peak Information Rate	[bytes/s] or [packets/s]
	CIR	Committed Information Rate	[bytes/s] or [packets/s]

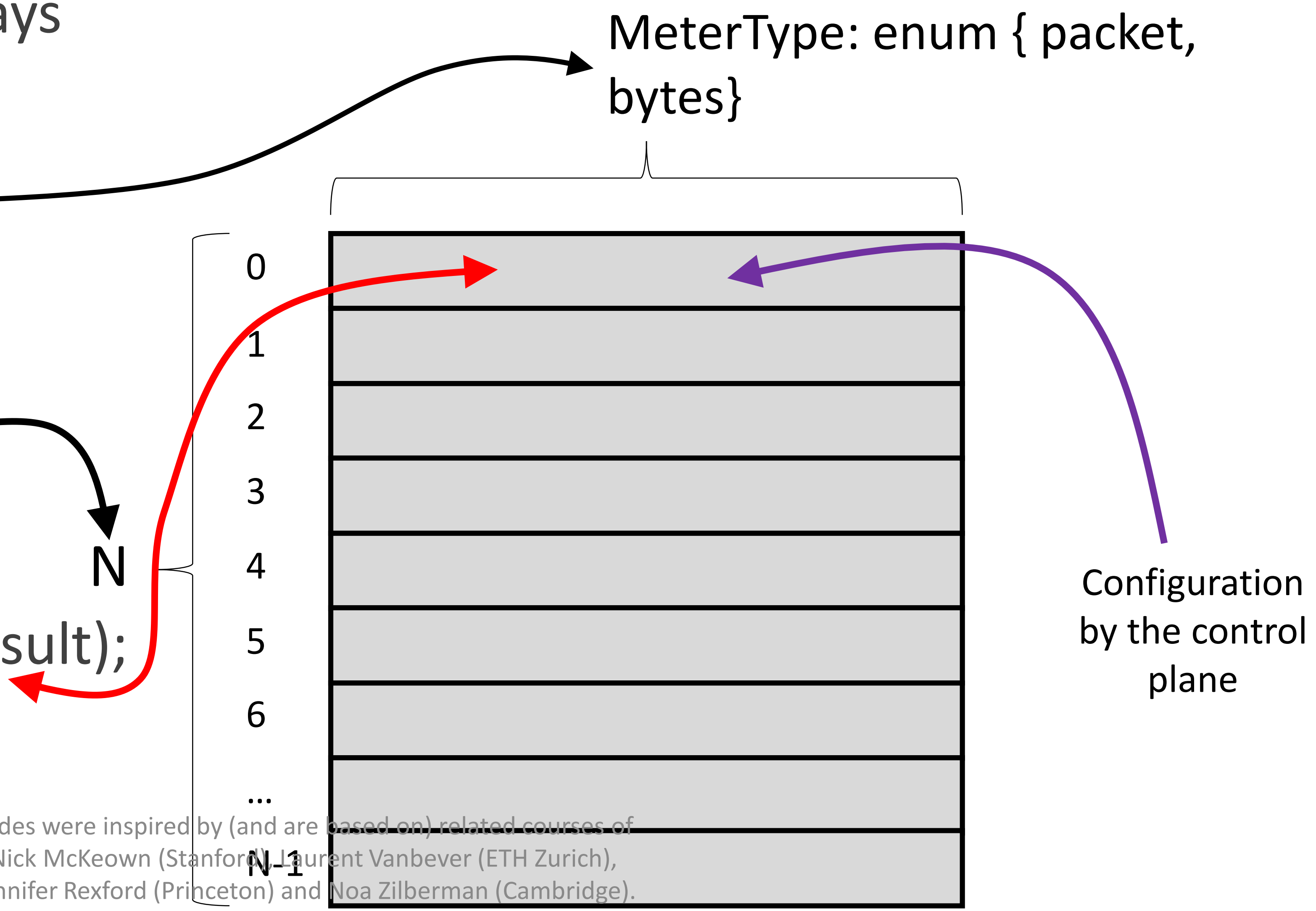
more info https://tools.ietf.org/html/rfc2698

meter

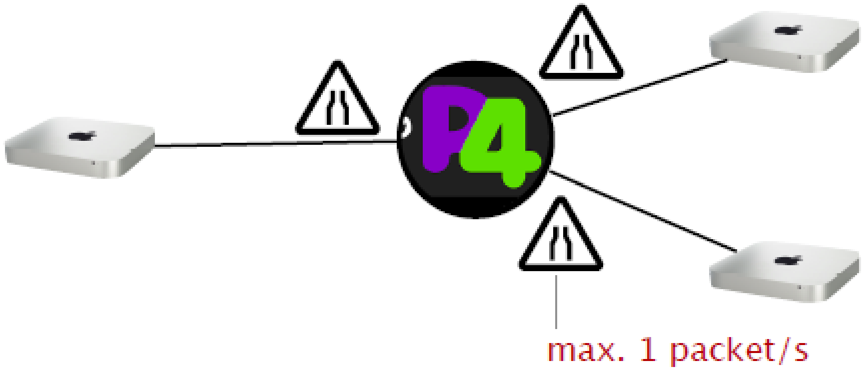
meters are assigned in arrays

meter(N,MeterType) m;

m.execute_meter<T>(0, result);

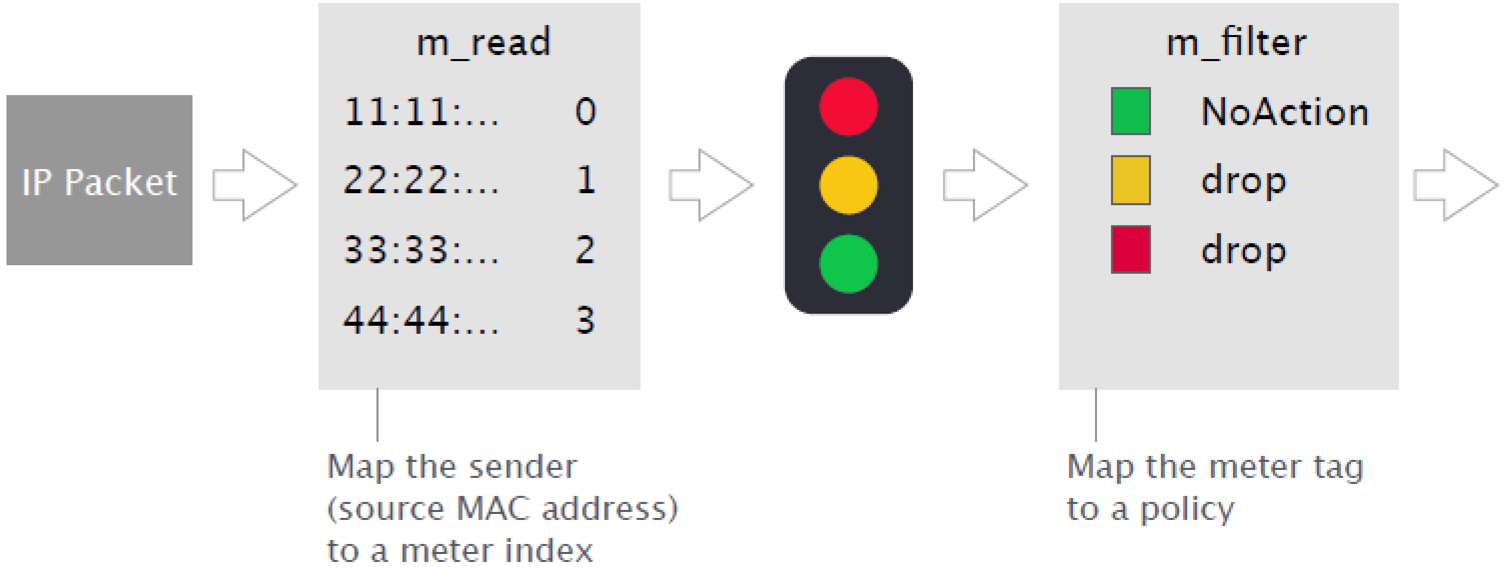


example: rate-limiter



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

example: rate-limiter



```
control MyIngress(...) {
        meter(32w16384, MeterType.packets) my_meter;

        action m_action(bit<32> meter_index) {
                my_meter.execute_meter<bit<32>>(meter_index, meta.meter_tag);
        }

        table m_read {
                key = { hdr.ethernet.srcAddr: exact; }
                actions = { m_action; NoAction; }
                ...
        }

        table m_filter {
                key = { meta.meter_tag: exact; }
                actions = { drop; NoAction; }
                ...
        }

        apply {
                m_read.apply();
                m_filter.apply();
        }
}
```

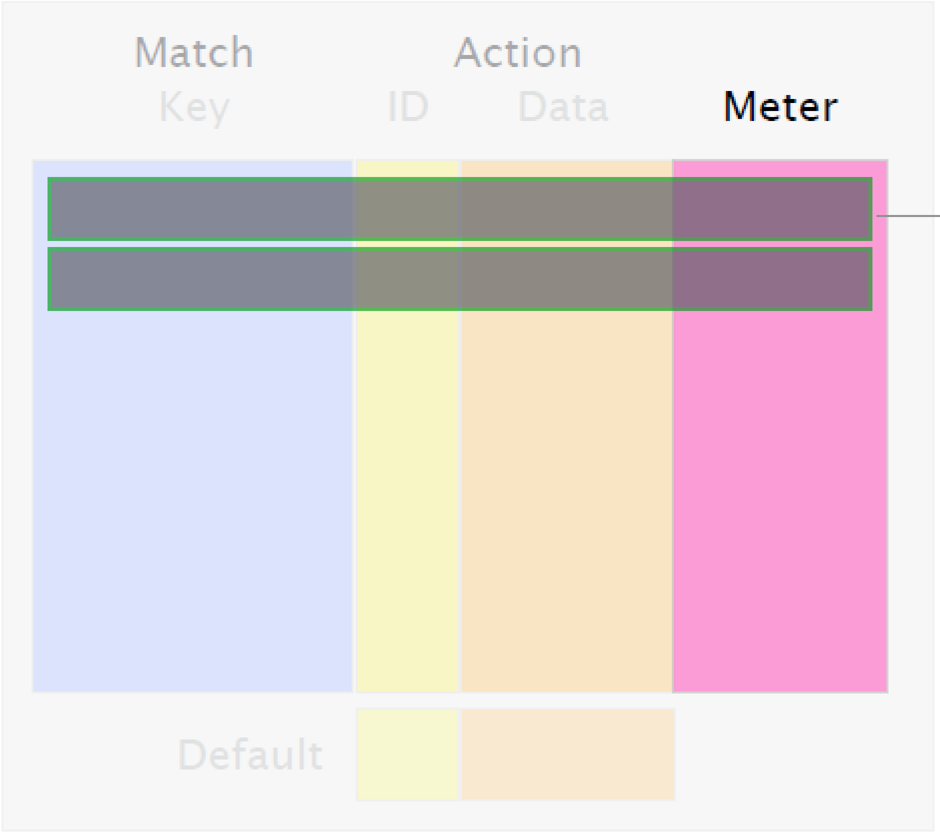
packet meter

execute meter & store the color in metafield meter_tag

Packet drop based on meter_tag

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

direct meters assigned to tables



Each entry has a meter cell that is executed when the entry matches

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

direct meter for a rate limiting use case

```
control MyIngress(...) {
        direct_meter<bit<32>>(MeterType.packets) my_meter;

        action m_action(bit<32> meter_index) {
                my_meter.read(meta.meter_tag);
        }

        table m_read {
                key = { hdr.ethernet.srcAddr: exact; }
                actions = { m_action; NoAction; }
                meters = my_meter;
        ...
        }

        table m_filter { ... }

        apply {
                m_read.apply();
                m_filter.apply();
        }
}
```

direct meter

read meter

Add a direct_meter instance to the table

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

stateful summary

	data plane		control plane	
	read	***write/modify***	***read***	***write/modify***
table	apply()	no	yes	yes
register	yes – read()	yes – write()	yes	yes
counter	no	yes – count()	yes	reset only
meter	yes	yes	no	configuration only

An example application

https://www.net.t-labs.tu-berlin.de/~stefan/neat18.pdf

https://www.youtube.com/watch?v=G4L2ys-_W9w#t=26m26s

https://p4.org/assets/P4WS_2018/Marco_Chiesa.pdf

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

Probabilistic data structures I.

Bloom filters

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

programming advanced data structs

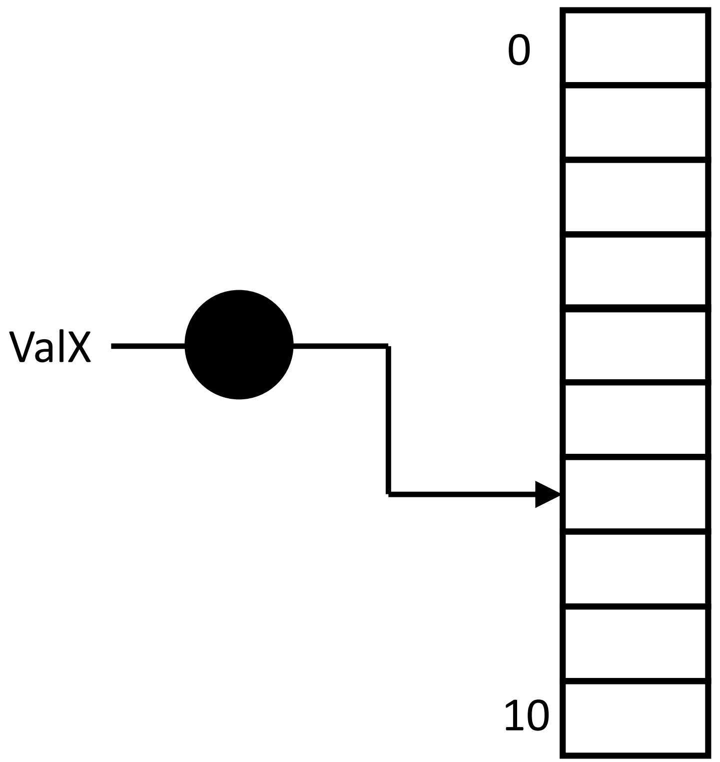
building blocks

built-in stateful data structures
arrays of registers, counters or meters

lots of limitations
limited number of operations and memory

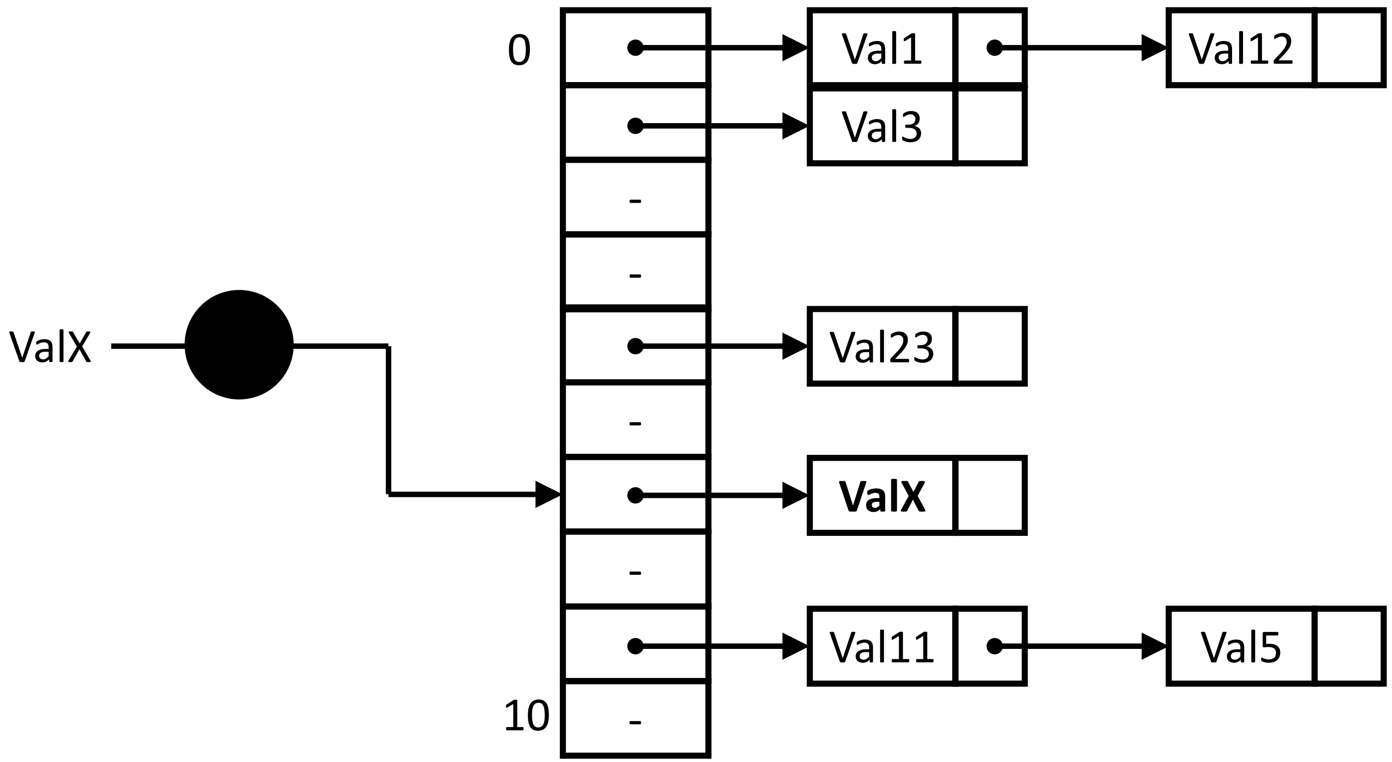
How to implement a set

1st approach – separate chaining



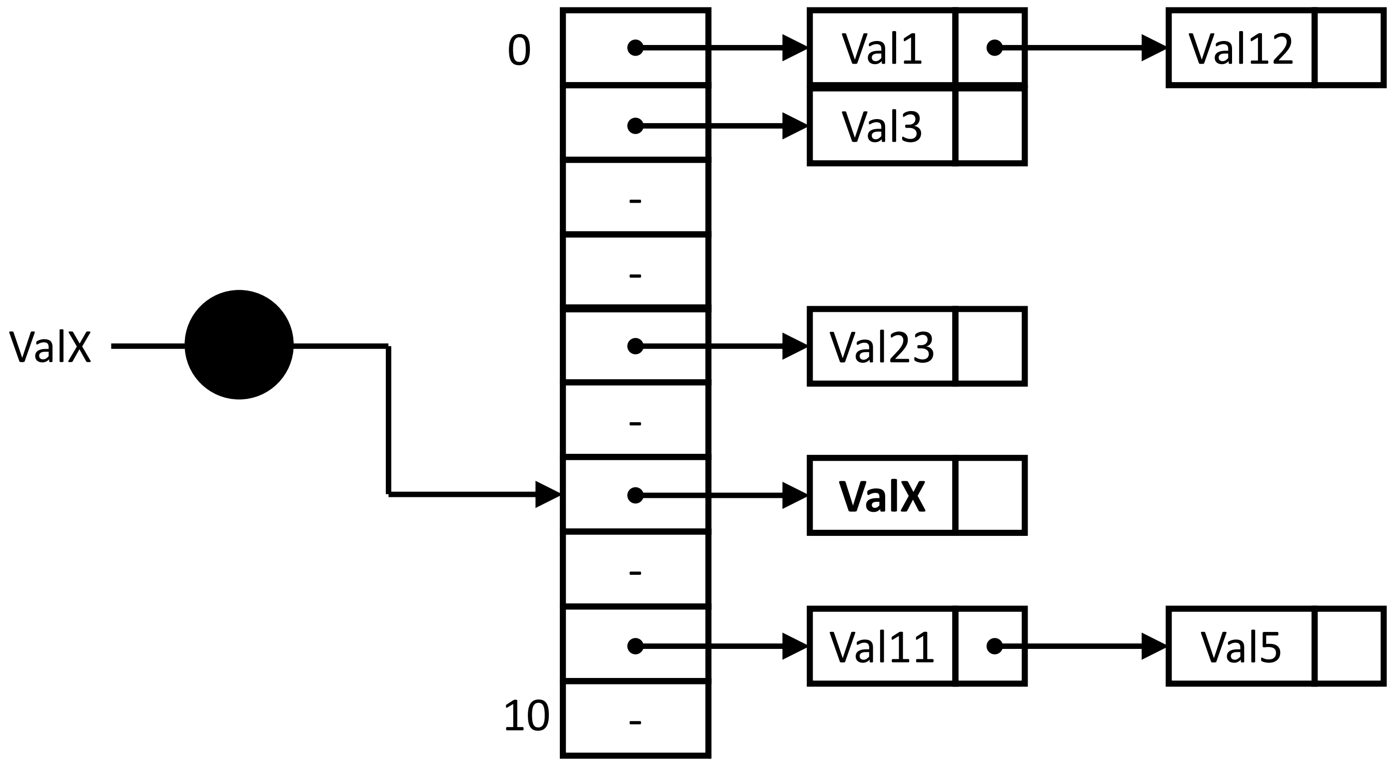
How to implement a set

1st approach – separate chaining



How to implement a set

1st approach – separate chaining

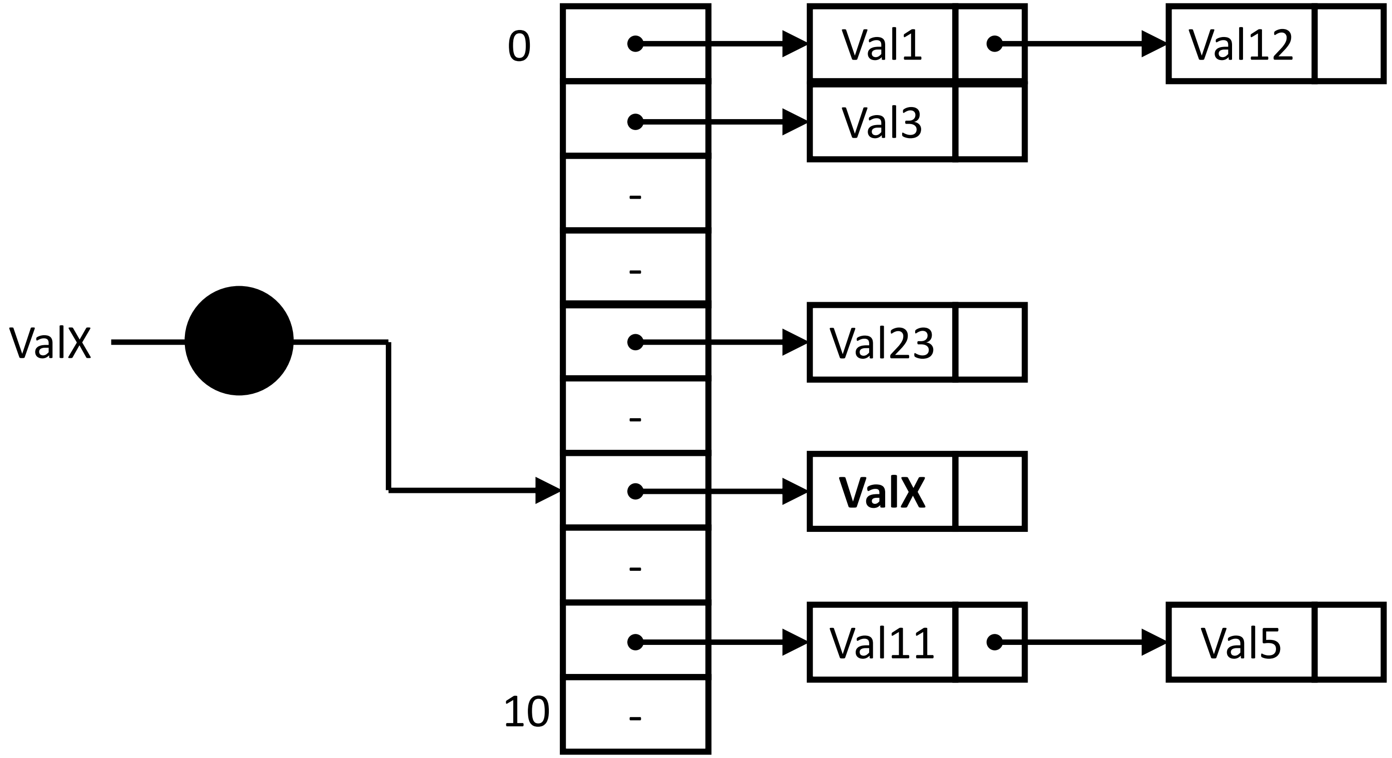


having N elements & M cells:

average list size: N/M

worst-case: N

How to implement a set
1st approach – separate chaining



having N elements & M cells:

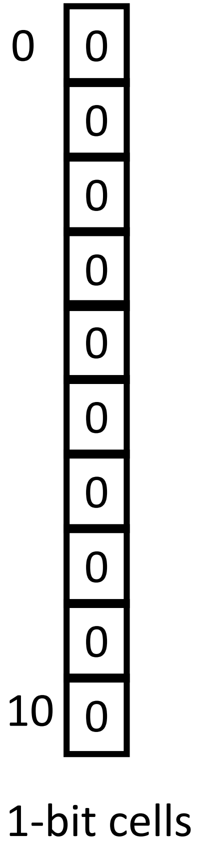
average list size: N/M
worst-case: N

Pros:
accurate and fast in the avg. case

Cons:
Only works in hw if N is small (<100)

How to implement a set

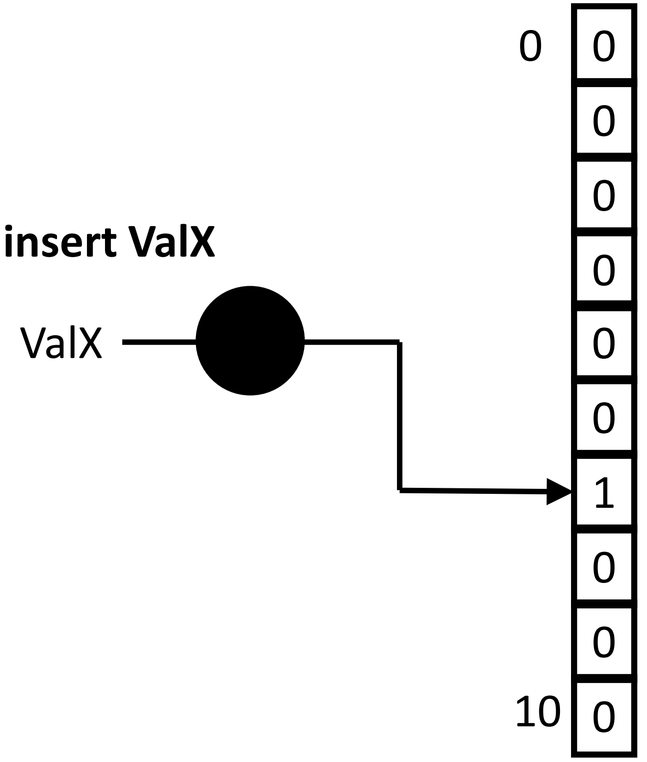
2nd approach – with probabilistic output



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

How to implement a set
2nd approach – with probabilistic output

insert ValX

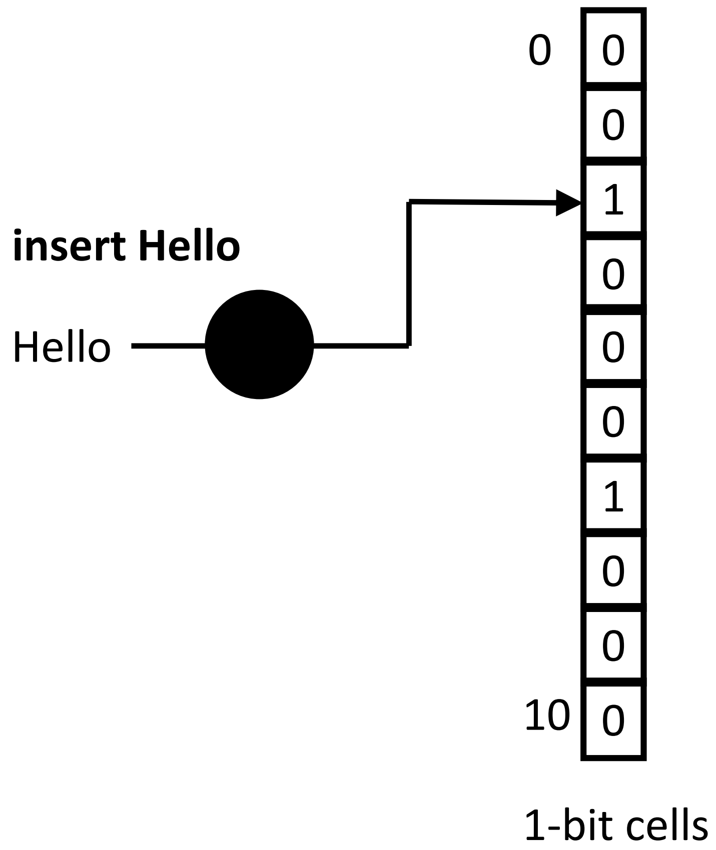


1-bit cells

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

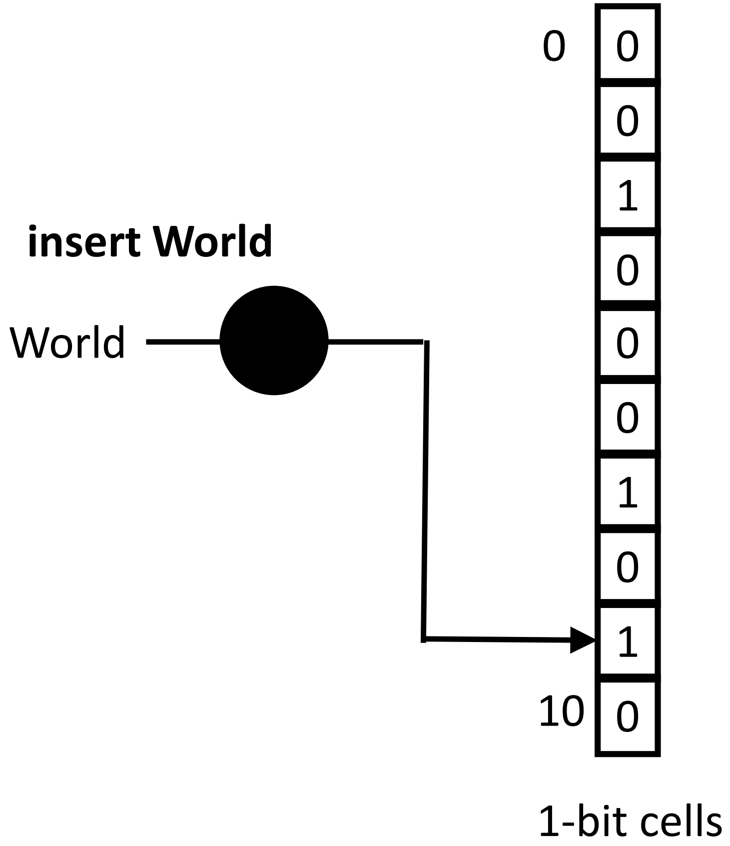
How to implement a set

2nd approach – with probabilistic output



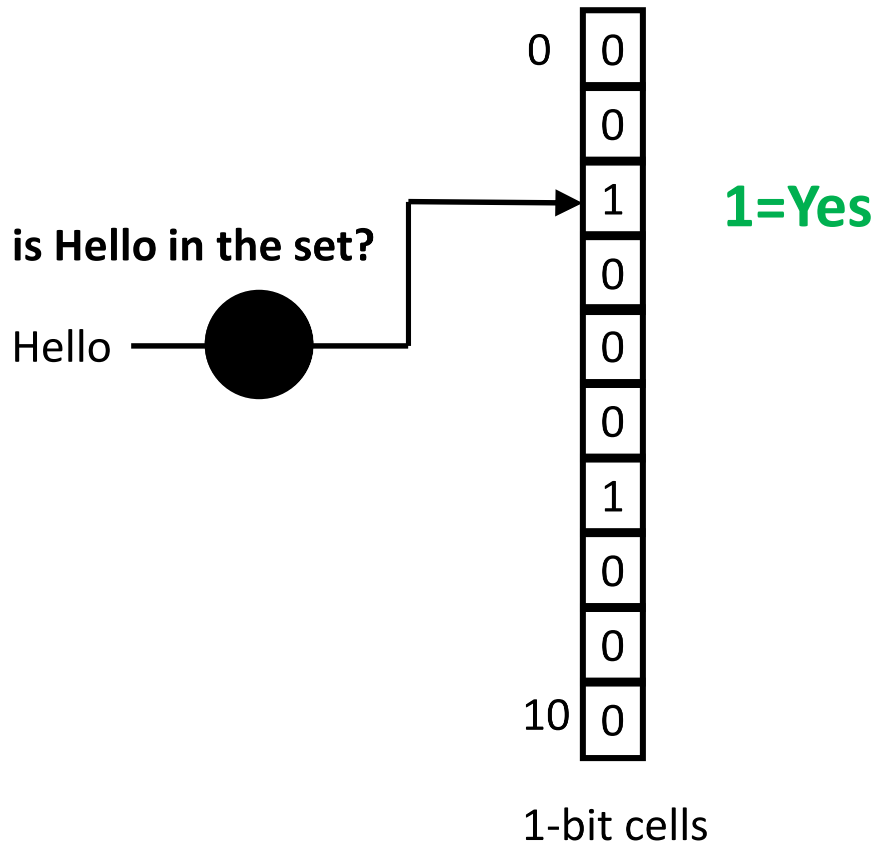
Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

How to implement a set
2nd approach – with probabilistic output



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

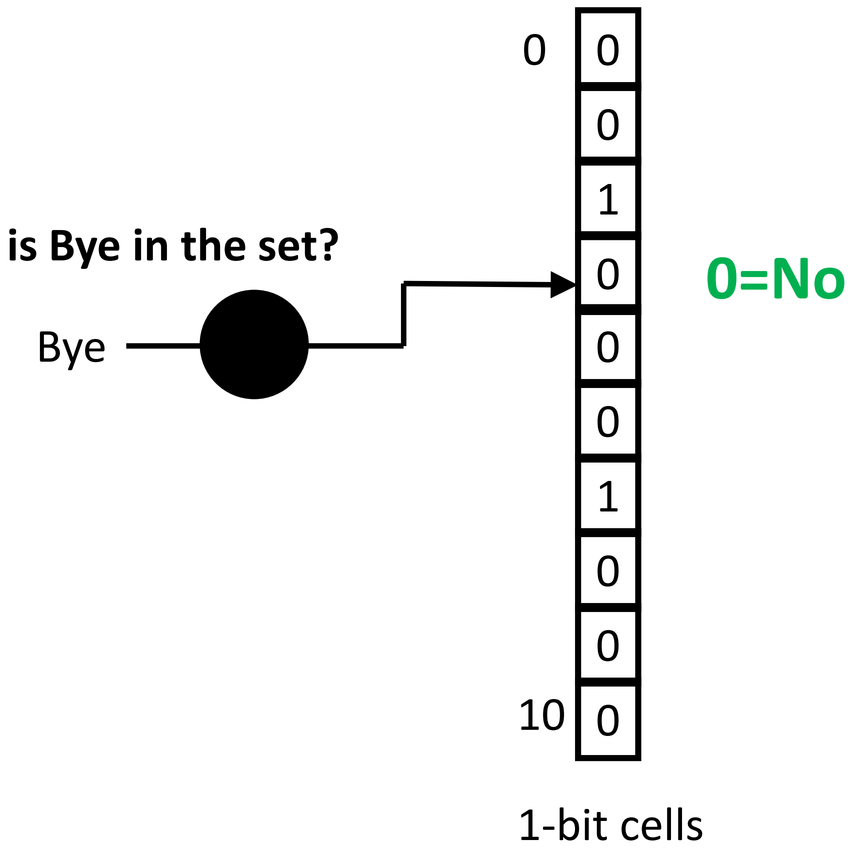
How to implement a set
2nd approach – with probabilistic output



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

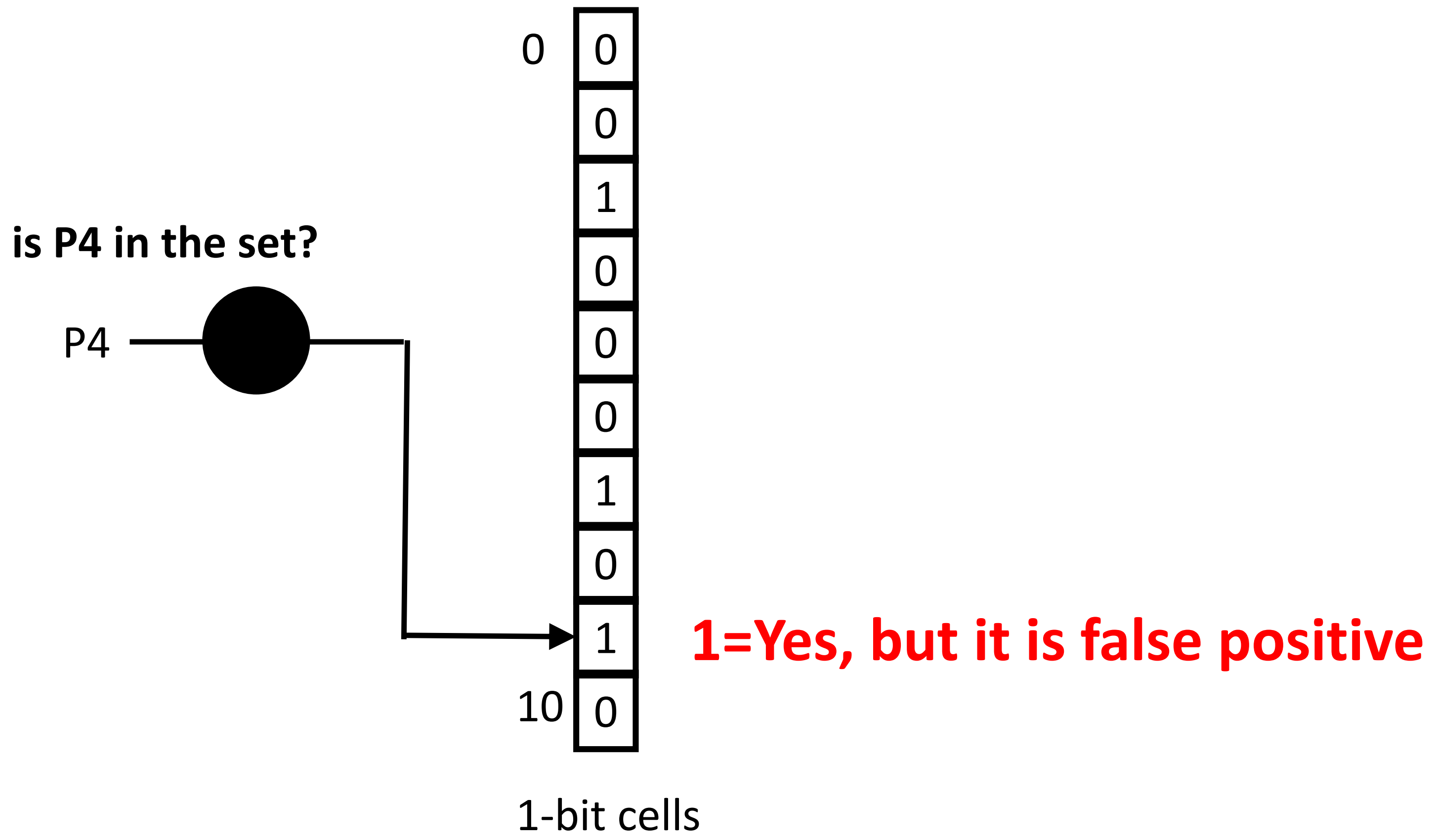
How to implement a set

2nd approach – with probabilistic output



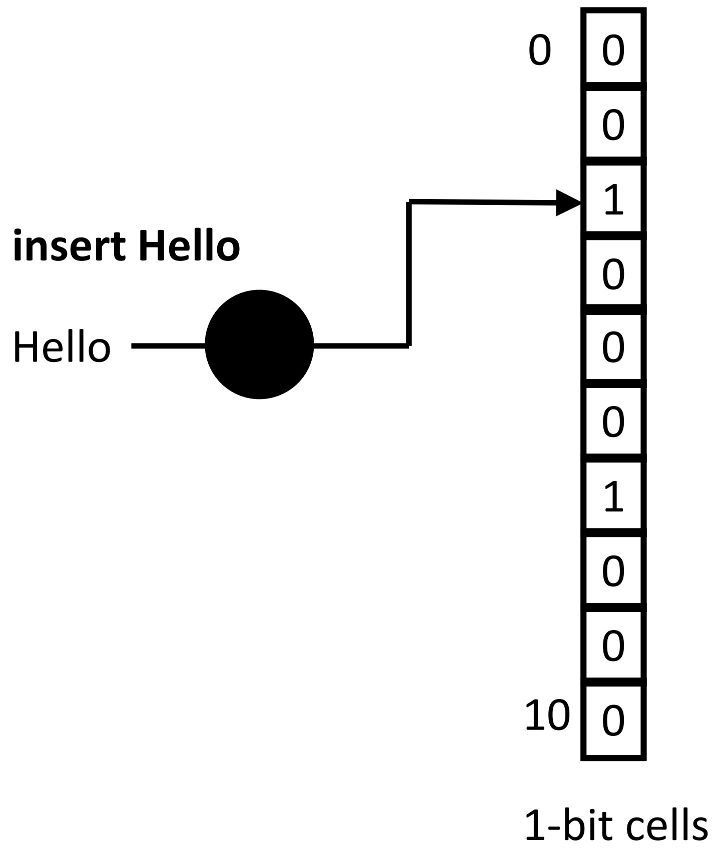
How to implement a set

2nd approach – with probabilistic output



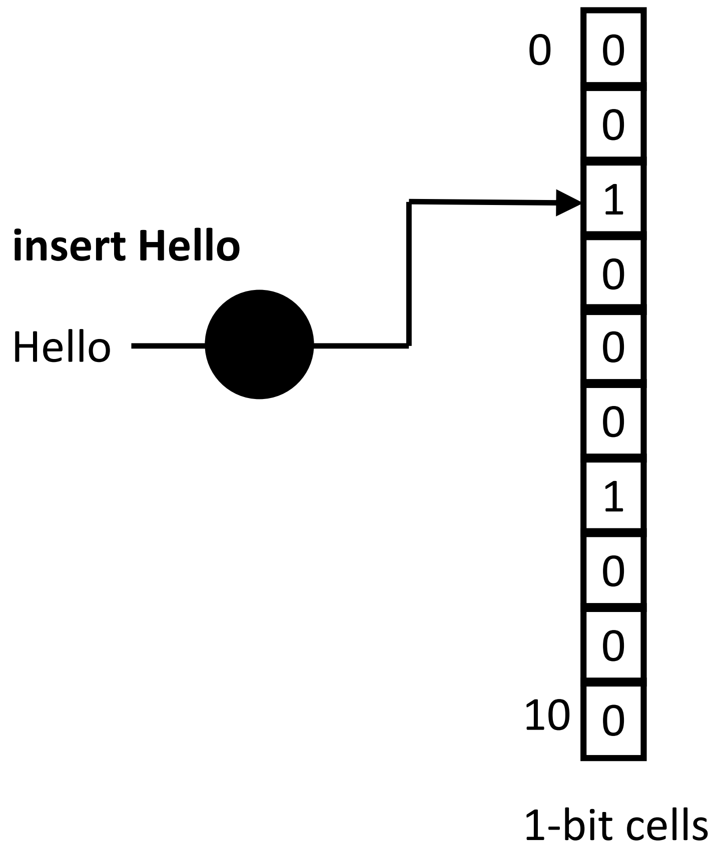
Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

How to implement a set
2nd approach – with probabilistic output



simple approach	
having N elements and M cells	
probability of an element to be mapped into a particular cell	**1/M**
probability of an element not to be mapped into a particular cell	**1-1/M**
probability of a cell to be 0	$(1-1/M)^N$
false positive rate (FPR)	$1-(1-1/M)^N$
false negative rate	**0**

How to implement a set
2nd approach – with probabilistic output

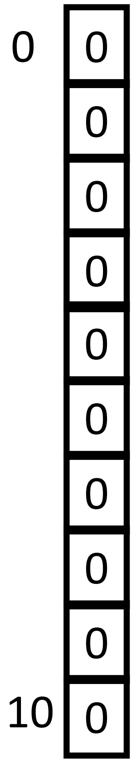


N	M	FPR
1000	10000	9.5%
1000	100000	1%

Pros:
simple and only one operationo per insertion and query

Cons:
roughly 100x more cells are required then N for a 1% FPR

How to implement a set
3rd approach – Bloom Filters

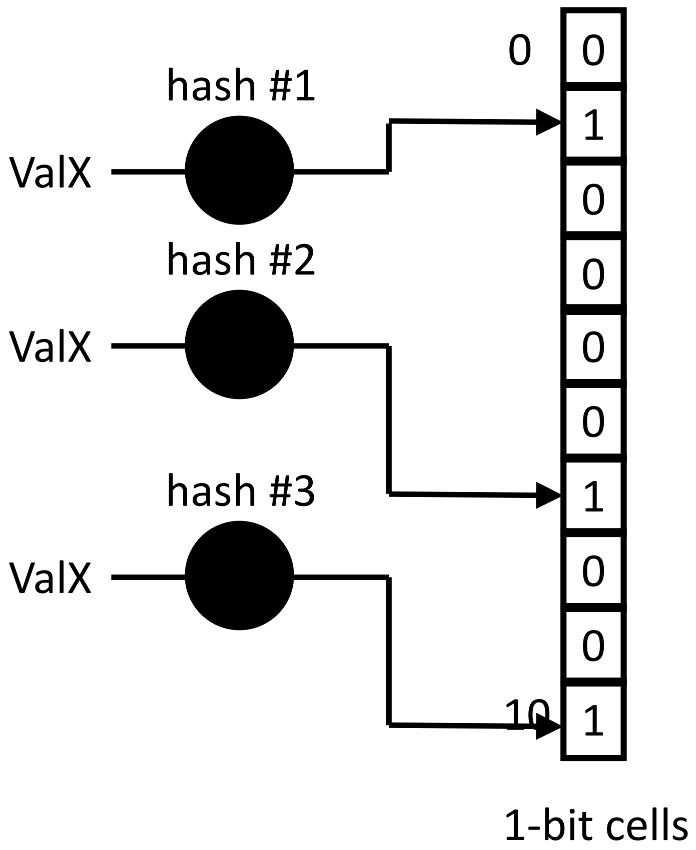


1-bit cells

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

How to implement a set
2nd approach – with probabilistic output

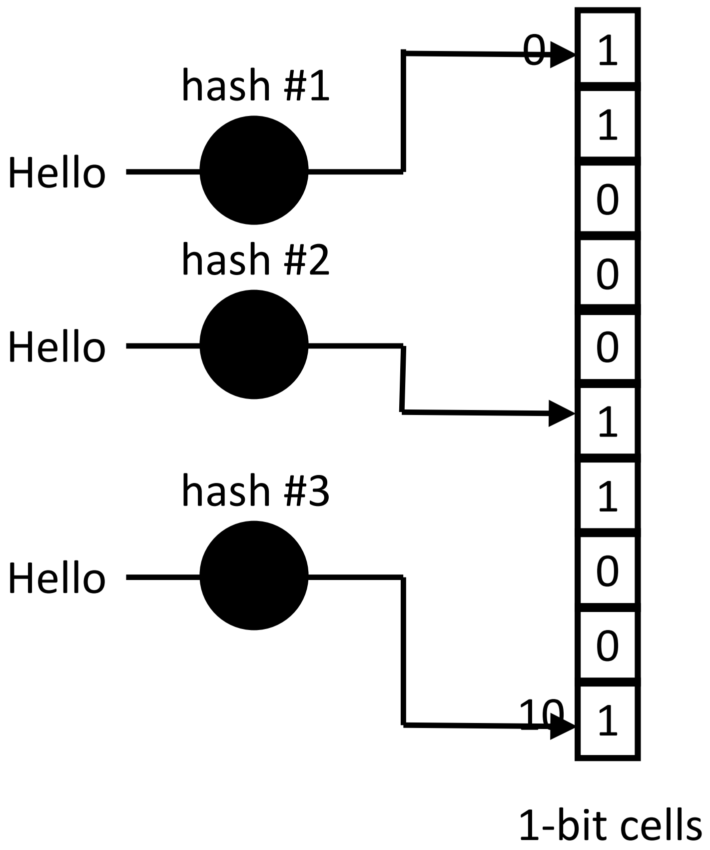
insert ValX



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

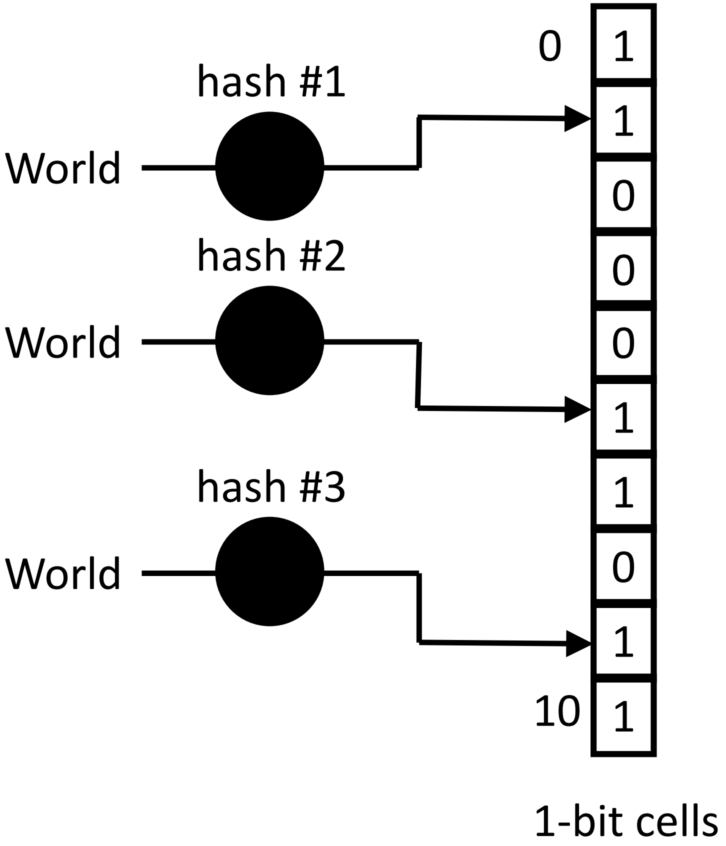
How to implement a set
2nd approach – with probabilistic output

insert Hello



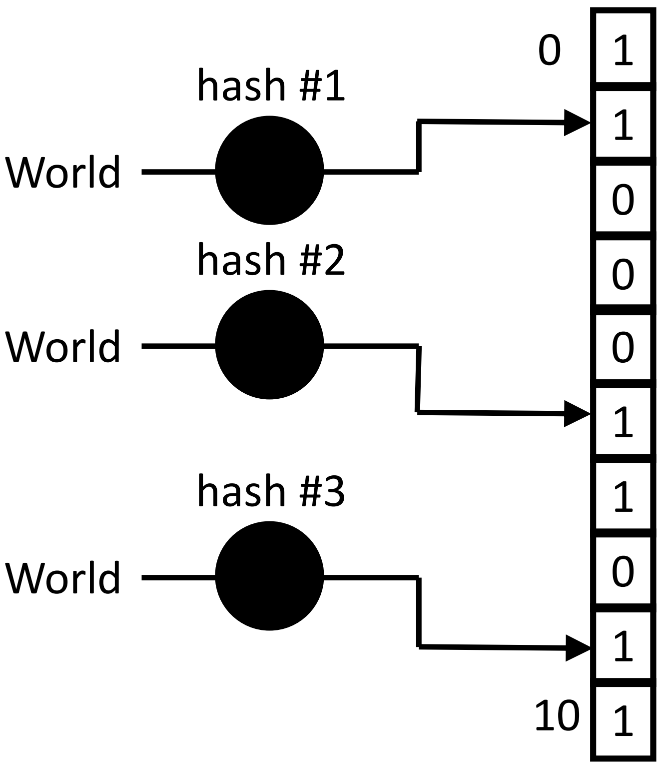
How to implement a set
2nd approach – with probabilistic output

insert World



How to implement a set
2nd approach – with probabilistic output

insert World



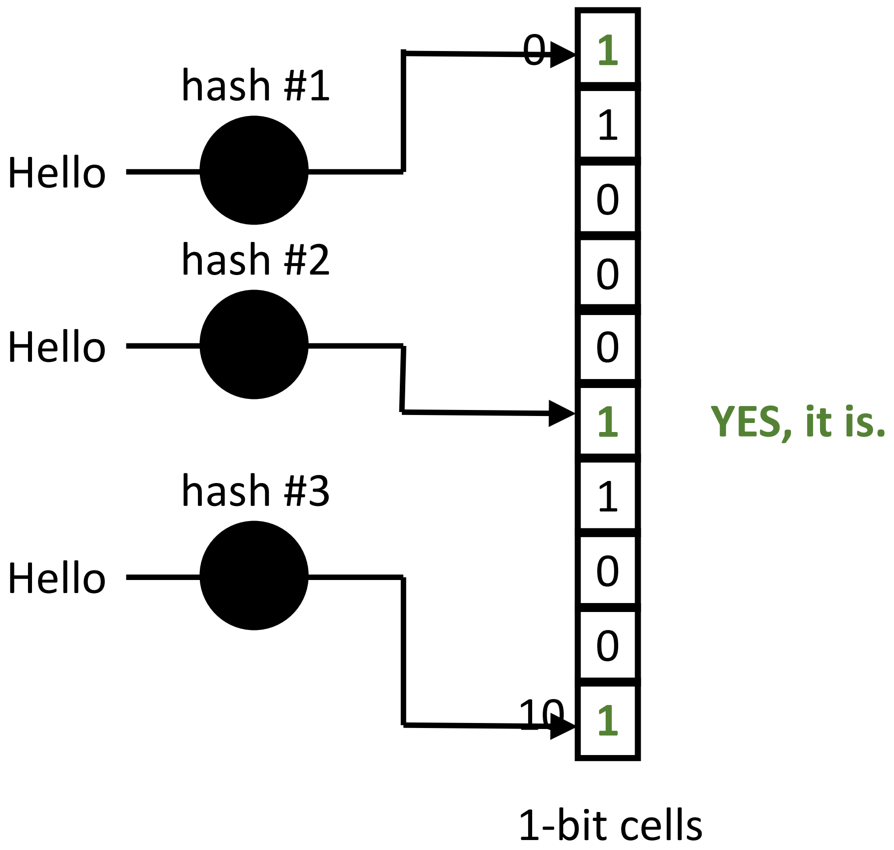
1-bit cells

An element is considered in the set if **all** the hash values map **to a cell with 1**

An element is not in the set if **at least** one hash value maps **to a cell with 0**

How to implement a set
2nd approach – with probabilistic output

is Hello in the set?

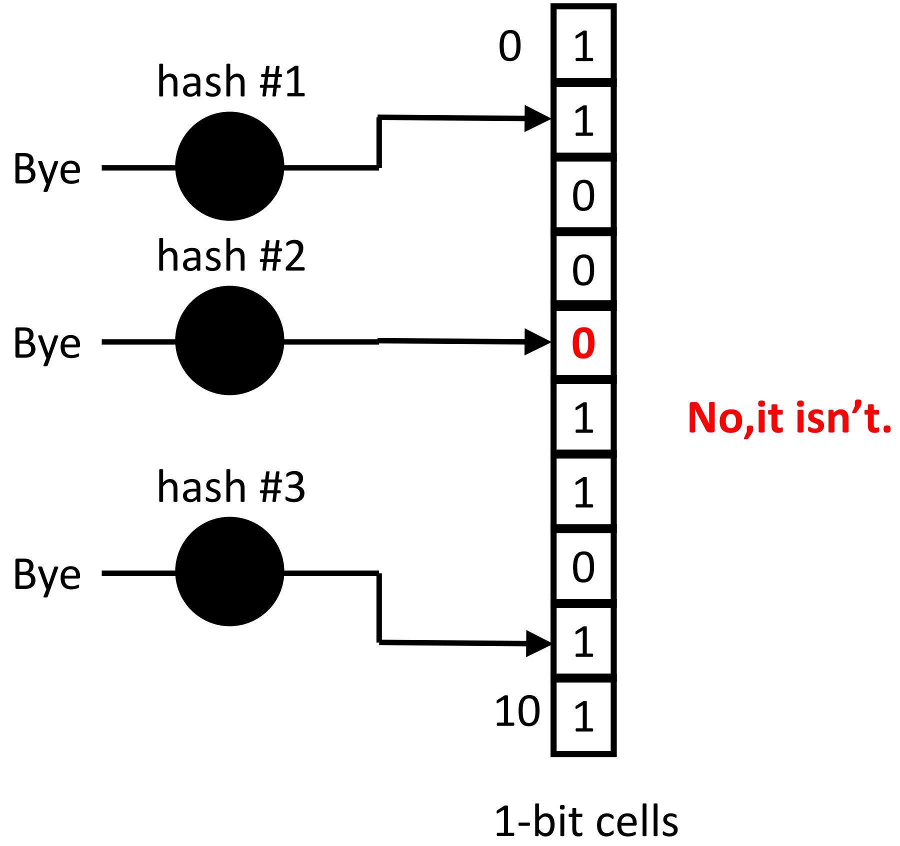


An element is considered in the set if **all** the hash values map **to a cell with 1**

An element is not in the set if **at least** one hash value maps **to a cell with 0**

How to implement a set
2nd approach – with probabilistic output

Is Bye in the set?



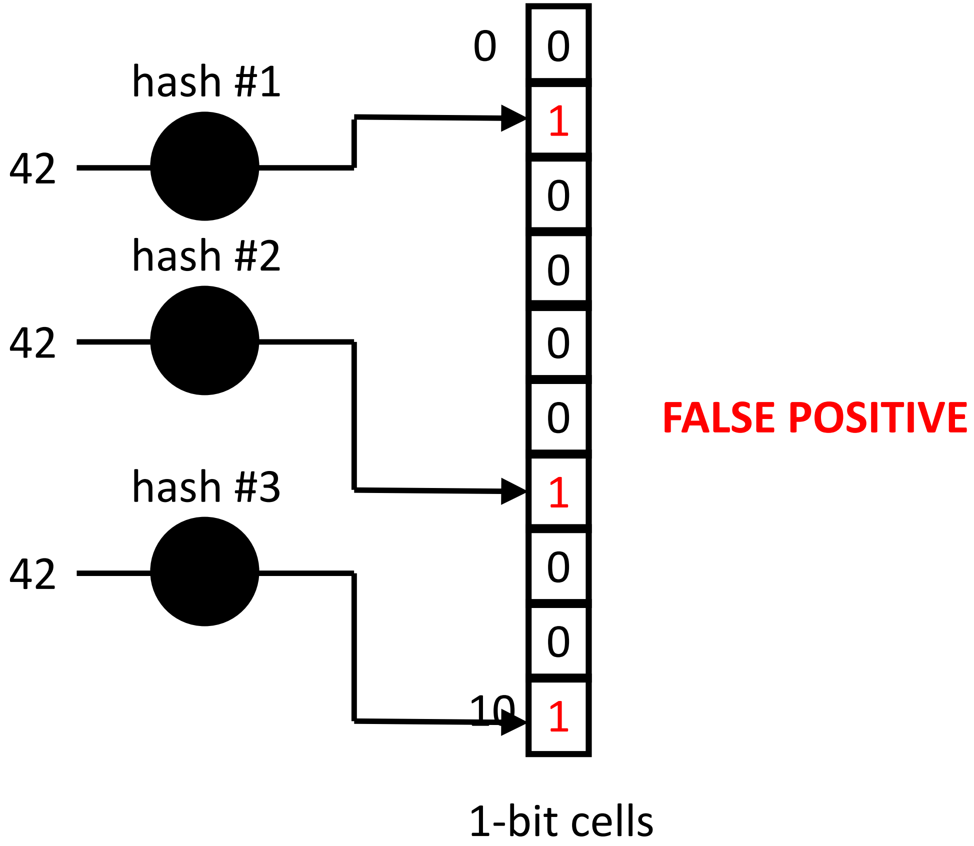
An element is considered in the set if **all** the hash values map **to a cell with 1**

An element is not in the set if **at least** one hash value maps **to a cell with 0**

Slides were inspired by (and are based on) related courses of Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich), Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

How to implement a set
2nd approach – with probabilistic output

Is 42 in the set?

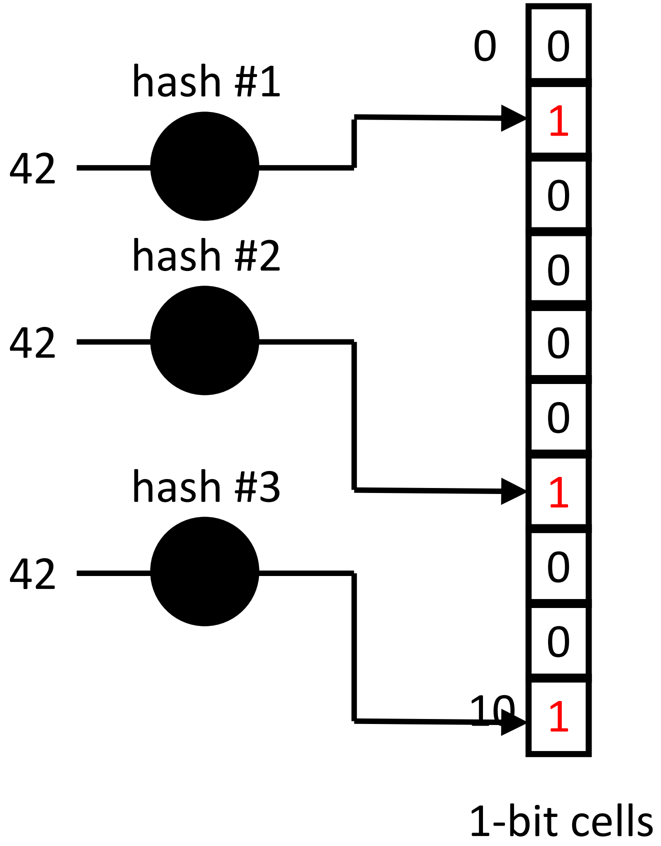


An element is considered in the set if **all** the hash values map **to a cell with 1**

An element is not in the set if **at least** one hash value maps **to a cell with 0**

How to implement a set
2nd approach – with probabilistic output

Is 42 in the set?



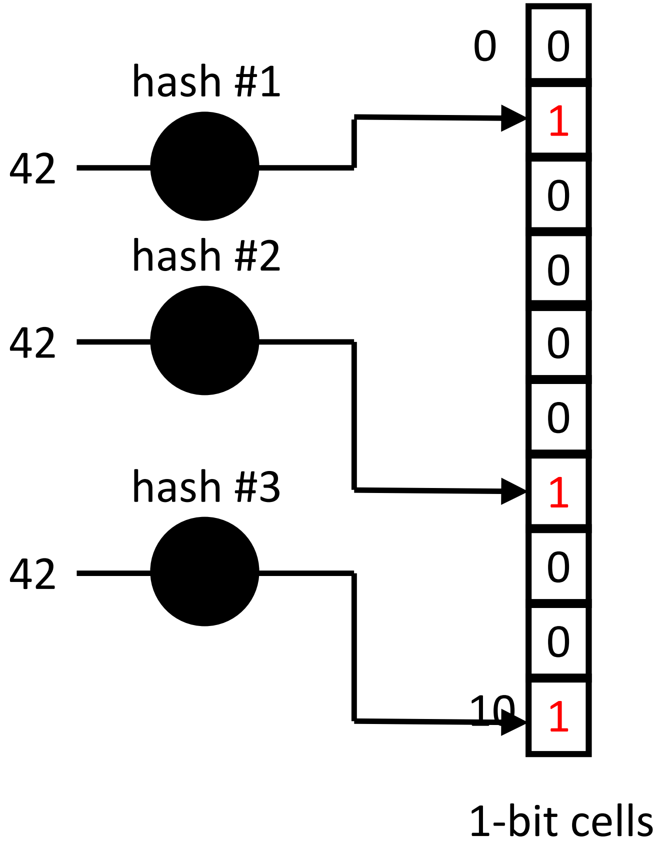
FALSE POSITIVE

N elements, M cells and K hash functions

probability of an element to be mapped into a particular cell	**1/M**
probability of an element not to be mapped into a particular cell	**1-1/M**
probability of a cell to be 0	$(1-1/M)^{KN}$
false positive rate	$(1-(1-1/M)^{KN})^{K}$
false negative rate	**0**

How to implement a set
2nd approach – with probabilistic output

Is 42 in the set?



N	M	K	FPR
1000	10000	7	0.82%
1000	100000	7	~0%

Pros:
10x less memory usage than the simple approach

Cons:
slightly more operations required (e.g. 7 instead of 1)

Dimension your Bloom Filter

- N elements
- M cells
- K hash functions
- FP false positive rate

Dimension your Bloom Filter

- N elements
- M cells
- K hash functions
- FP false positive rate

asymptotic approx.

$$FP = (1 - (1 - \frac{1}{M})^{KN})^K \approx (1 - e^{-KN/M})^K$$

with calculus you can dimension your bloom filter

Dimension your Bloom Filter

N = 1000

M = 10000

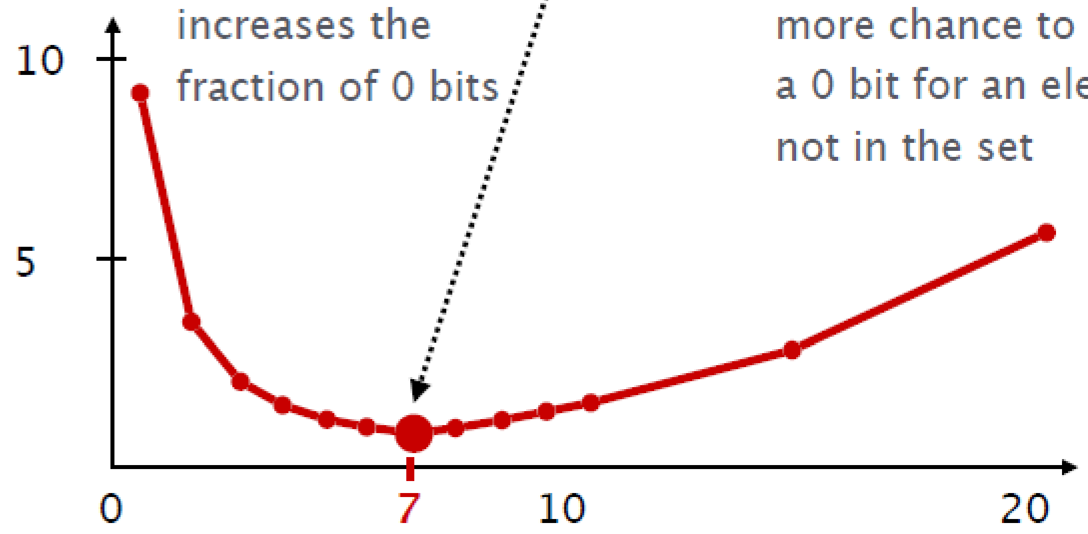
K hash functions

FP false positive rate

there is always a global minimum when $K = \ln 2 * (M/N)$ found by taking the derivative of $\approx (1 - e^{-KN/M})^K$

increases the fraction of 0 bits

more chance to find a 0 bit for an element not in the set



Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

Implementation of a Bloom Filter in P4₁₆

You will have to use hash functions

v1model

```
enum HashAlgorithm {
    crc32,
    crc32_custom,
    crc16,
    s,
    random,
    identity,
    csum16,
    xor16
}
```

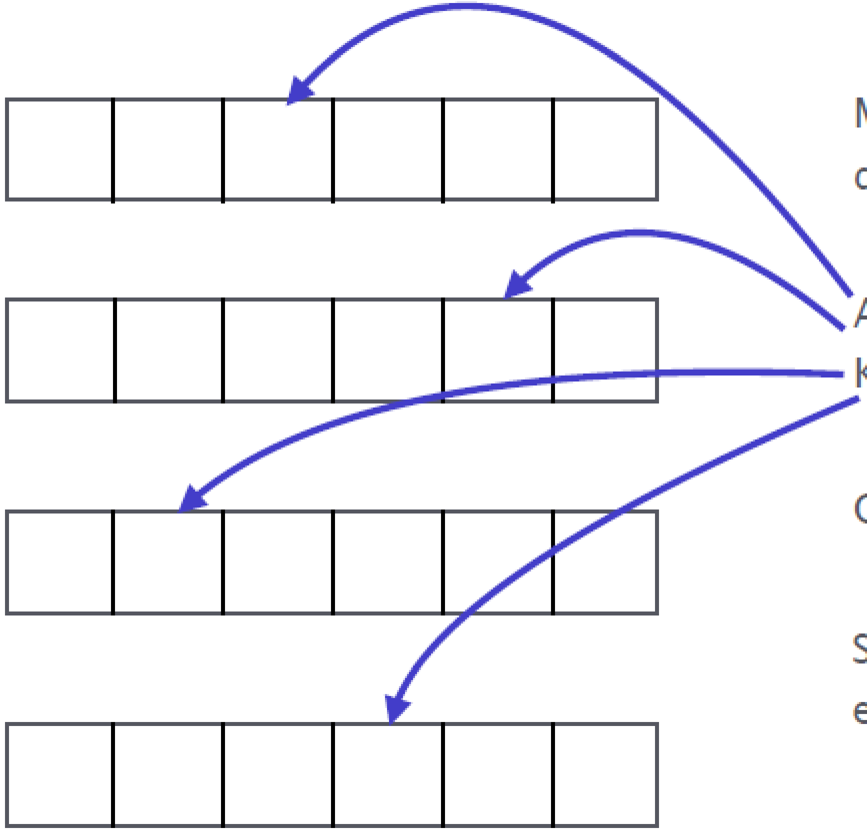
```
extern register<T> {

    register(bit<32> size);

    void read(out T result, in bit<32> index);
    void write(in bit<32> index, in T value);
}
```

```
extern void hash<O, T, D, M>(out O result,
  in HashAlgorithm algo, in T base, in D data, in M max);
```

Implementation in P4 with 2 hash functions

```
control MyIngress(…) {

  register register<bit<1>>(NB_CELLS) bloom_filter;

   apply {
      hash(meta.index1, HashAlgorithm.my_hash1, 0,
        {meta.dstPrefix, packet.ip.srcIP}, NB_CELLS);
      hash(meta.index2, HashAlgorithm.my_hash2, 0,
        {meta.dstPrefix, packet.ip.srcIP}, NB_CELLS);

      if (meta.to_insert == 1) {
        bloom_filter.write(meta.index1, 1);
        bloom_filter.write(meta.index2, 1);
      }

      if (meta.to_query == 1) {
        bloom_filter.read(meta.query1, meta.index1);
        bloom_filter.read(meta.query2, meta.index2);

        if (meta.query1 == 0 || meta.query2 == 0) {
          meta.is_stored = 0;
        }
        else {
          meta.is_stored = 1;
        }
      }
   }
}
```

Depending on the hardware limitations, splitting the bloom filter might be required



M cells are split into M/K disjoint groups

An element is hashed to K cells, one in each group

One hash function per group

Same performance, may be easier to implement or parallelize

Because deletions are not possible, the **controller** may need to regularly **reset** the bloom filters

Resetting a bloom filter takes some time during which it is not usable

Common trick: use two bloom filters and use one when the controller resets the other one

Why deletion is not easy?

Slides were inspired by (and are based on) related courses of
Nick McKeown (Stanford), Laurent Vanbever (ETH Zurich),
Jennifer Rexford (Princeton) and Noa Zilberman (Cambridge).

Solution

Counting Bloom Filters