DoS Protection

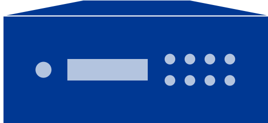
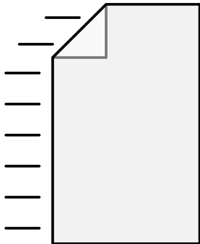
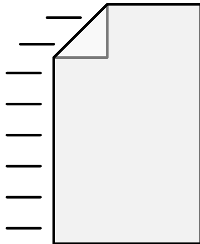
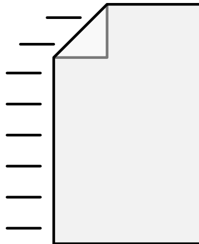
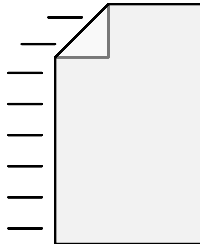
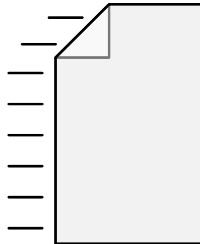
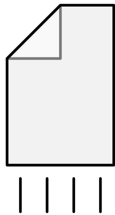
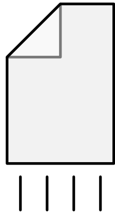
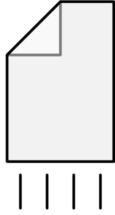
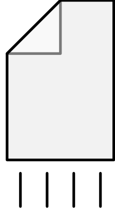
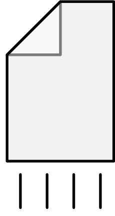
using multi-prefix query counting

Lukáš Ondráček · 27. 10. 2024

cz.nic CZ DOMAIN REGISTRY

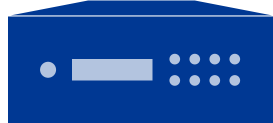
Overview

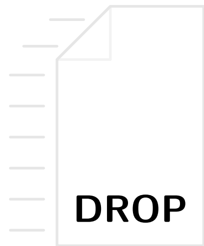
- **amplification att.**

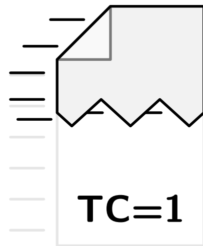












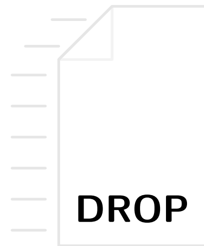
Overview

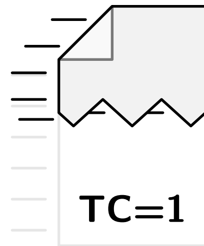
- amplification att.
  - **rate limiting**

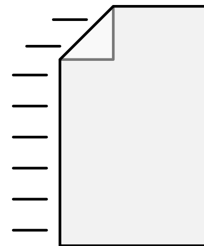


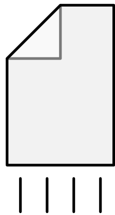


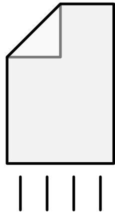


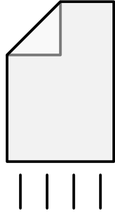


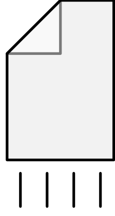


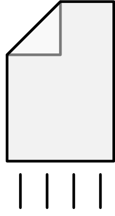






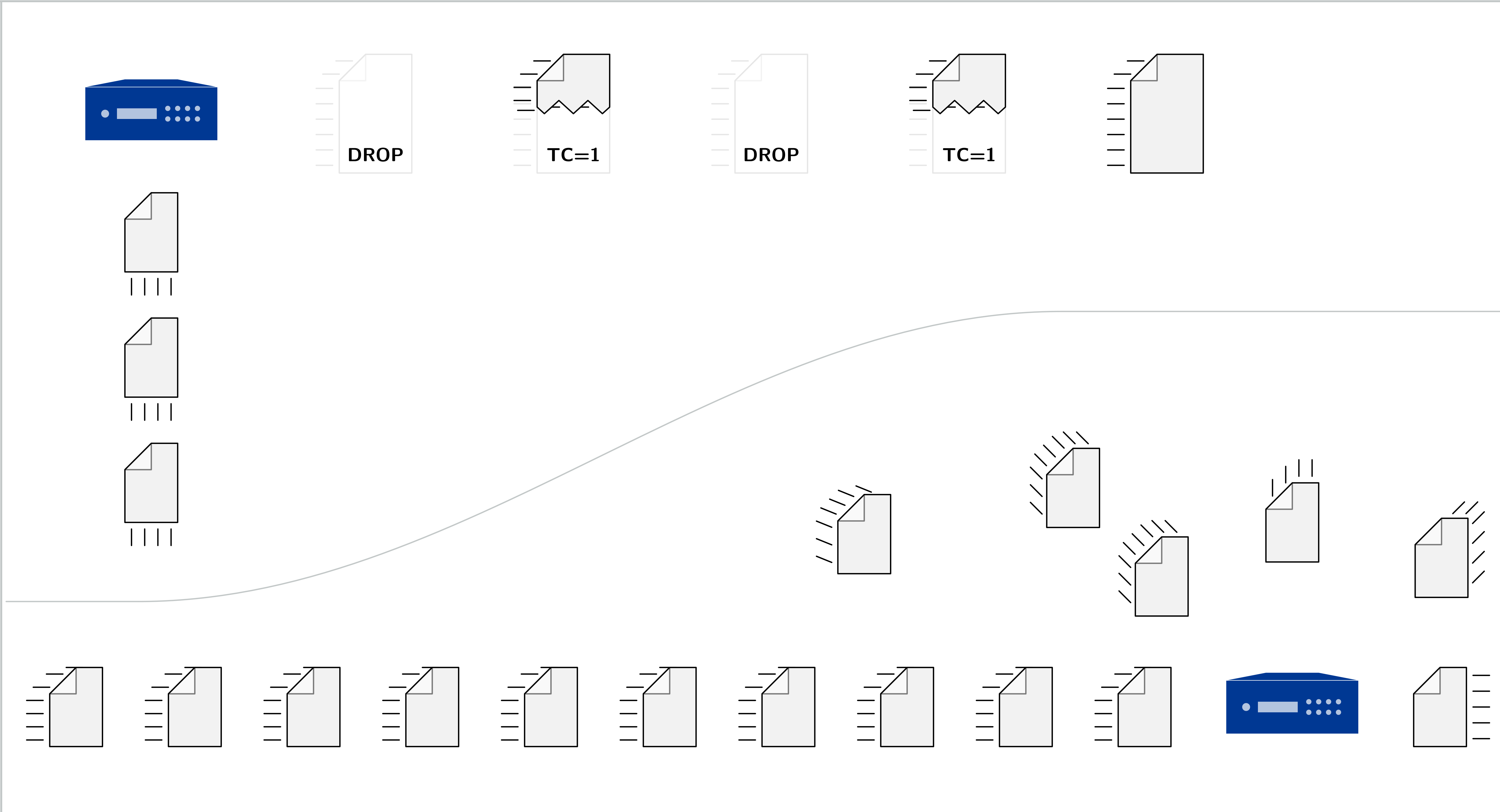






Overview

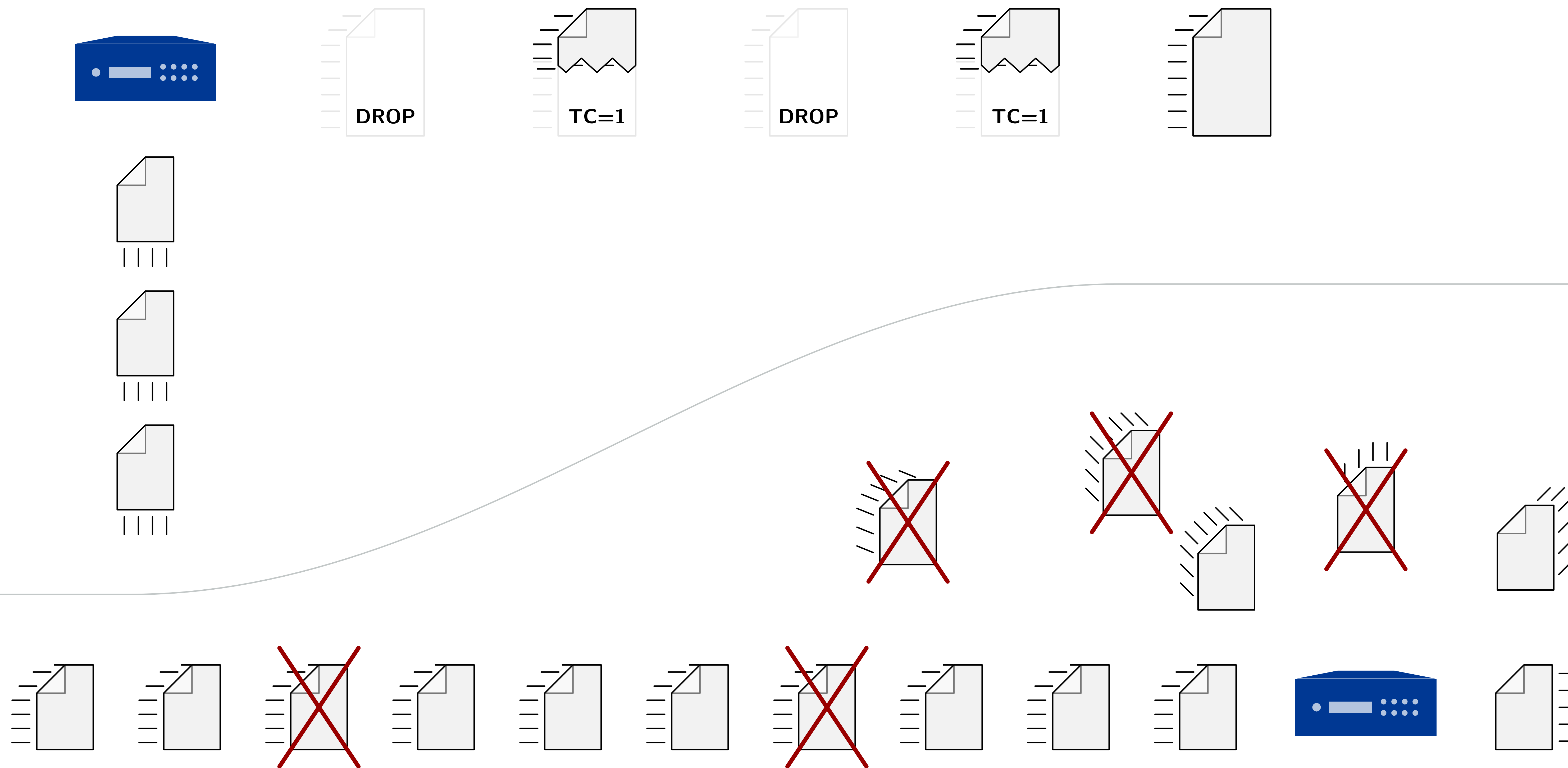
- amplification att.
  - rate limiting
- **server overload**



cz.nic | CZ DOMAIN REGISTRY

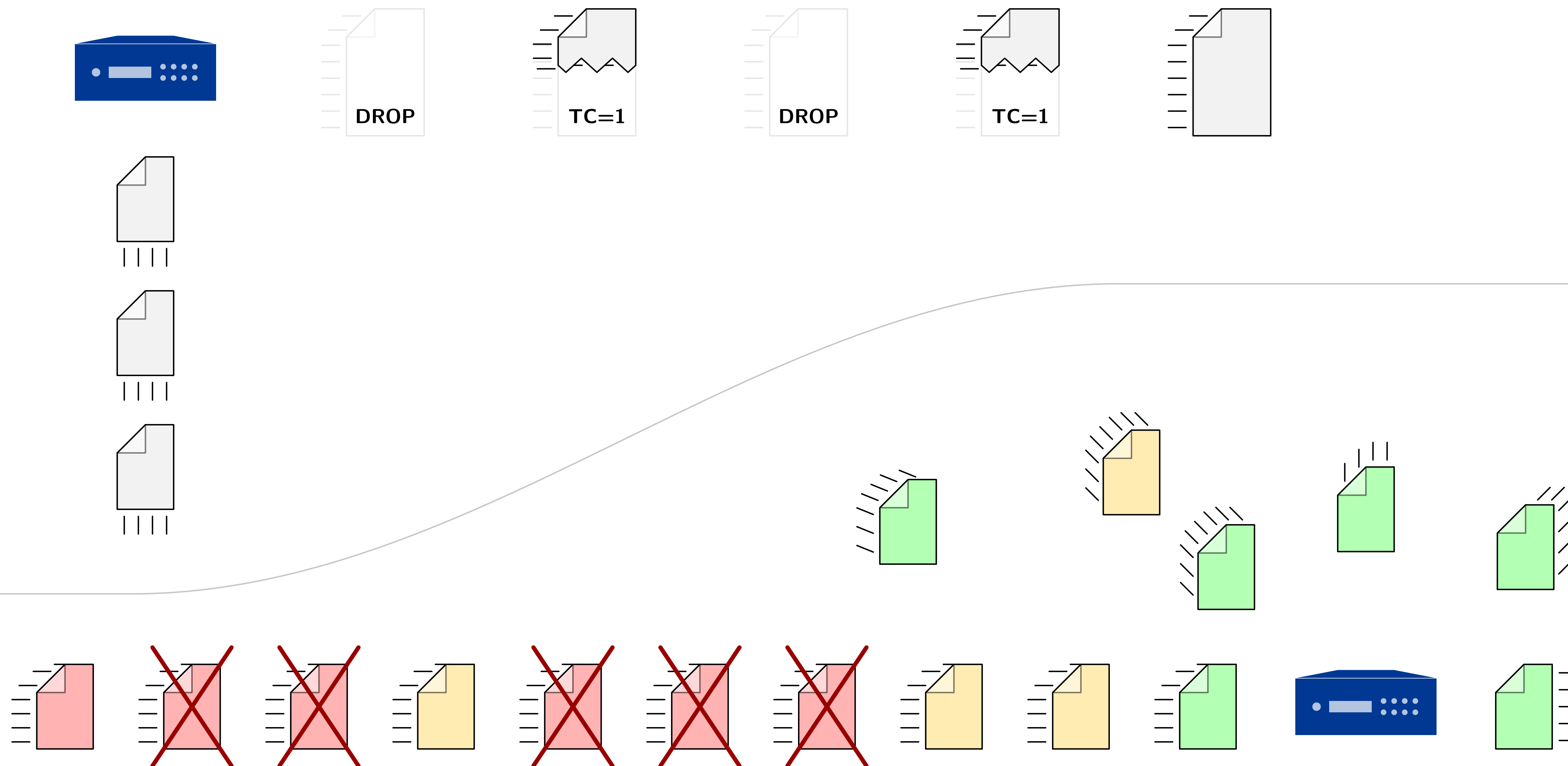
Overview

- amplification att.
  - rate limiting
- **server overload**



Overview

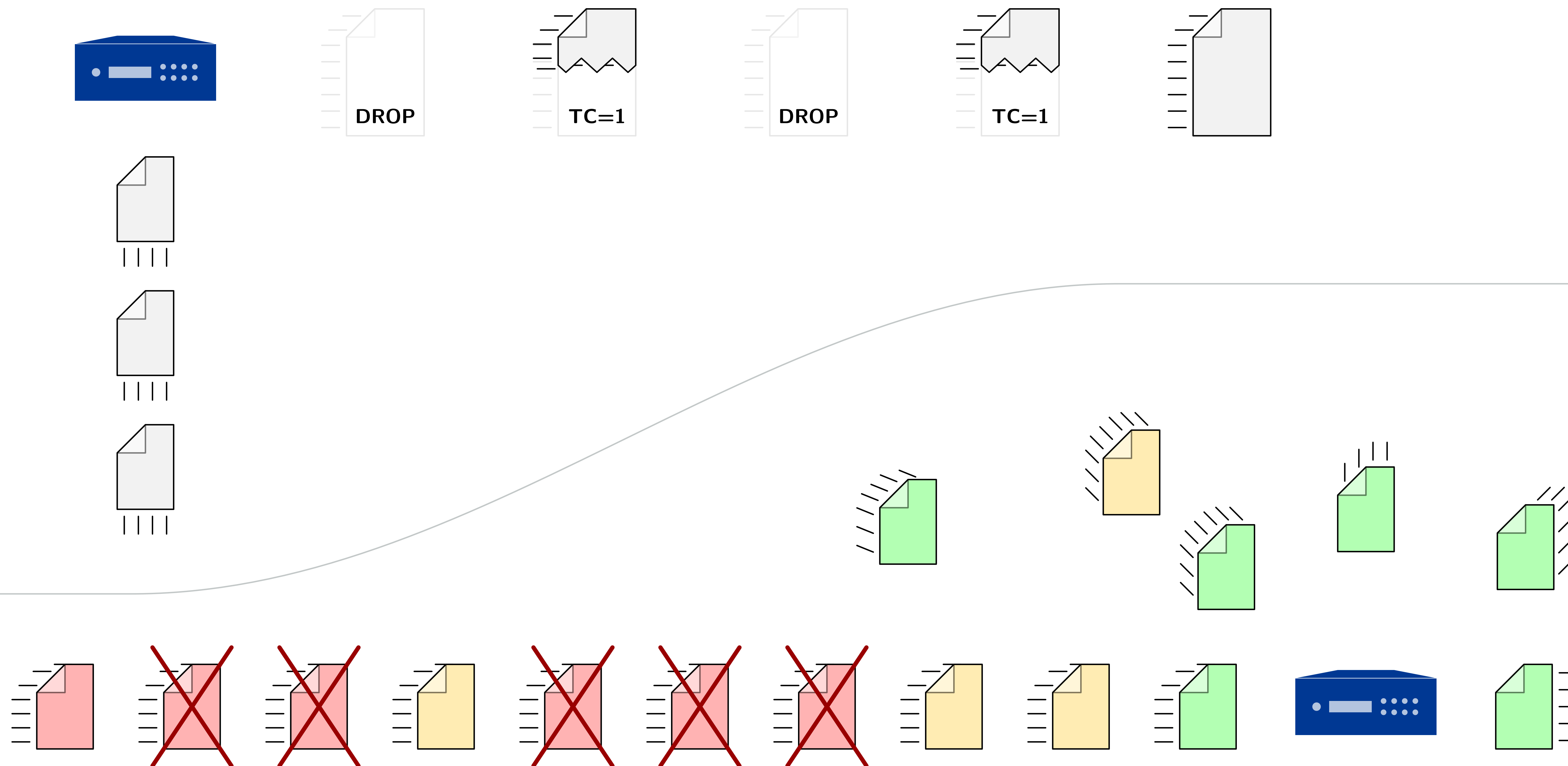
- amplification att.
  - rate limiting
- server overload
  - **prioritization**



cz.nic | CZ DOMAIN REGISTRY

Overview

- amplification att.
  - rate limiting
- server overload
  - prioritization



Limiting individual clients

- counters for addresses
  - instant limit L_I

⋮	
172.16.96.1	count in $[0, L_I)$
⋮	
2001:db8::734	count in $[0, L_I)$
⋮	

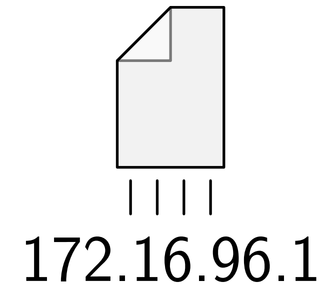
cz.nic | CZ DOMAIN REGISTRY

Limiting individual clients

- counters for addresses
  - instant limit L_I

⋮	
172.16.96.1	count in $[0, L_I)$
⋮	
2001:db8::734	count in $[0, L_I)$
⋮	

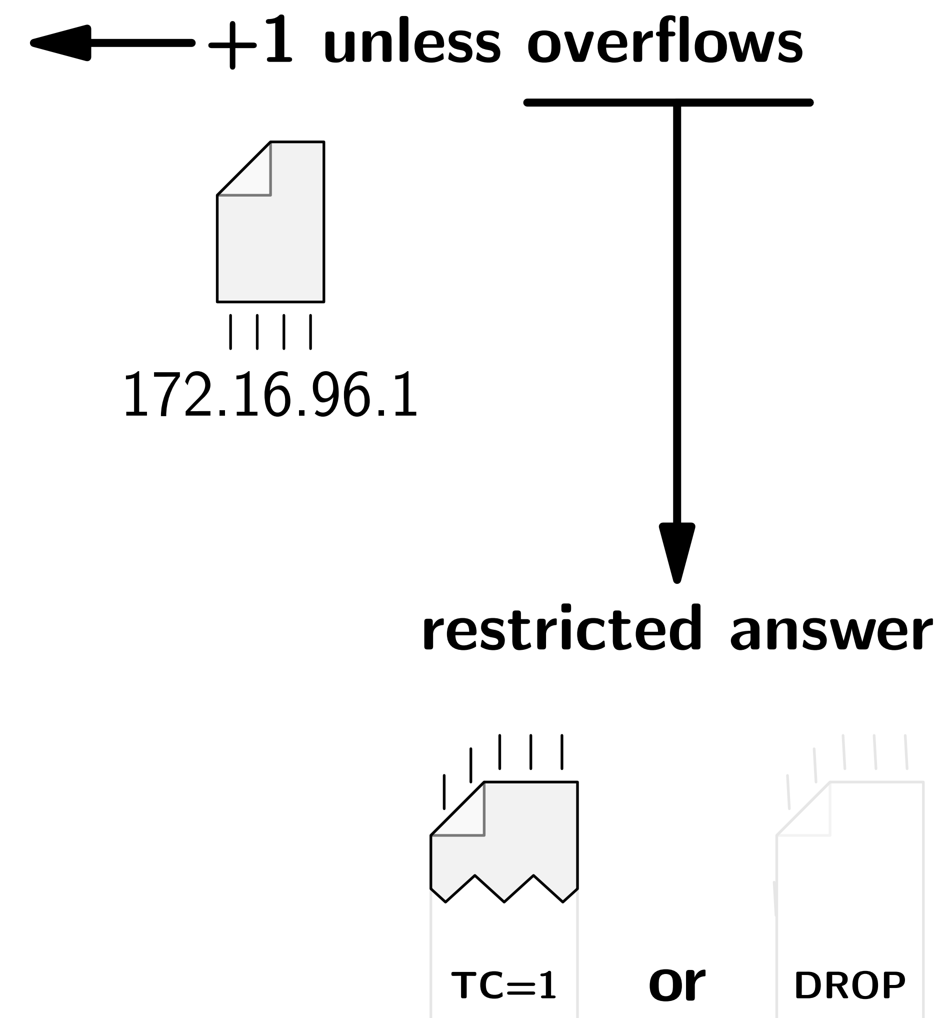
← +1 unless overflows



Limiting individual clients

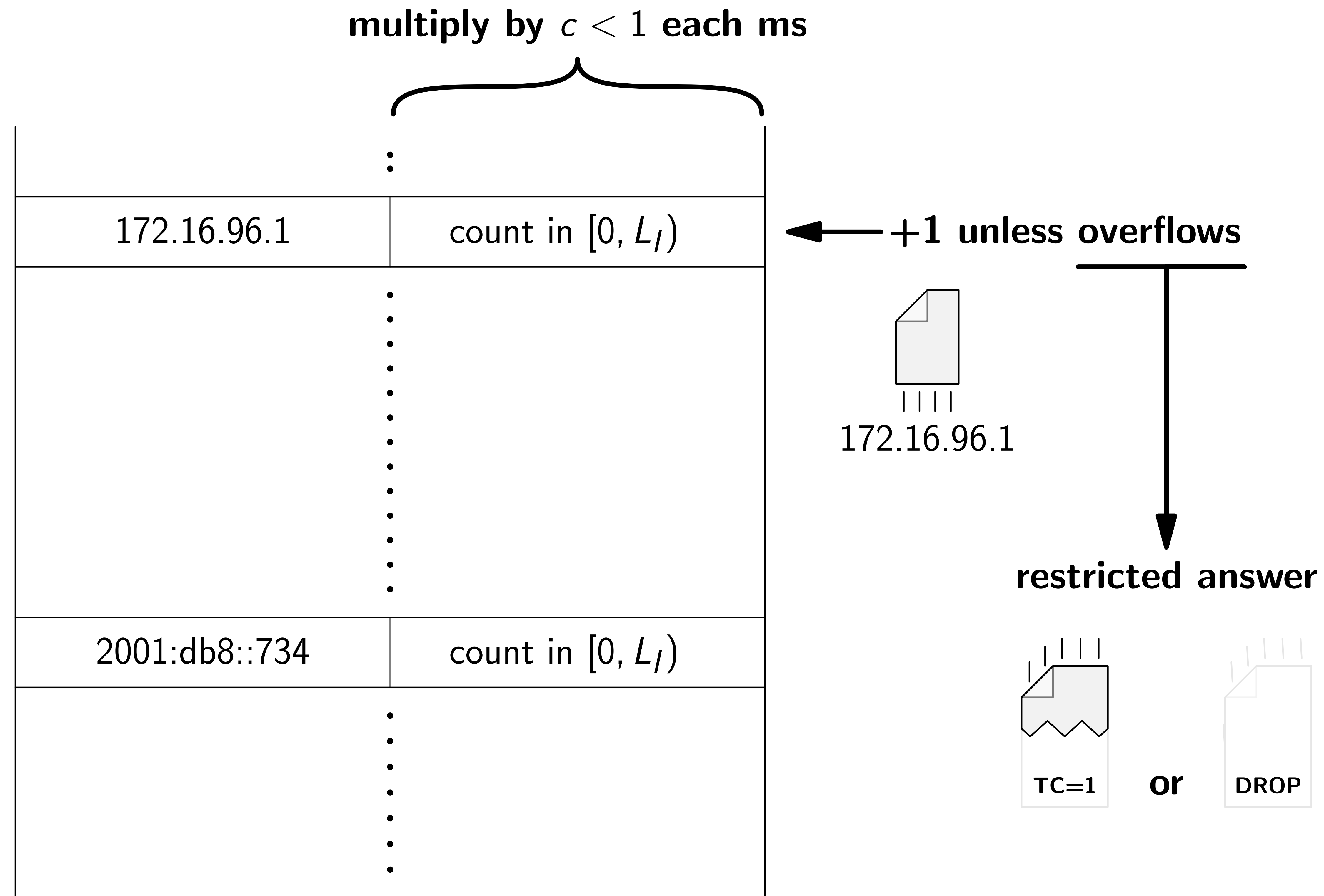
- **counters for addresses**
  - **instant limit L_I**

⋮	
172.16.96.1	count in $[0, L_I)$
⋮	
2001:db8::734	count in $[0, L_I)$
⋮	



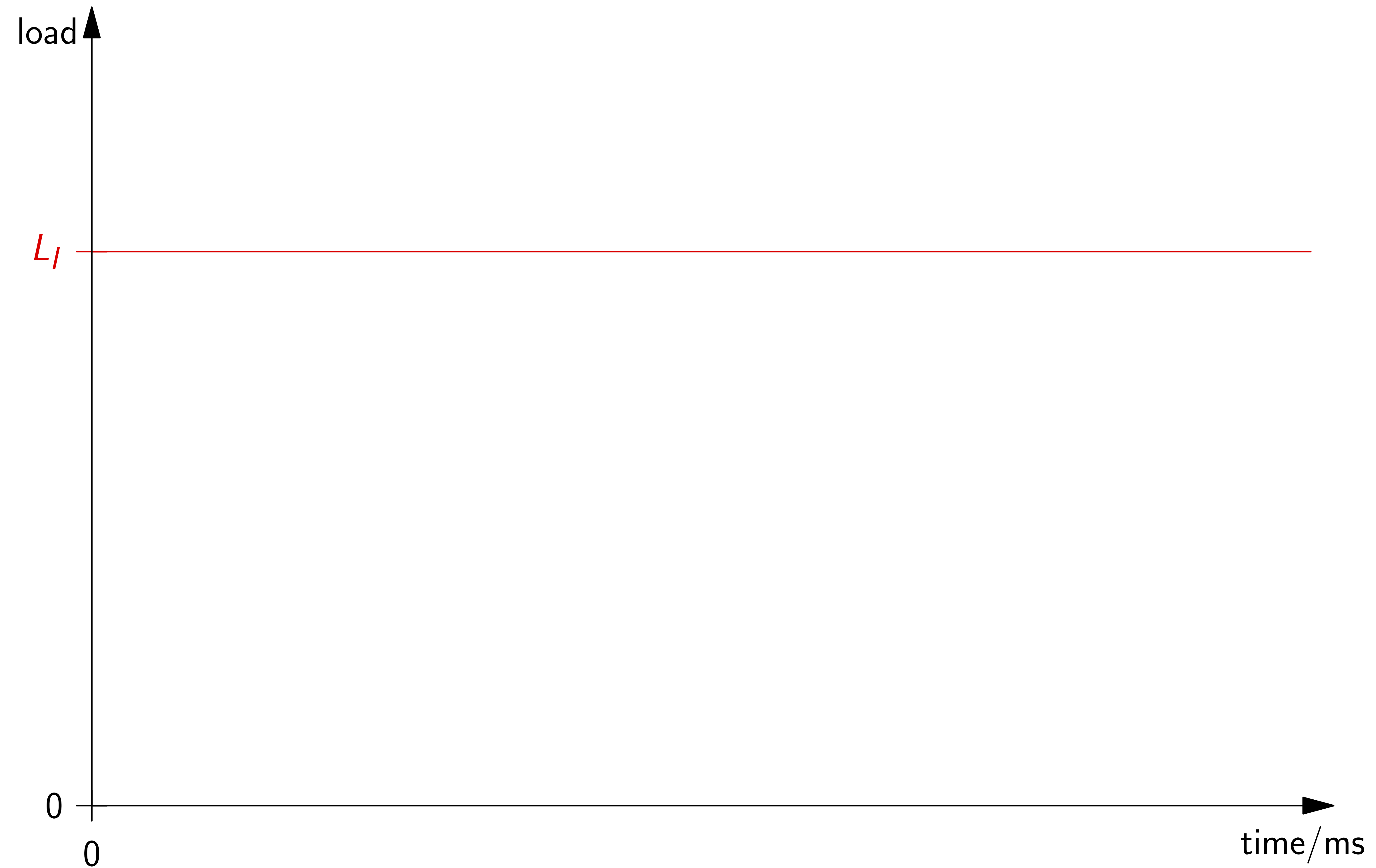
Limiting individual clients

- counters for addresses
  - instant limit L_I
- **exponential decay**
  - **rate limit**



Exponential decay

- **instant limit L_I**



cz.nic | CZ DOMAIN REGISTRY

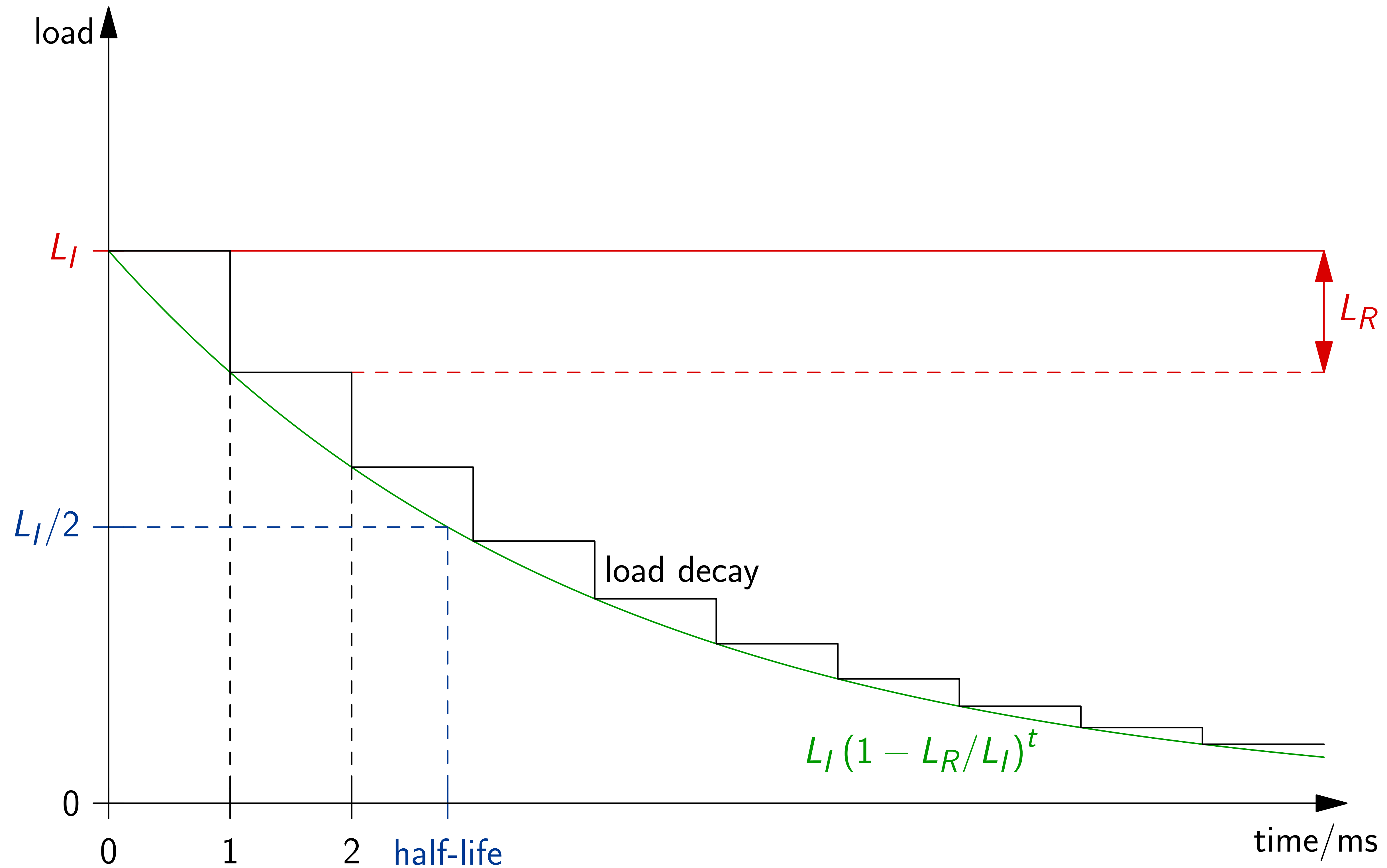
Exponential decay

- instant limit L_I
- **rate limit L_R**
  - **per ms**

cz.nic | CZ DOMAIN REGISTRY

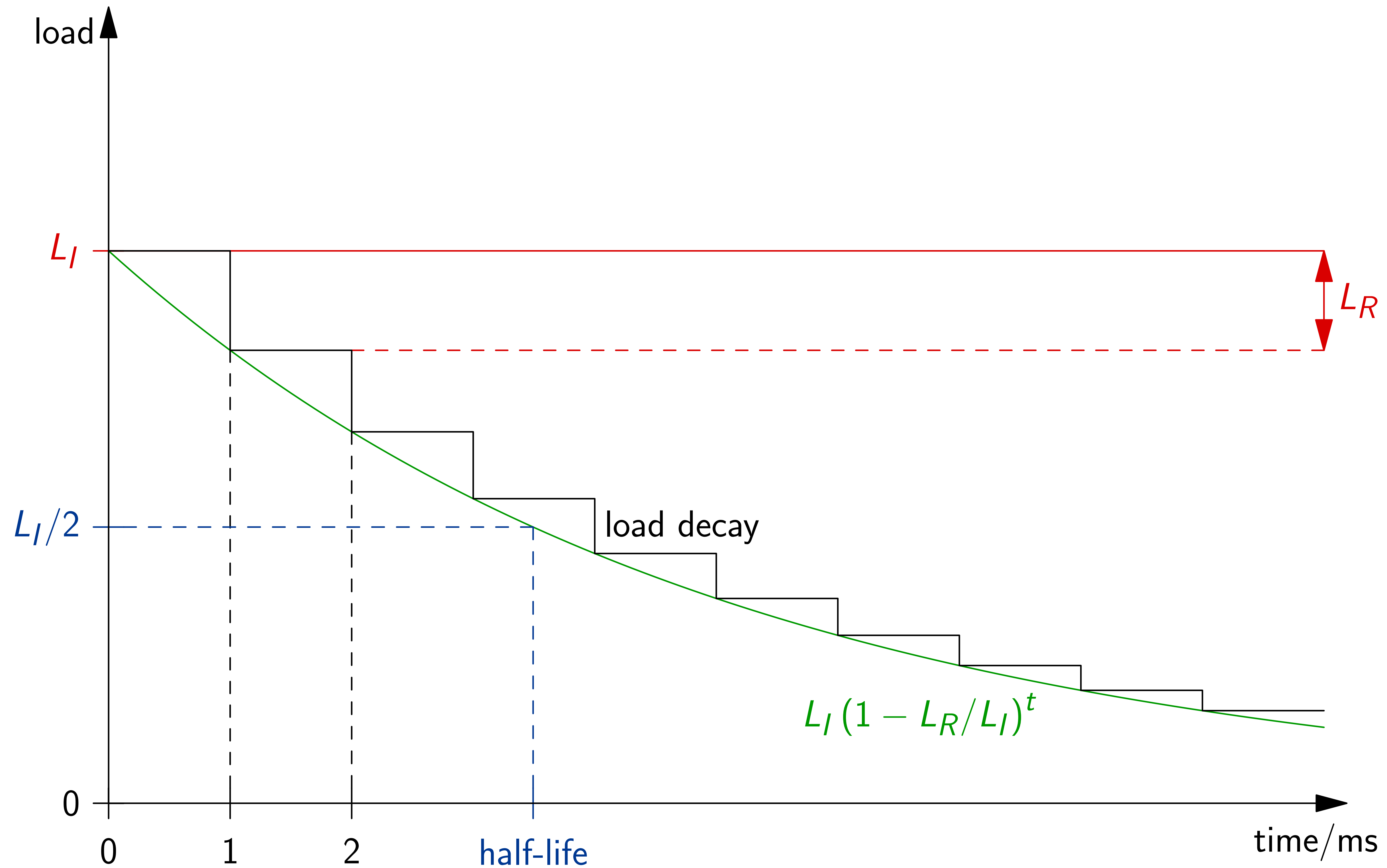
Exponential decay

- instant limit L_I
- rate limit L_R
  - per ms
- **half-life**



Exponential decay

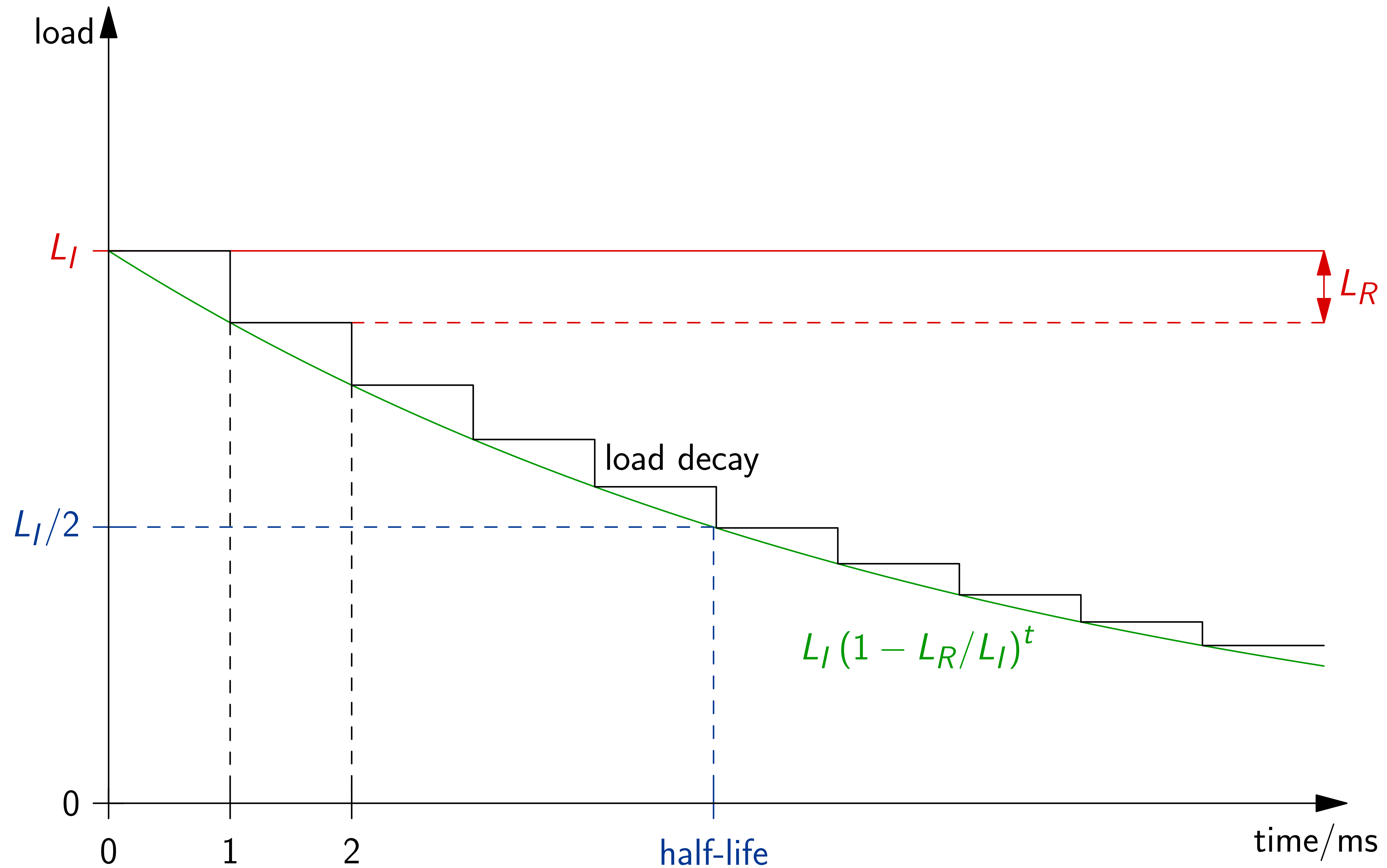
- instant limit L_I
- rate limit L_R
  - per ms
- half-life



cz.nic | CZ DOMAIN REGISTRY

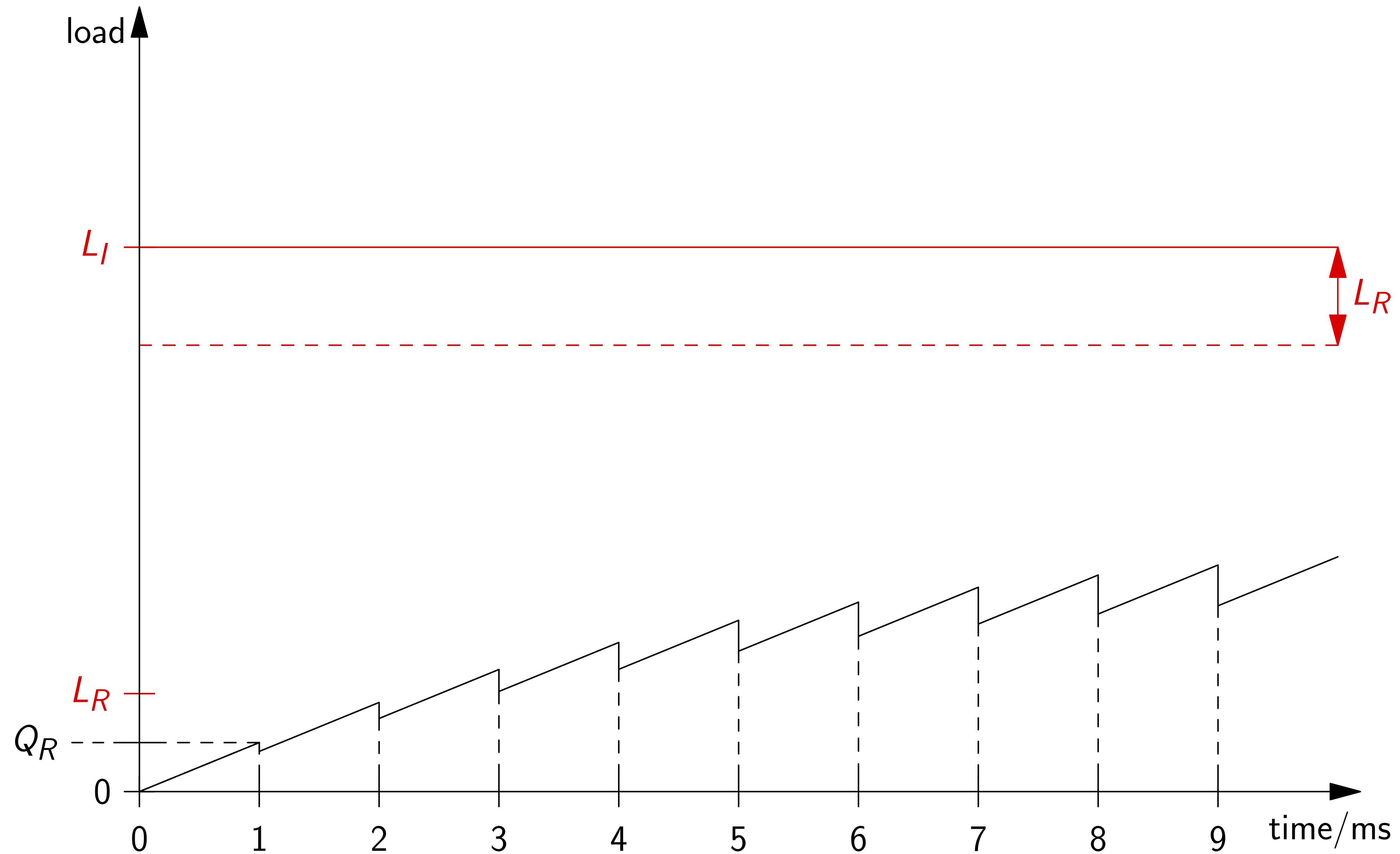
Exponential decay

- instant limit L_I
- rate limit L_R
  - per ms
- half-life



Constant query rate example

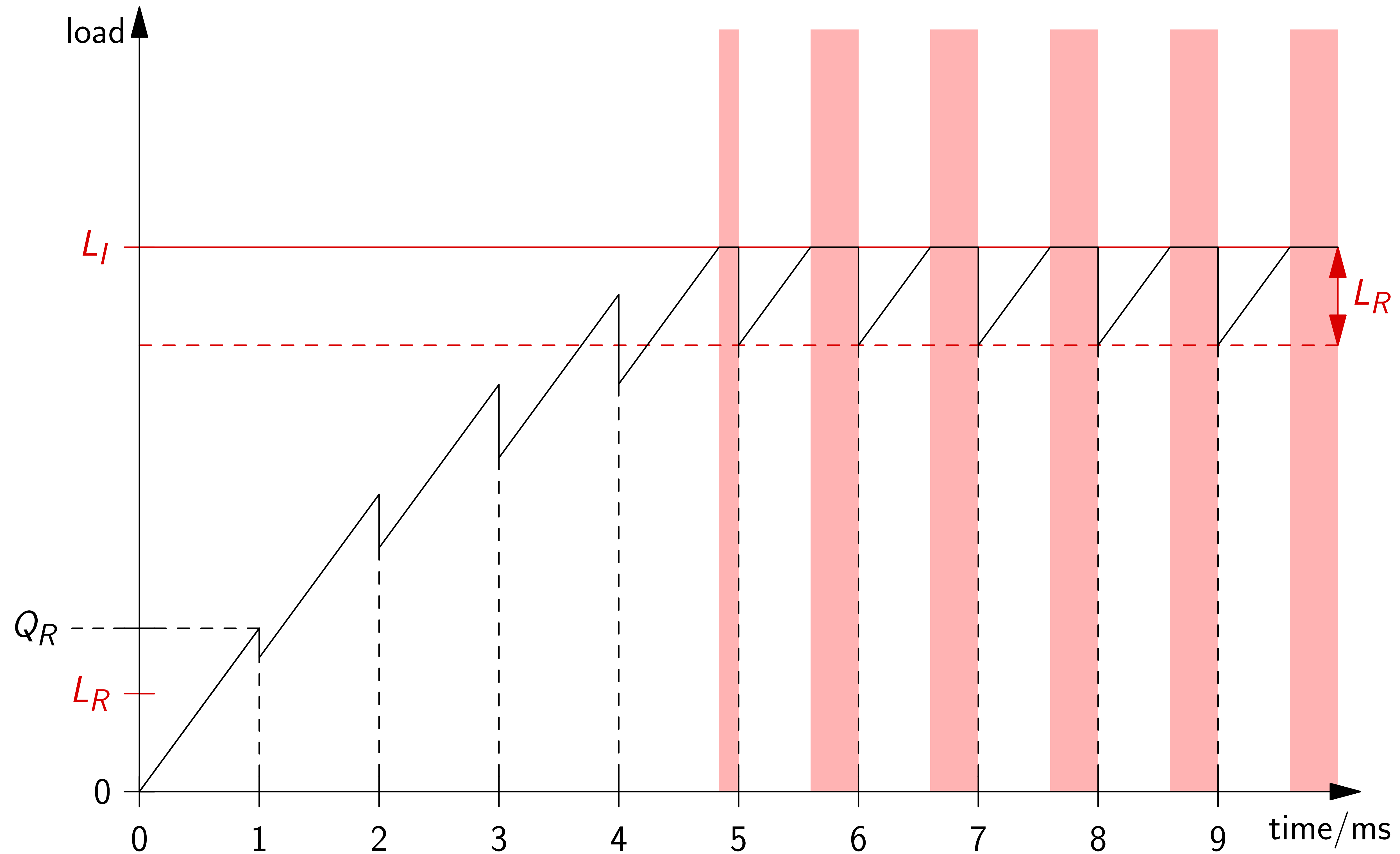
- instant limit L_I
- rate limit L_R
  - per ms
- **query rate Q_R**
  - **per ms**



cz.nic | CZ DOMAIN REGISTRY

Constant query rate example

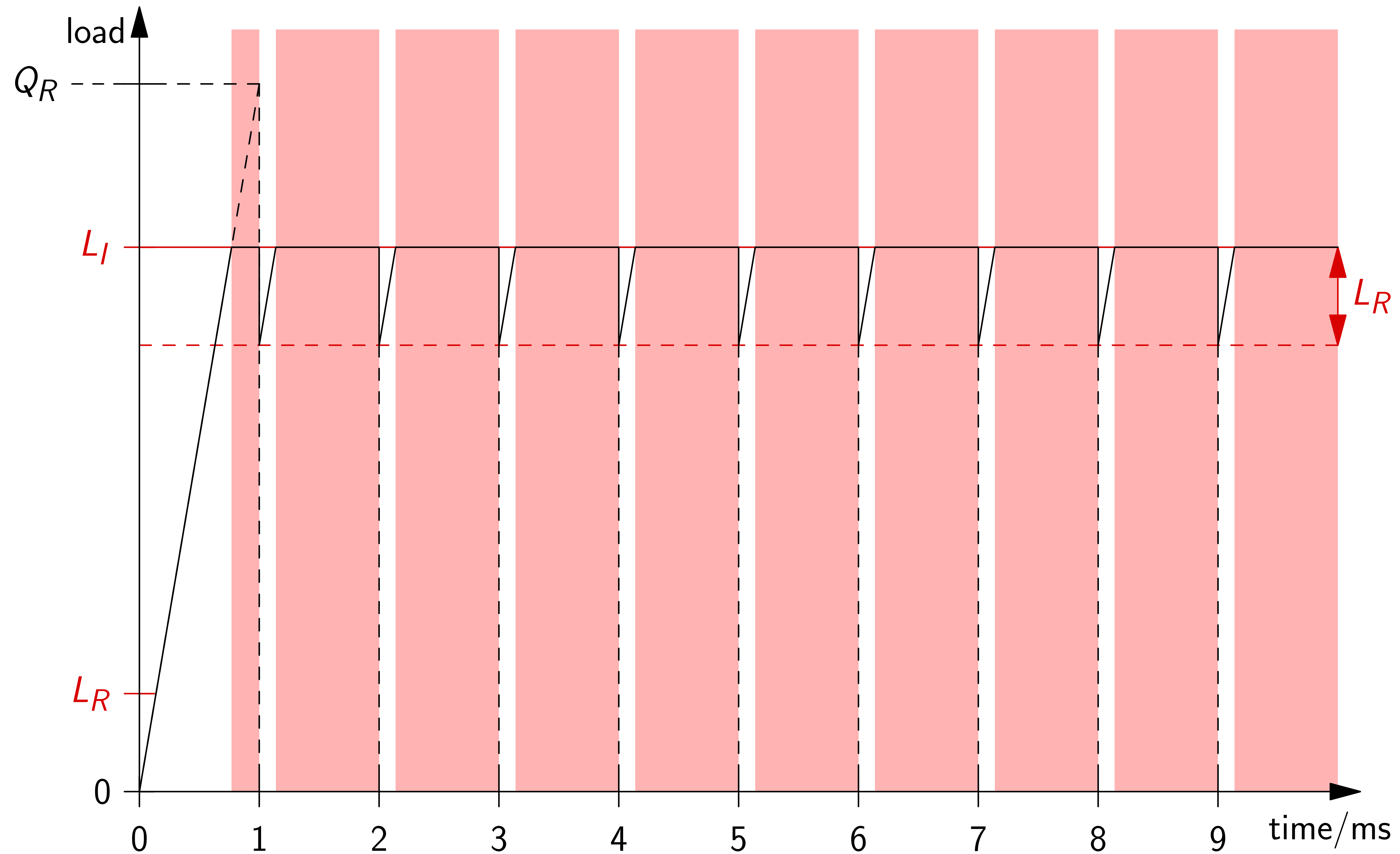
- instant limit L_I
- rate limit L_R
  - per ms
- query rate Q_R
  - per ms



cz.nic | CZ DOMAIN REGISTRY

Constant query rate example

- instant limit L_I
- rate limit L_R
  - per ms
- query rate Q_R
  - per ms



Limiting networks

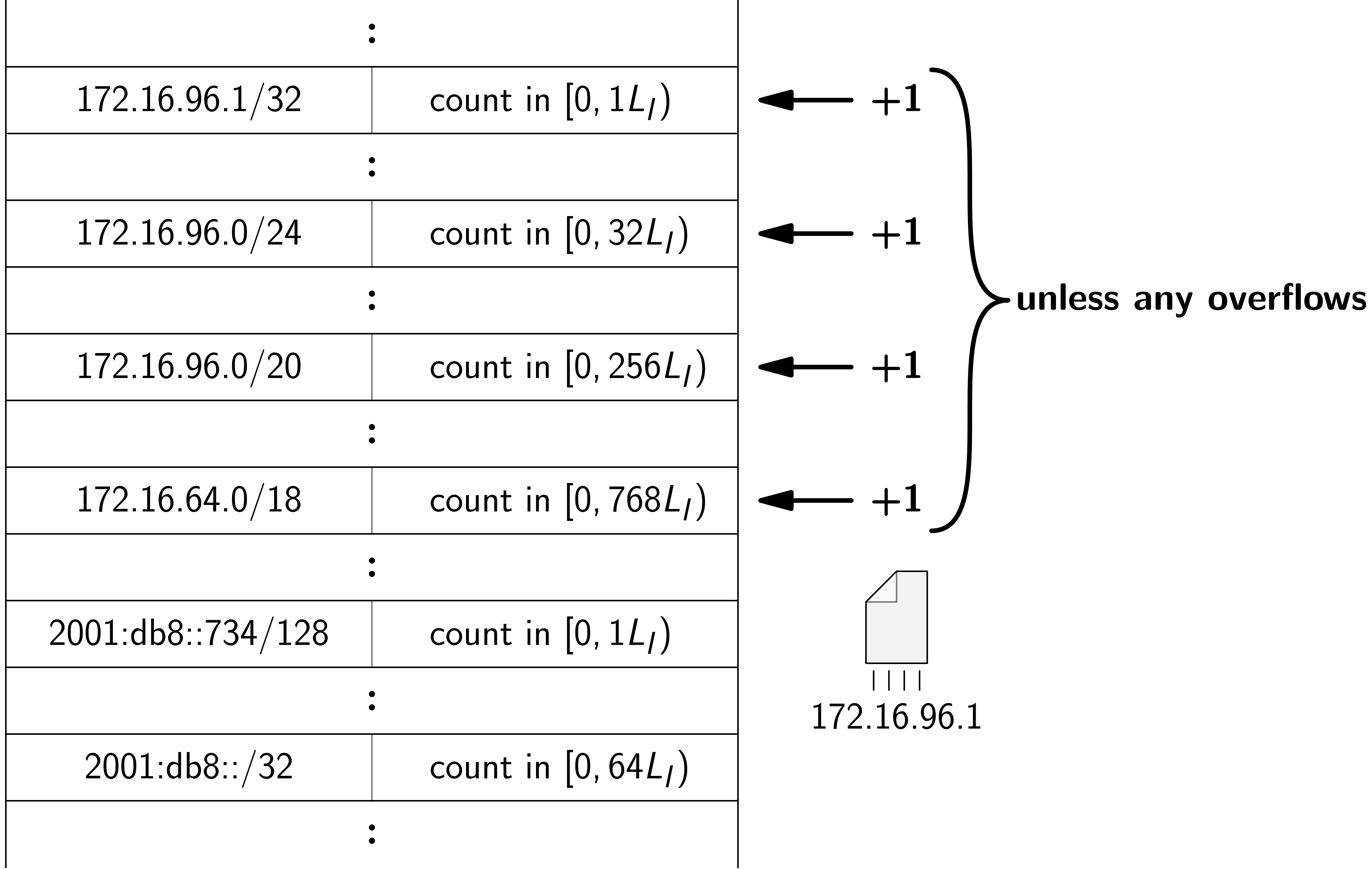
- IPv4
  - /32: 1
  - /24: 32
  - /20: 256
  - /18: 768
- IPv6
  - /128: 1
  - /64: 2
  - /56: 3
  - /48: 4
  - /32: 64

⋮	
172.16.96.1/**32**	count in $[0, \mathbf{1}L_l)$
⋮	
172.16.96.0/**24**	count in $[0, \mathbf{32}L_l)$
⋮	
172.16.96.0/**20**	count in $[0, \mathbf{256}L_l)$
⋮	
172.16.64.0/**18**	count in $[0, \mathbf{768}L_l)$
⋮	
2001:db8::734/**128**	count in $[0, \mathbf{1}L_l)$
⋮	
2001:db8::/**32**	count in $[0, \mathbf{64}L_l)$
⋮	

cz.nic | CZ DOMAIN REGISTRY

Limiting networks

- IPv4
  - /32: 1
  - /24: 32
  - /20: 256
  - /18: 768
- IPv6
  - /128: 1
  - /64: 2
  - /56: 3
  - /48: 4
  - /32: 64

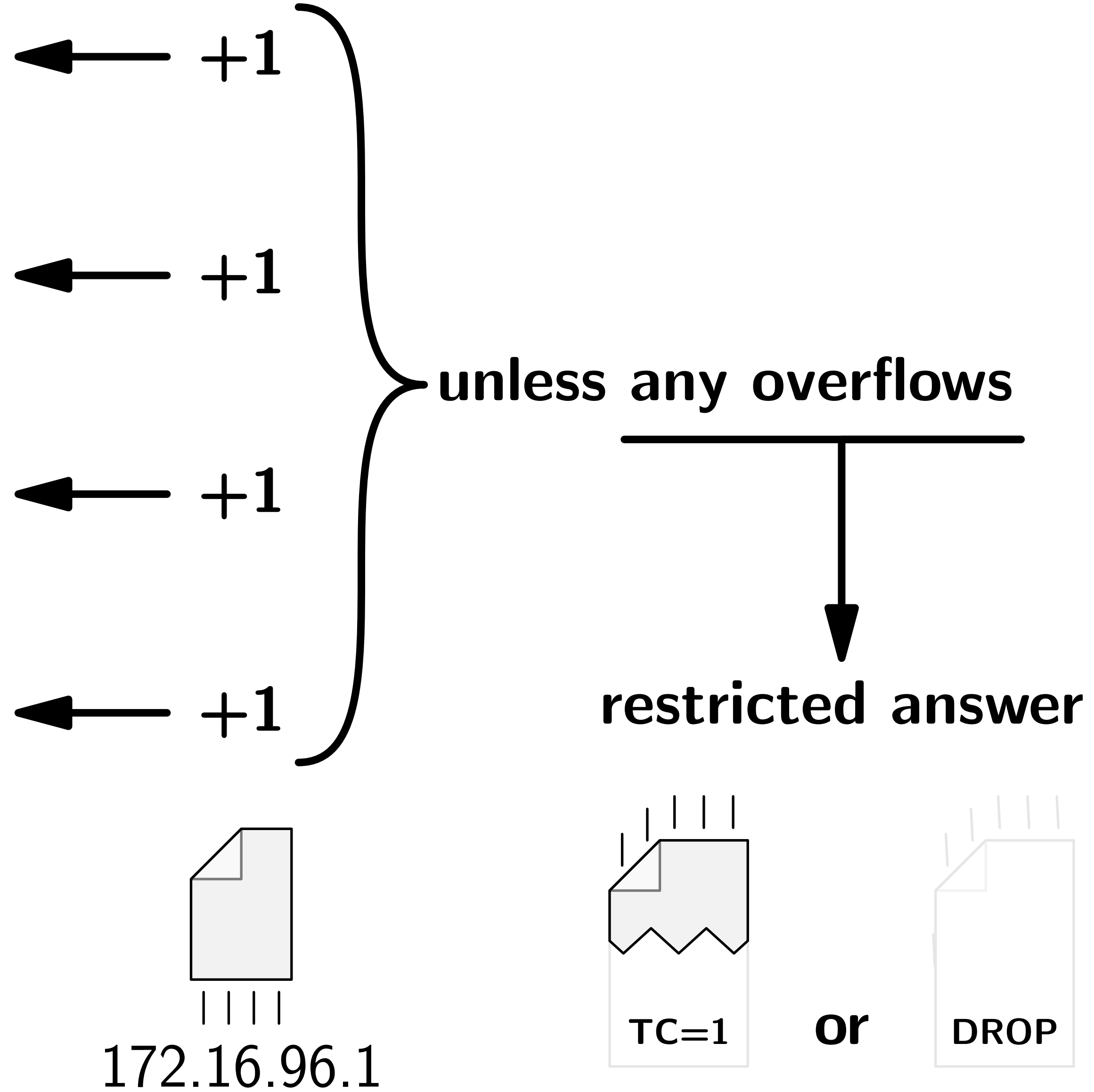


cz.nic CZ DOMAIN REGISTRY

Limiting networks

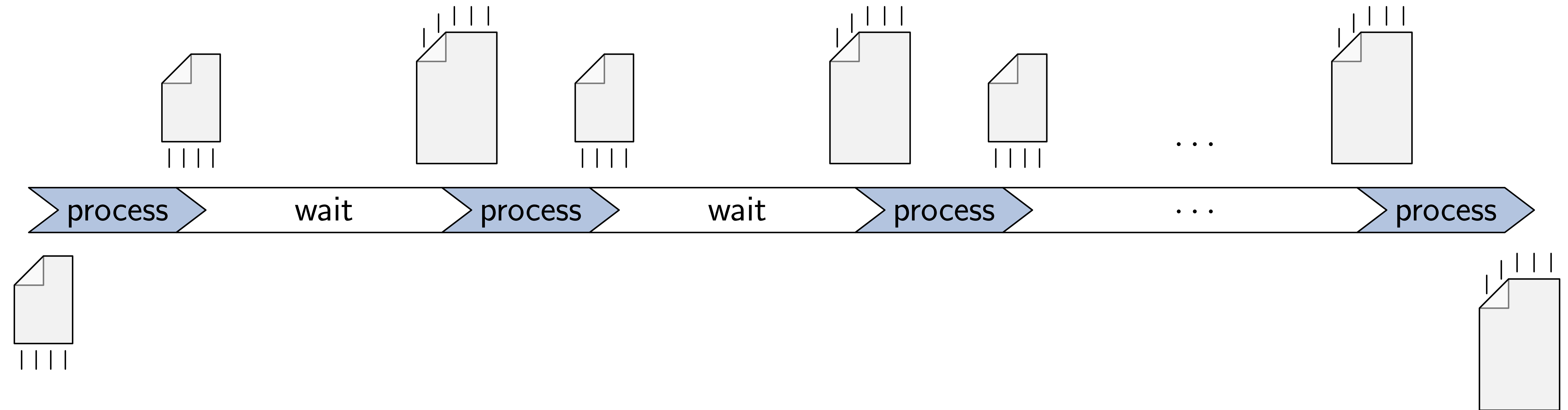
- IPv4
  - /32: 1
  - /24: 32
  - /20: 256
  - /18: 768
- IPv6
  - /128: 1
  - /64: 2
  - /56: 3
  - /48: 4
  - /32: 64

⋮	
172.16.96.1/32	count in $[0, 1L_l)$
⋮	
172.16.96.0/24	count in $[0, 32L_l)$
⋮	
172.16.96.0/20	count in $[0, 256L_l)$
⋮	
172.16.64.0/18	count in $[0, 768L_l)$
⋮	
2001:db8::734/128	count in $[0, 1L_l)$
⋮	
2001:db8::/32	count in $[0, 64L_l)$
⋮	



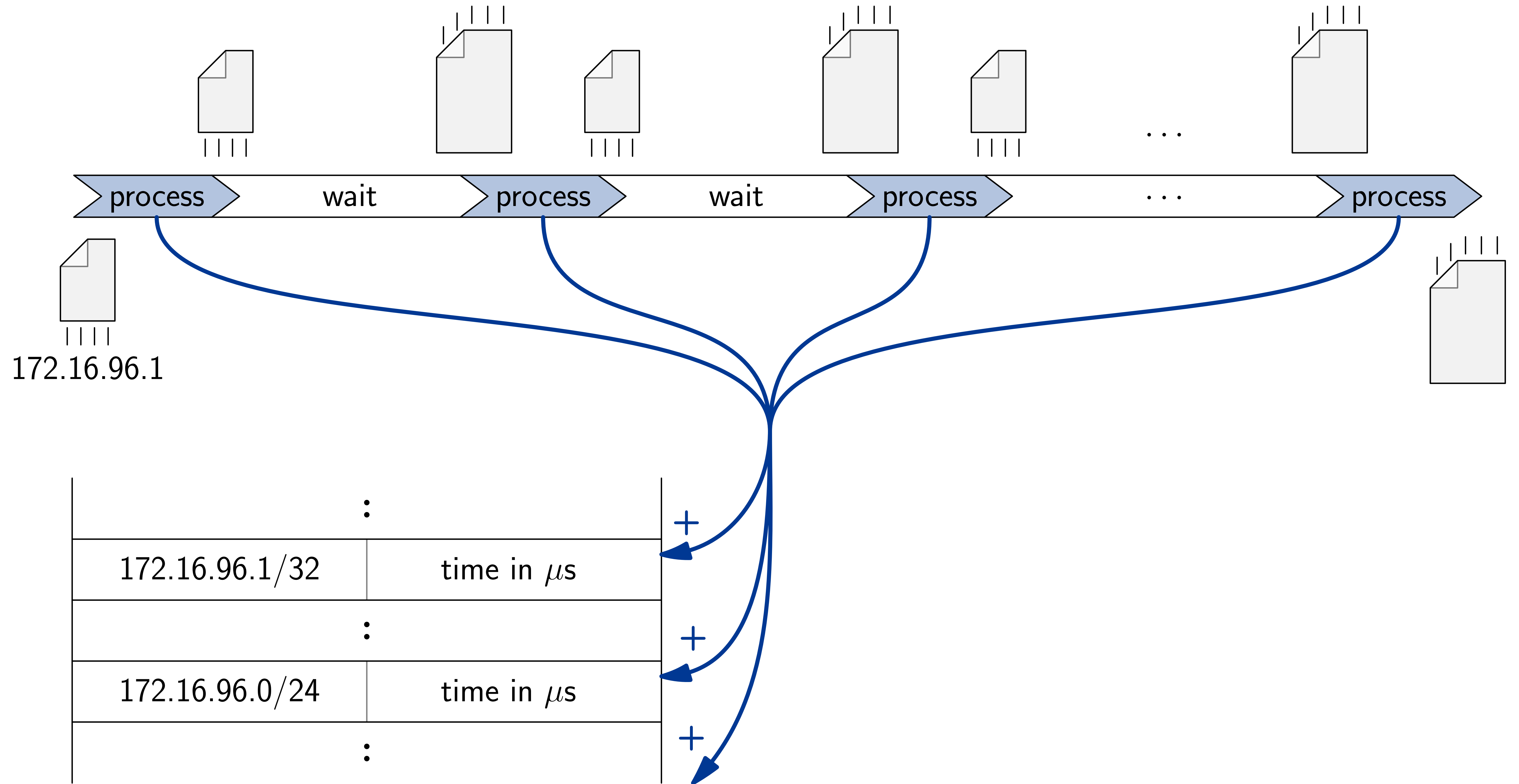
Prioritization

- **measuring time**
  - **only cpu, no wait**



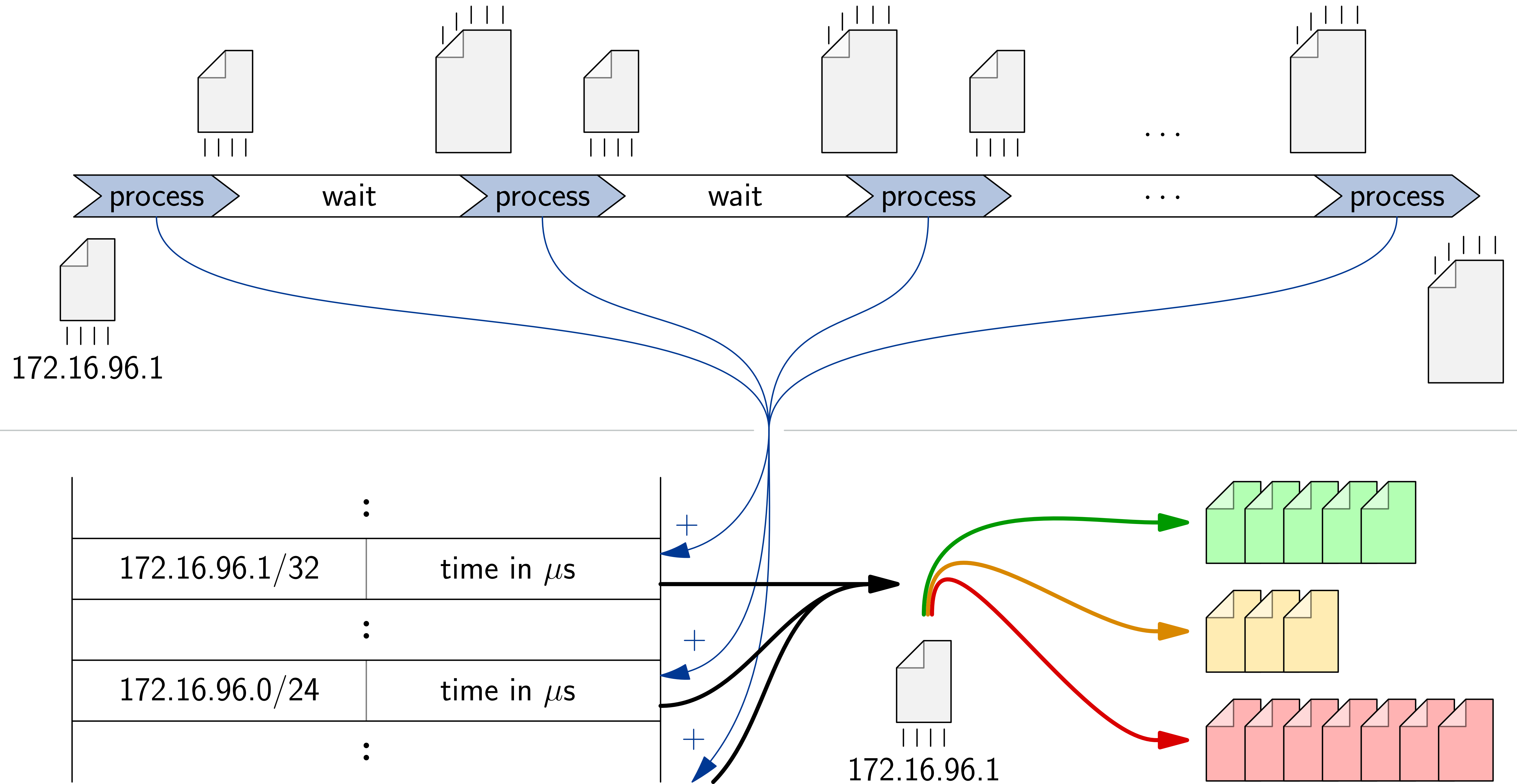
Prioritization

- measuring time
  - only cpu, no wait
  - **add to table values**



Prioritization

- measuring time
  - only cpu, no wait
  - add to table values
- **defer to queues**
  - **based on values**



cz.nic | CZ DOMAIN REGISTRY

Implementation

- **hashing**

$\text{hash}(172.16.96.1/32) = 1011110110000110101011000011010100111010011101101001000001011111$

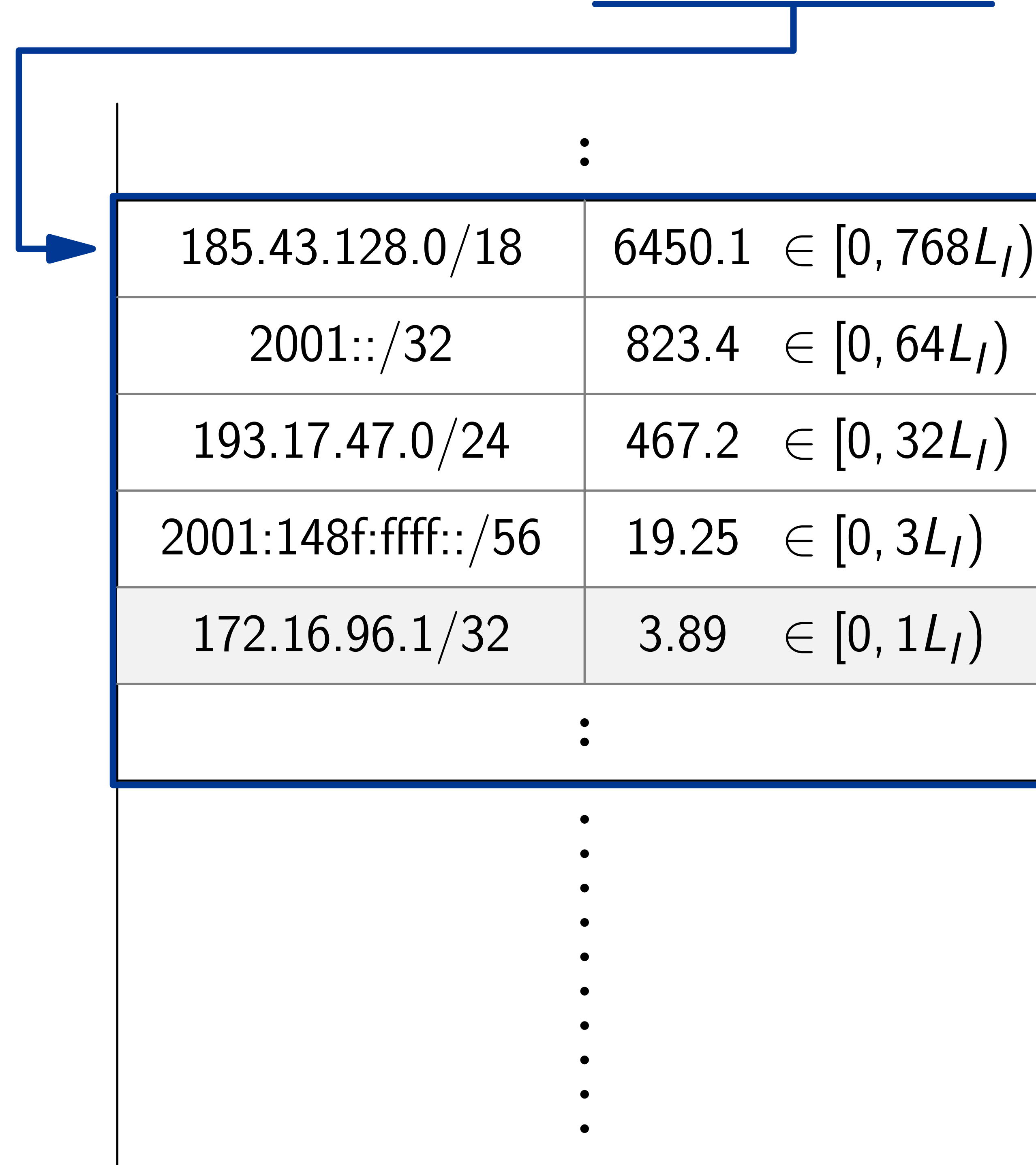
⋮		
172.16.96.1/32	3	$\in [0, 1L_l)$
⋮		
172.16.96.0/24	15.34	$\in [0, 32L_l)$
⋮		
172.16.96.0/20	123	$\in [0, 256L_l)$
⋮		
2001:db8::734/128	7.569	$\in [0, 1L_l)$
⋮		
2001:db8::/32	33.21	$\in [0, 64L_l)$
⋮		

cz.nic CZ DOMAIN REGISTRY

Implementation

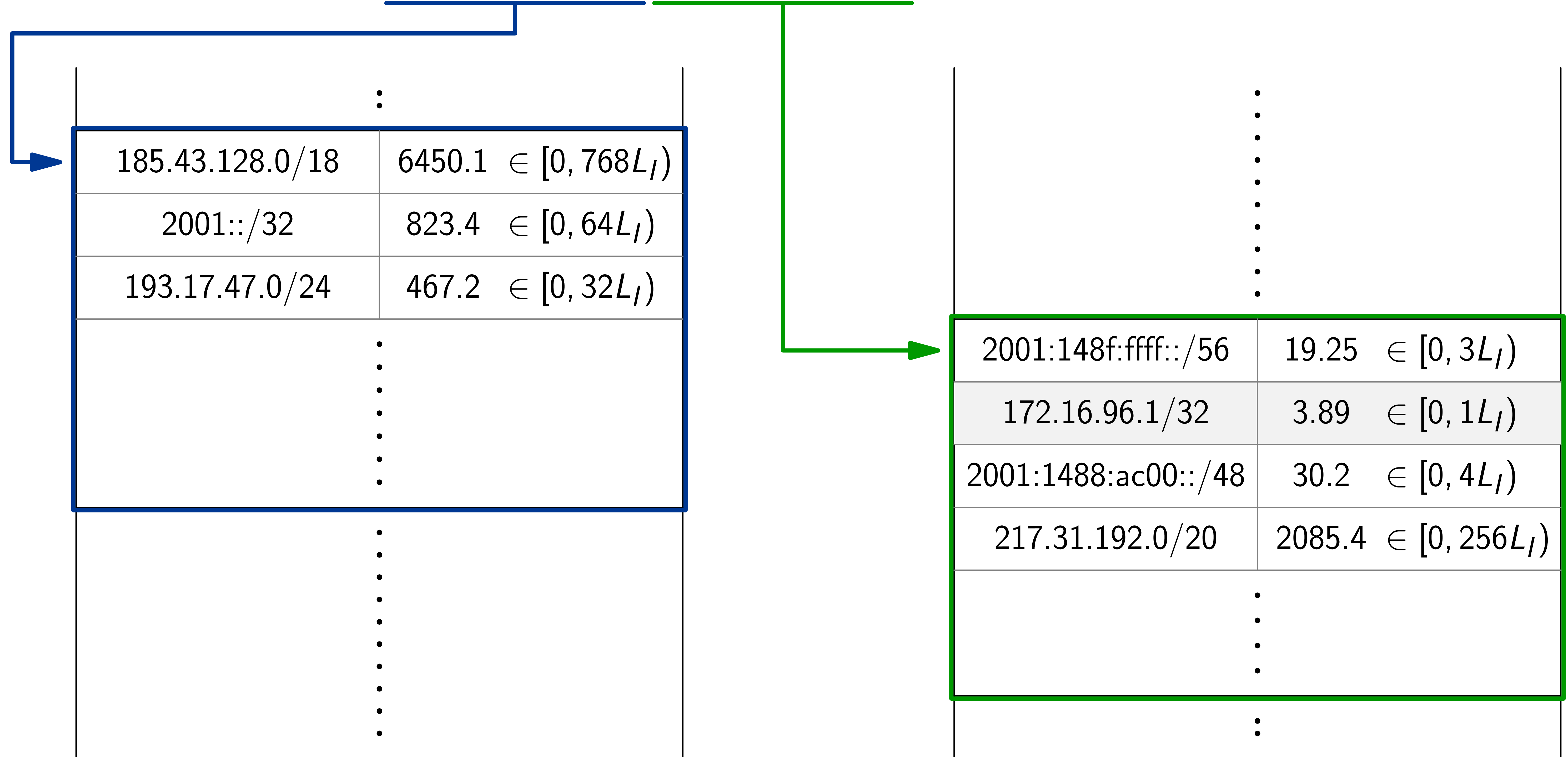
- hashing
  - **buckets**

$\text{hash}(172.16.96.1/32) = 1011110110000110101011000011010100111010011101101001000001011111$



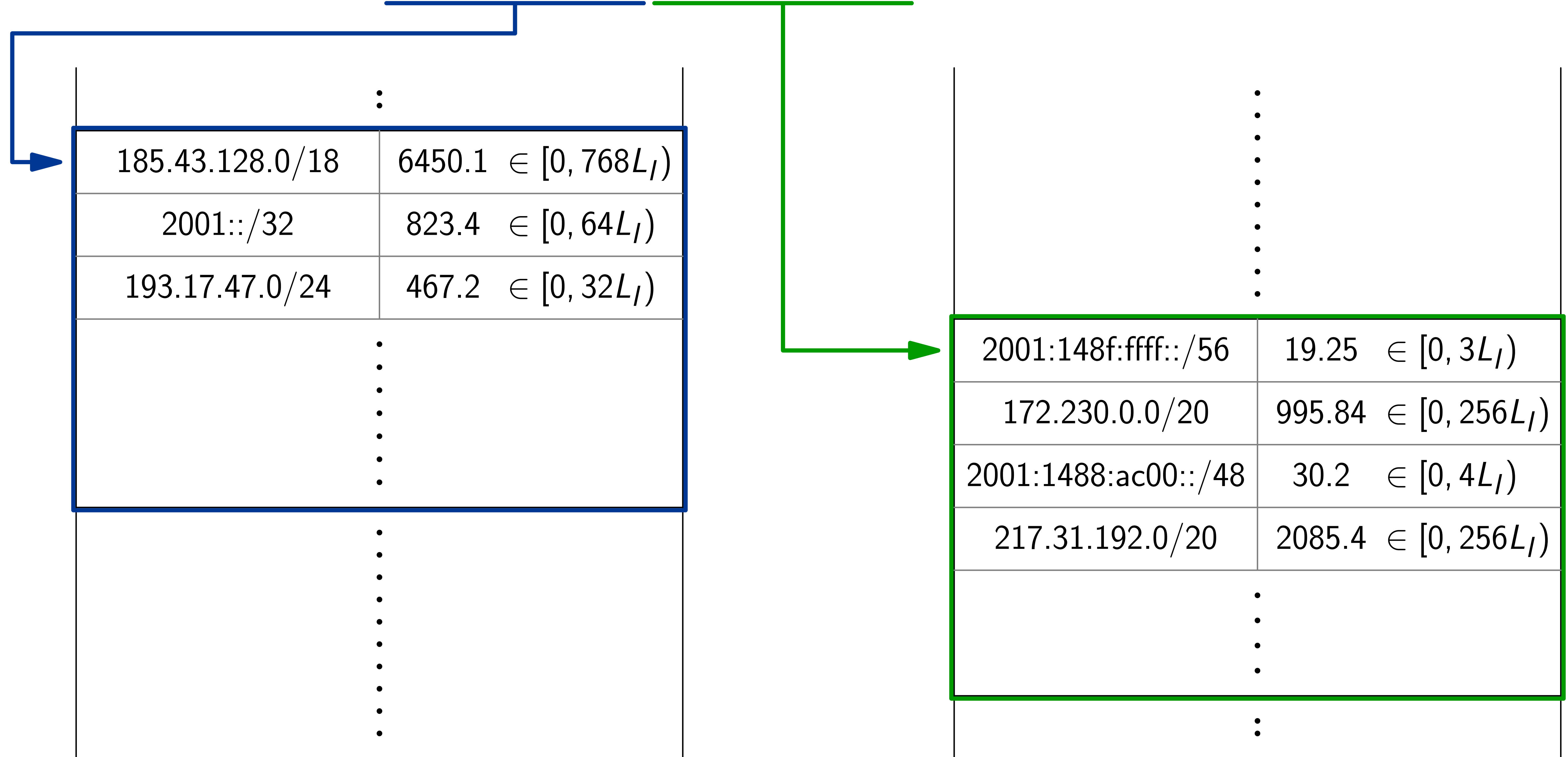
Implementation

- hashing
  - buckets
  - **two tables**



Implementation

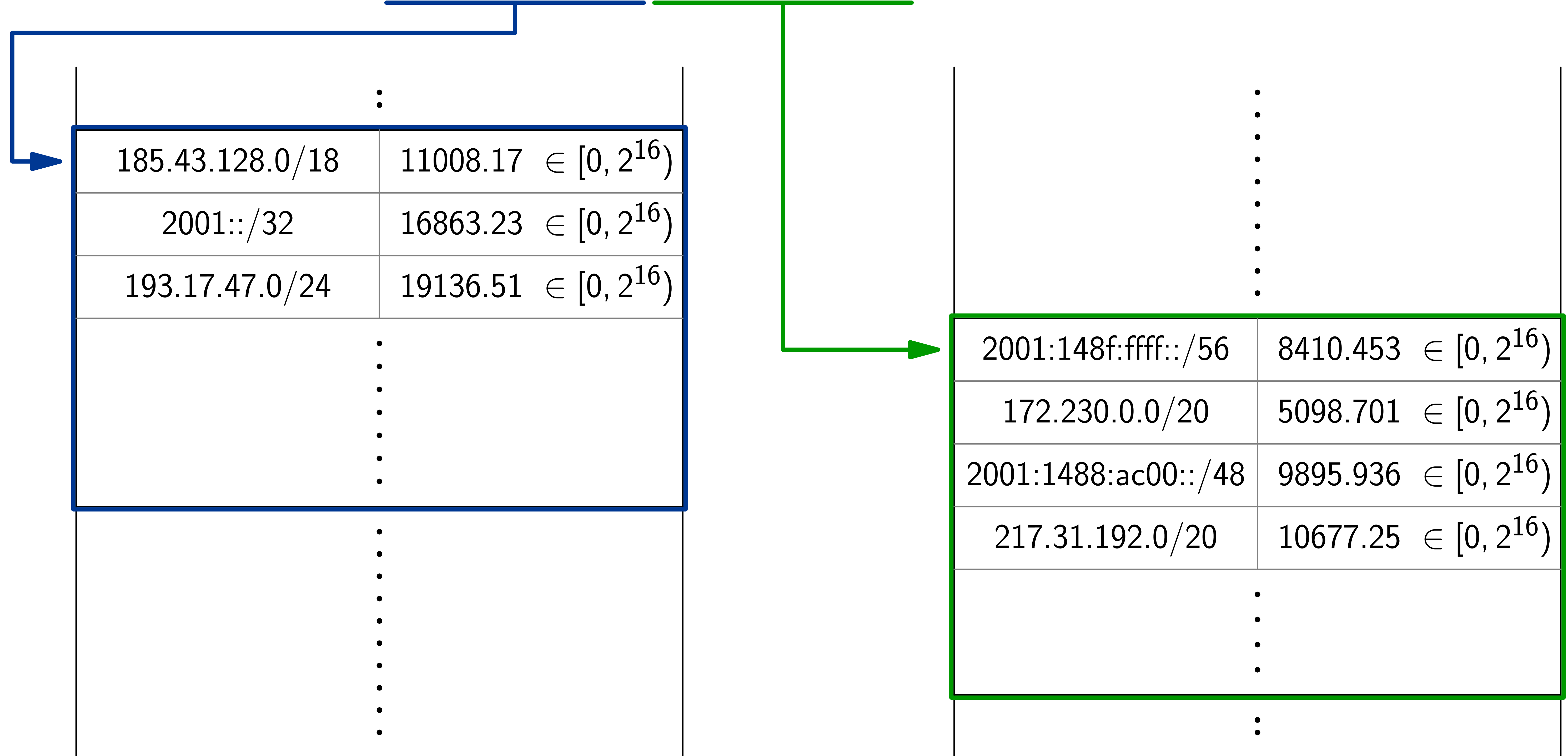
- hashing
  - buckets
  - two tables
- **evicting**



cz.nic | CZ DOMAIN REGISTRY

Implementation

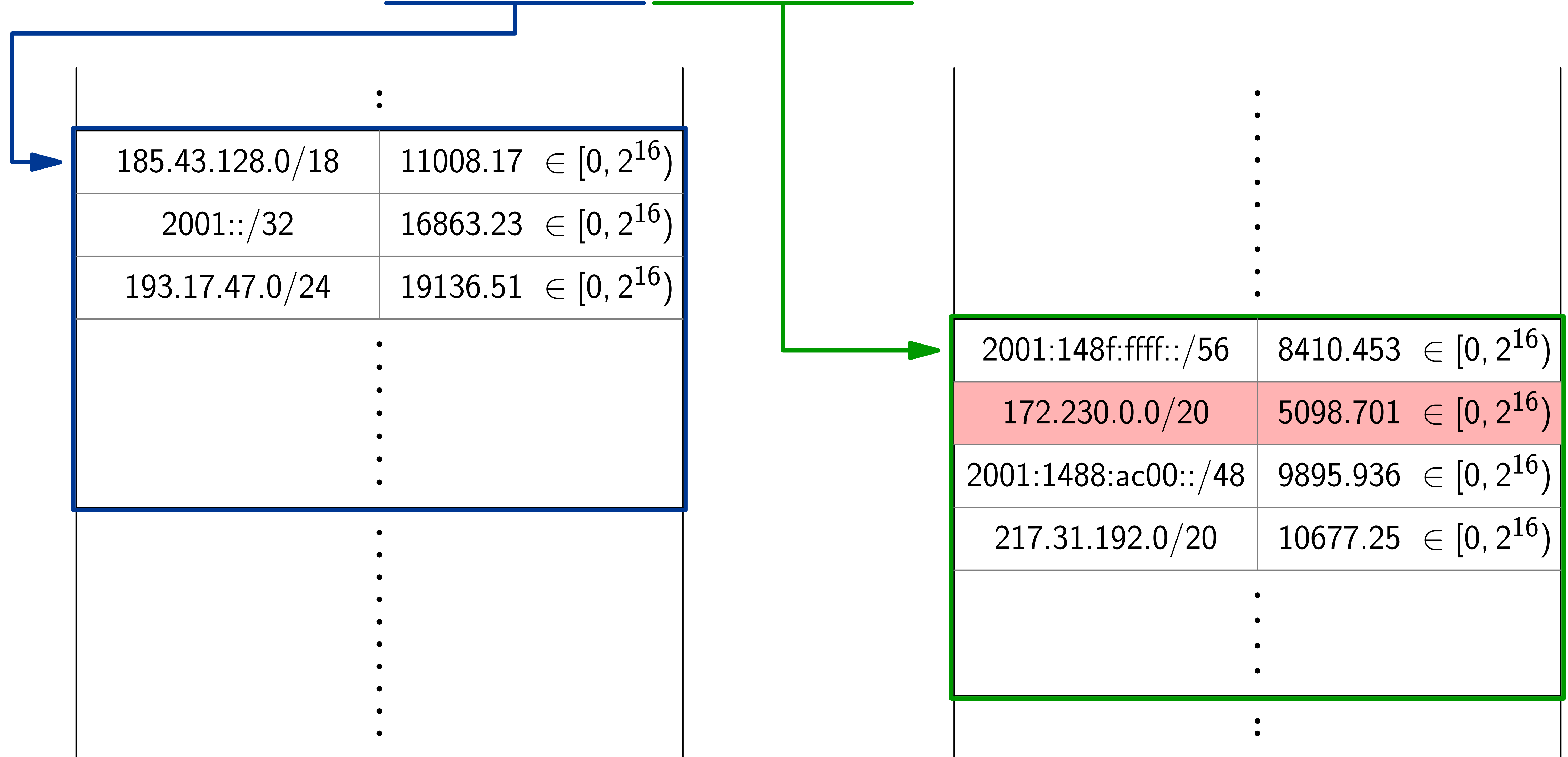
- hashing
  - buckets
  - two tables
- evicting
  - **normalized limits**



cz.nic | CZ DOMAIN REGISTRY

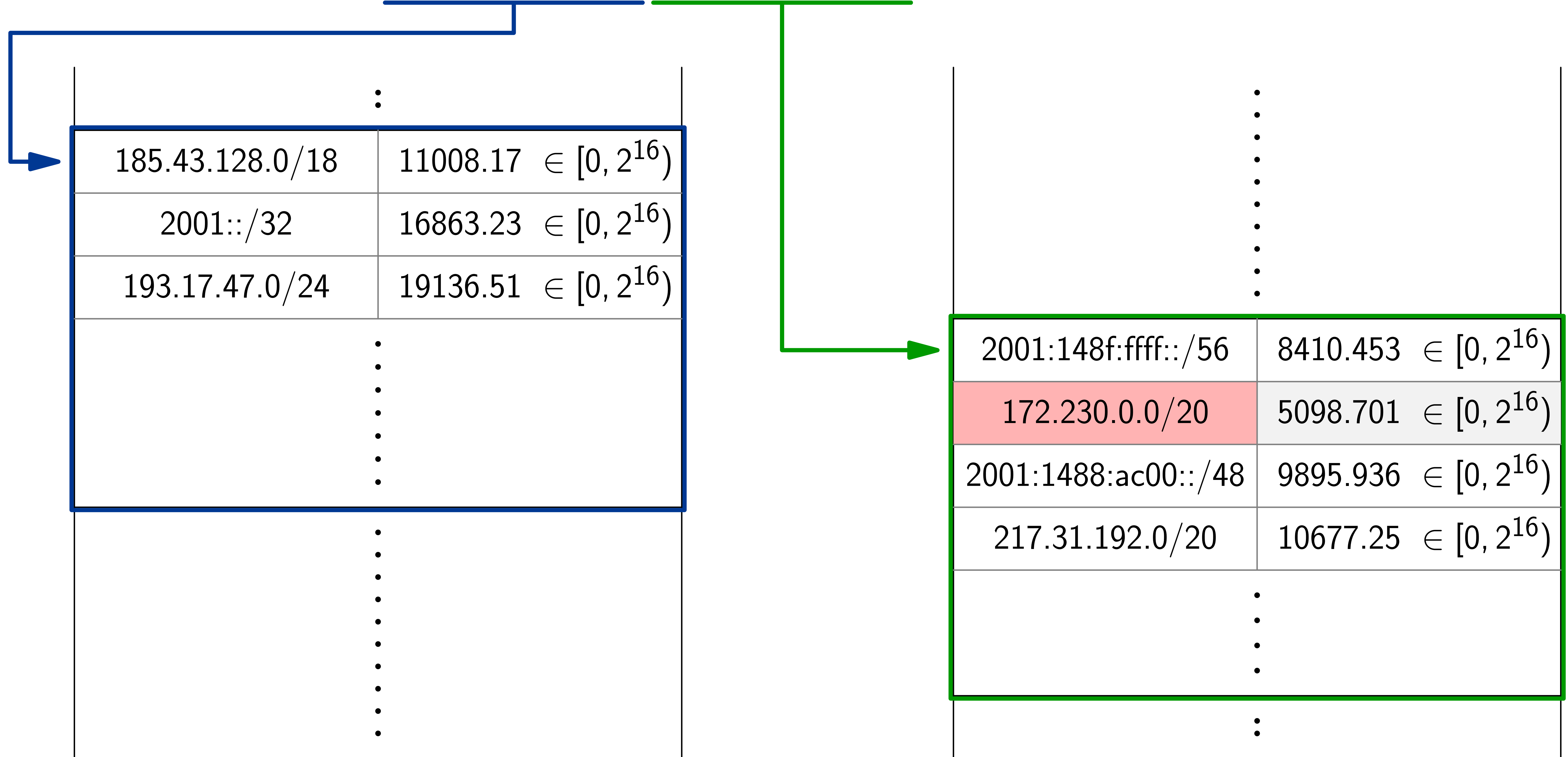
Implementation

- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - **choosing minimal**



Implementation

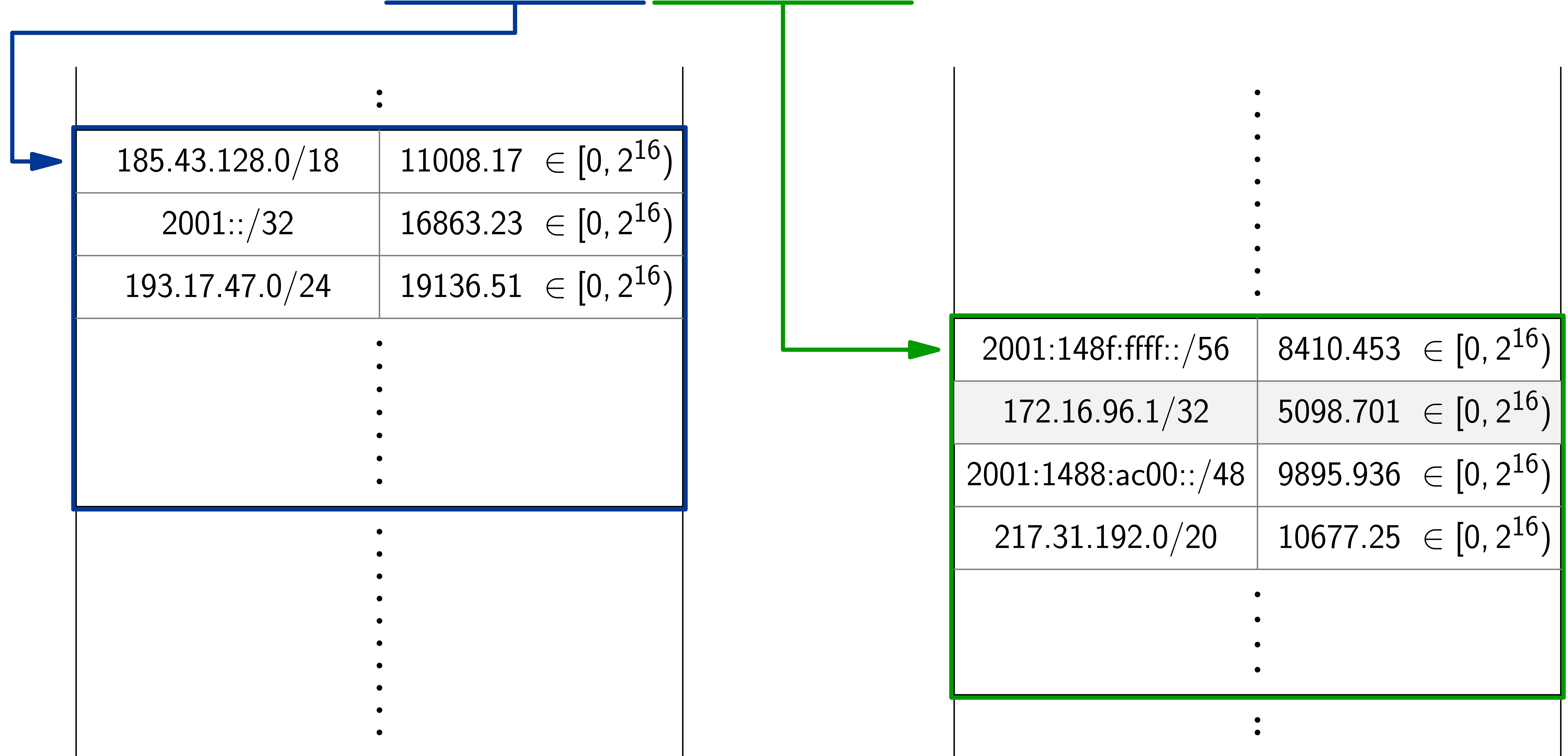
- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - **keeping value**



CZ.NIC CZ DOMAIN REGISTRY

Implementation

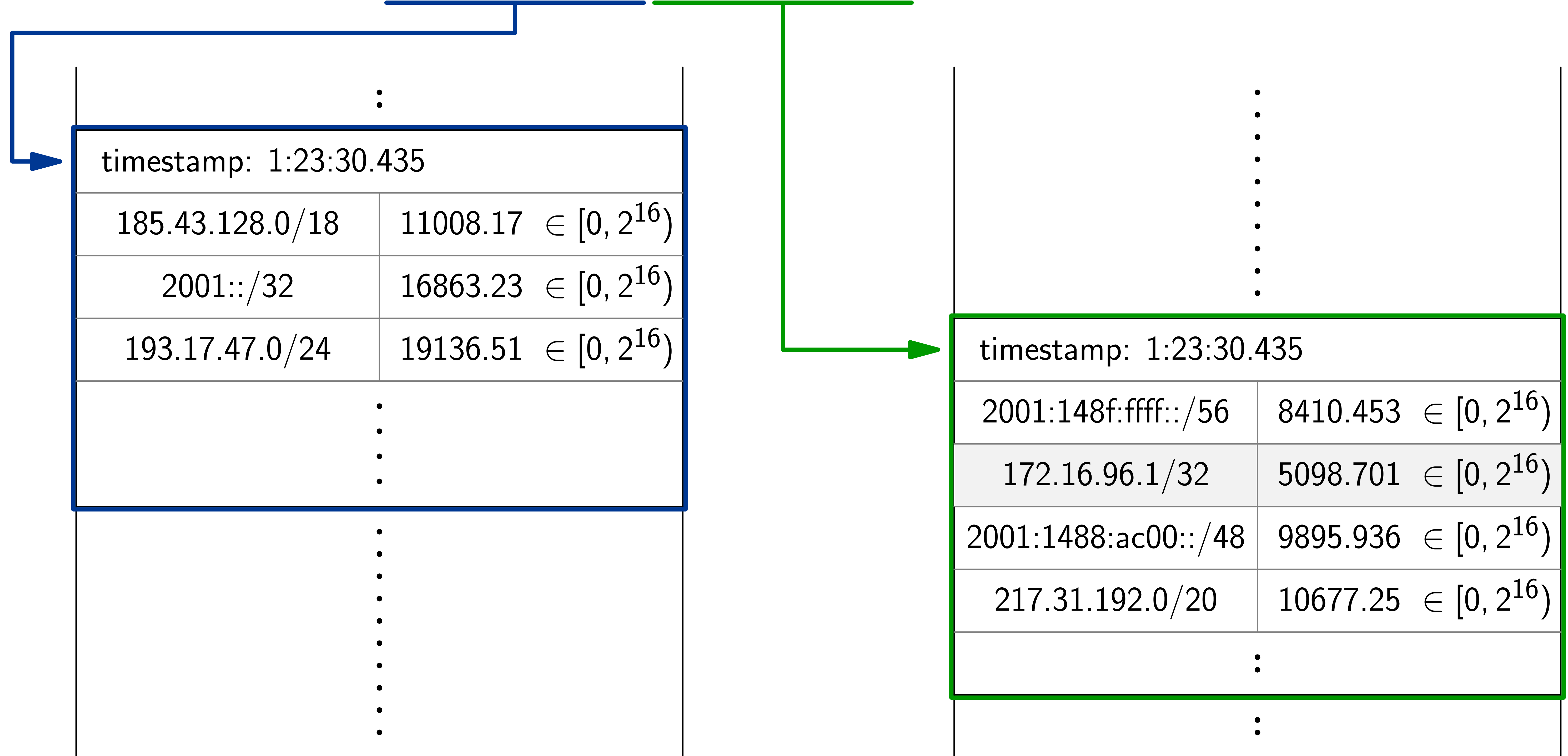
- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- **lazy decay**



CZ.NIC CZ DOMAIN REGISTRY

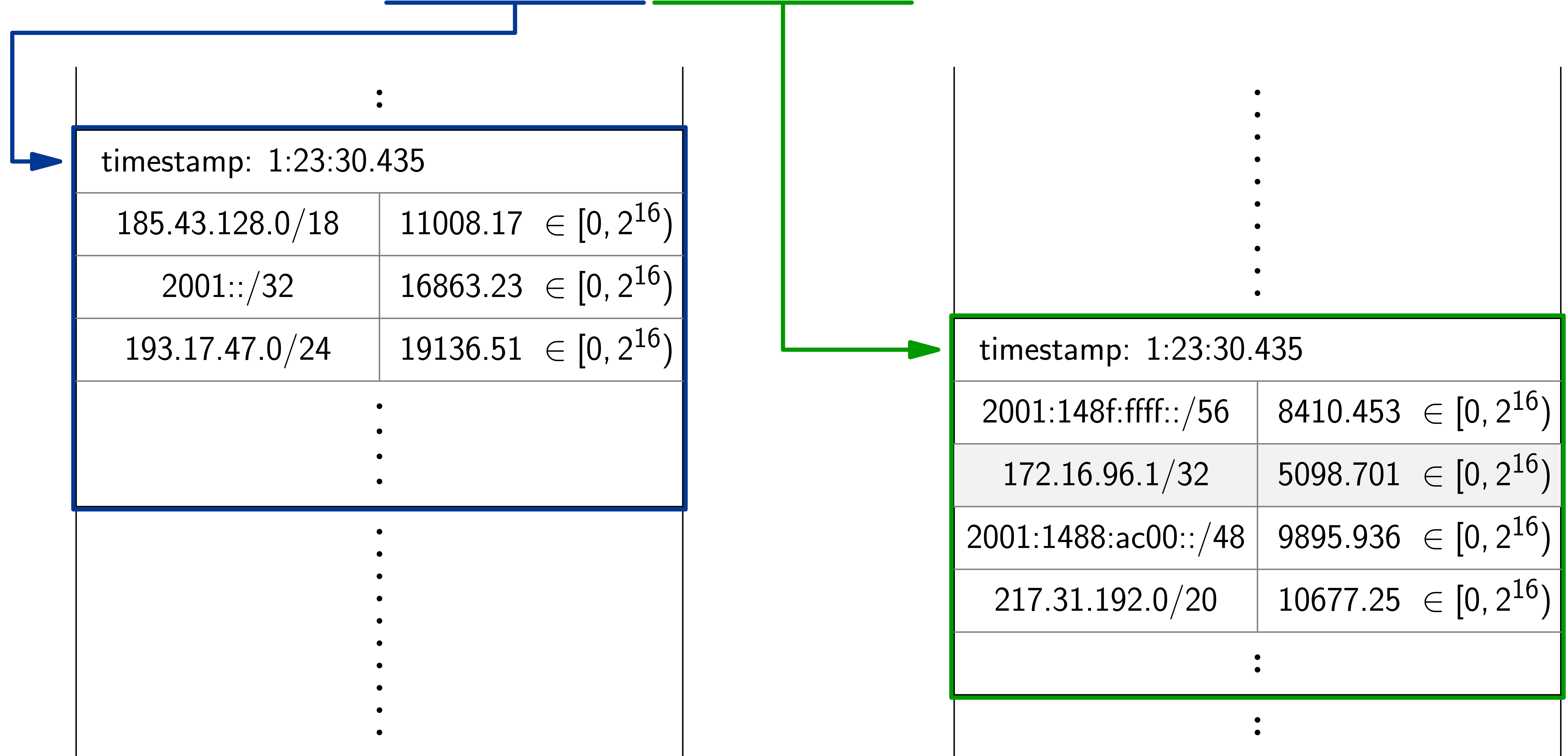
Implementation

- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- **lazy decay**



Implementation

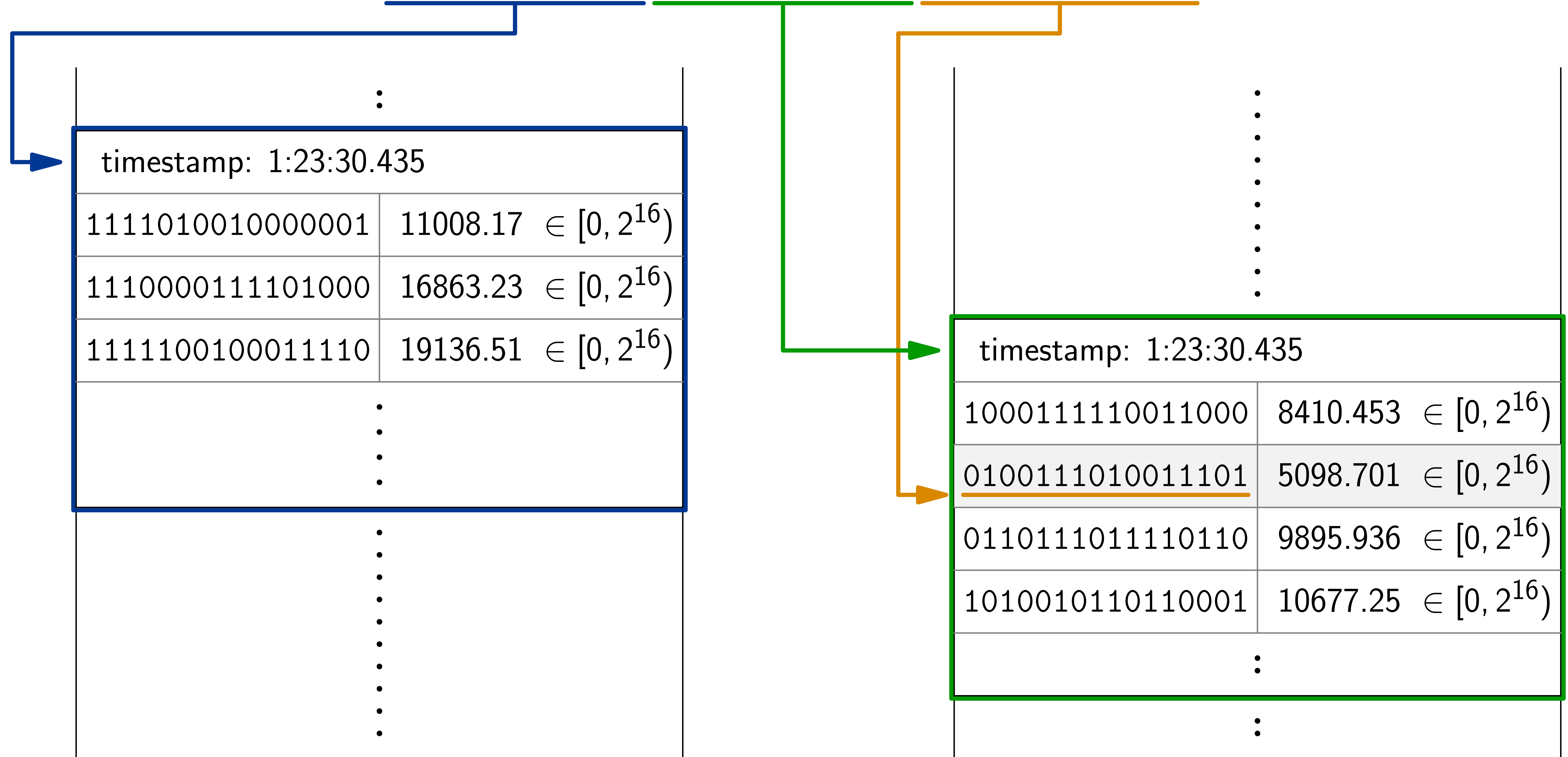
- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- **memory layout**



cz.nic | CZ DOMAIN REGISTRY

Implementation

- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- memory layout
  - **hashed labels**



cz.nic | CZ DOMAIN REGISTRY

Implementation

- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- memory layout
  - hashed labels
  - **prob. rounding**

1111010010000001	11008.17 $\in [0, 2^{16})$
1110000111101000	16863.23 $\in [0, 2^{16})$
1111100100011110	19136.51 $\in [0, 2^{16})$
⋮	

1000111110011000	8410.453 $\in [0, 2^{16})$
0100111010011101	5098.701 $\in [0, 2^{16})$
0110111011110110	9895.936 $\in [0, 2^{16})$
1010010110110001	10677.25 $\in [0, 2^{16})$
⋮	

cz.nic | CZ DOMAIN REGISTRY

Implementation

- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- memory layout
  - hashed labels
  - **prob. rounding**

$\mathrm{hash}(172.16.96.1/32) = 1011110110000110101011000011010100111$

rounding: 5098 . 7008

16-bit base value

16-bit probability of rounding up (70.08 %)

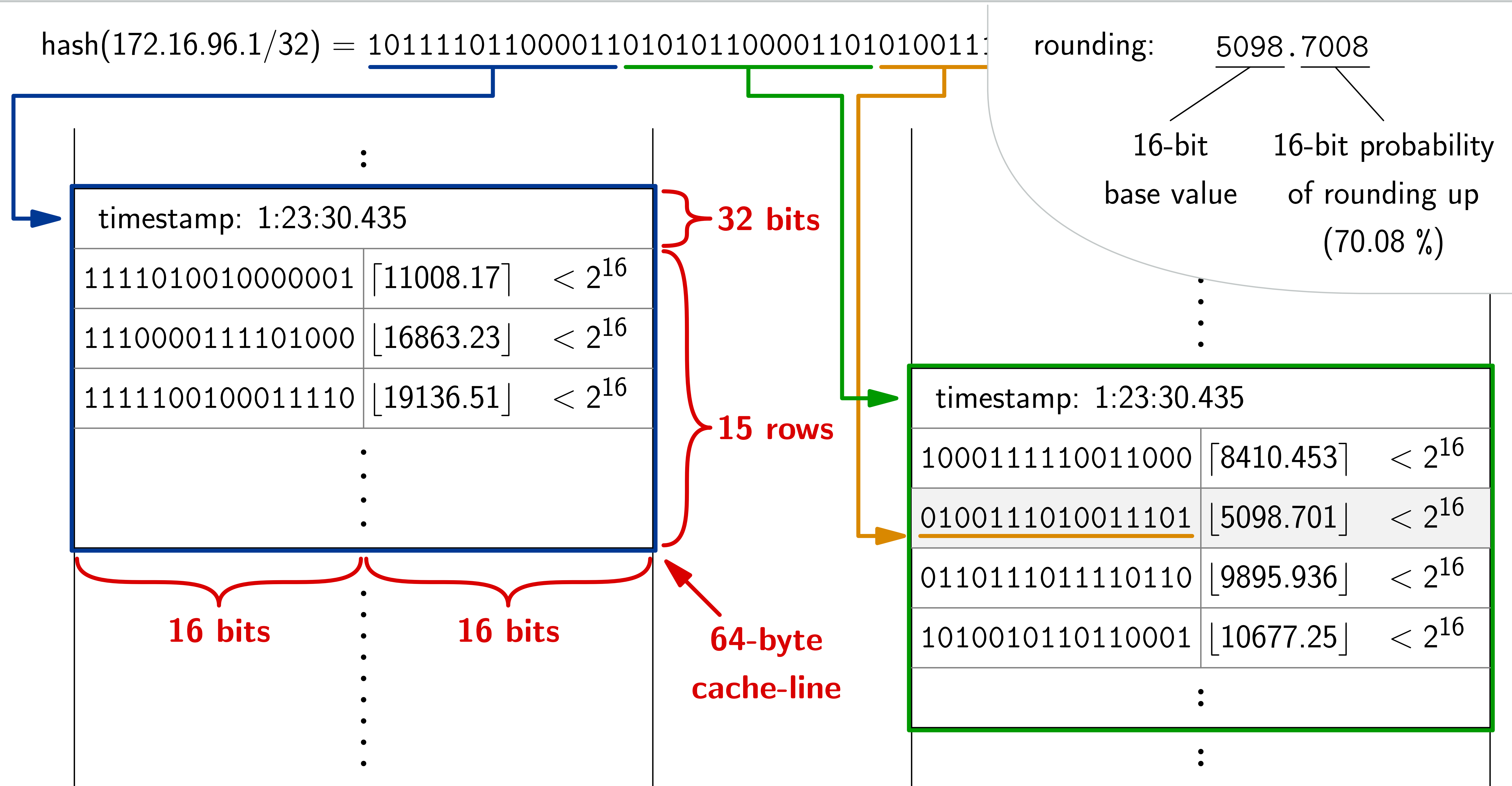
timestamp: 1:23:30.435		
1111010010000001	$\lceil 11008.17 \rceil$	$< 2^{16}$
1110000111101000	$\lfloor 16863.23 \rfloor$	$< 2^{16}$
1111100100011110	$\lfloor 19136.51 \rfloor$	$< 2^{16}$
⋮		

timestamp: 1:23:30.435		
1000111110011000	$\lceil 8410.453 \rceil$	$< 2^{16}$
0100111010011101	$\lfloor 5098.701 \rceil$	$< 2^{16}$
0110111011110110	$\lfloor 9895.936 \rceil$	$< 2^{16}$
1010010110110001	$\lfloor 10677.25 \rceil$	$< 2^{16}$
⋮		

cz.nic | CZ DOMAIN REGISTRY

Implementation

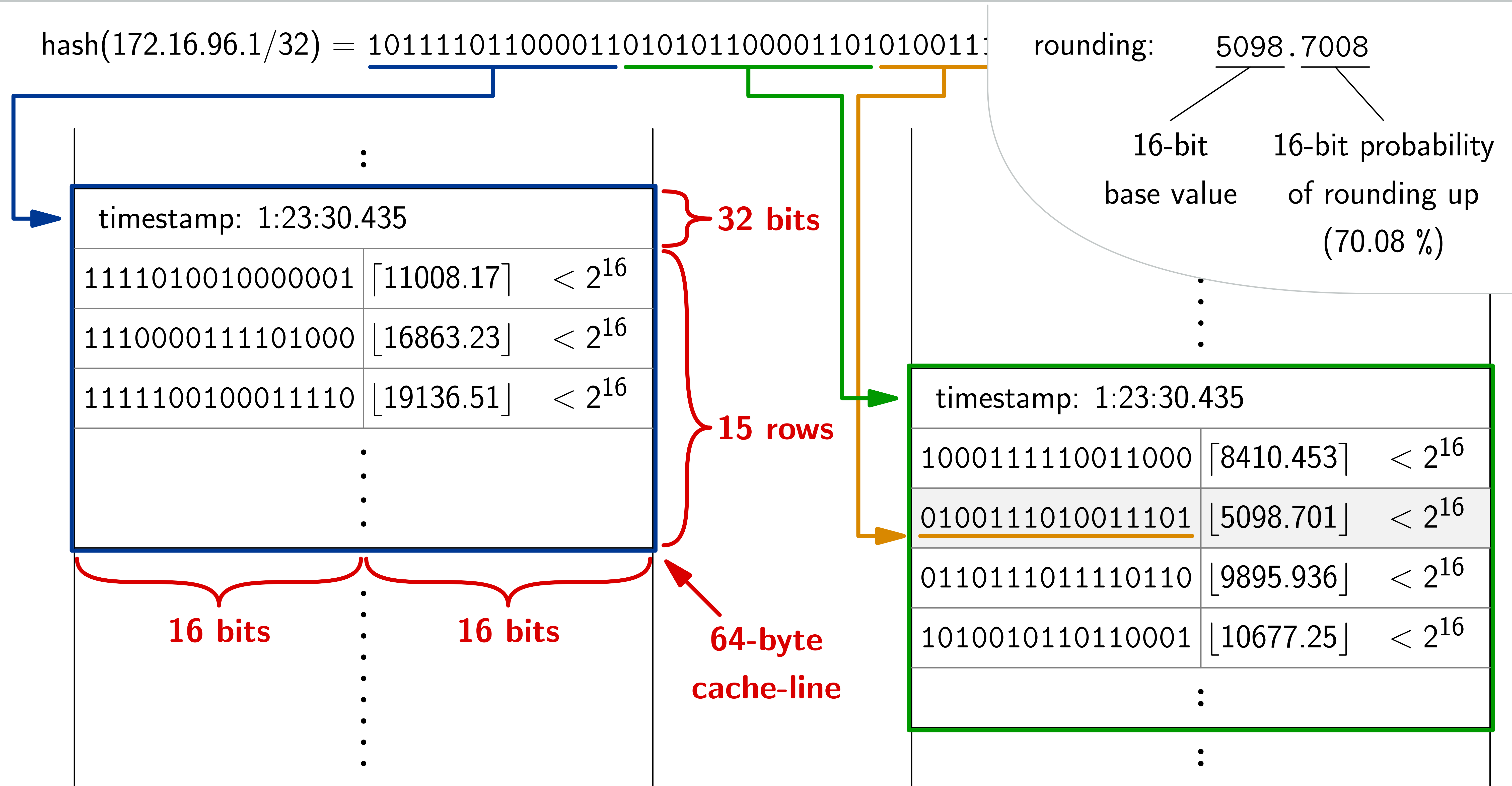
- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- memory layout
  - hashed labels
  - prob. rounding
  - **fit in cache-line**



cz.nic | CZ DOMAIN REGISTRY

Implementation

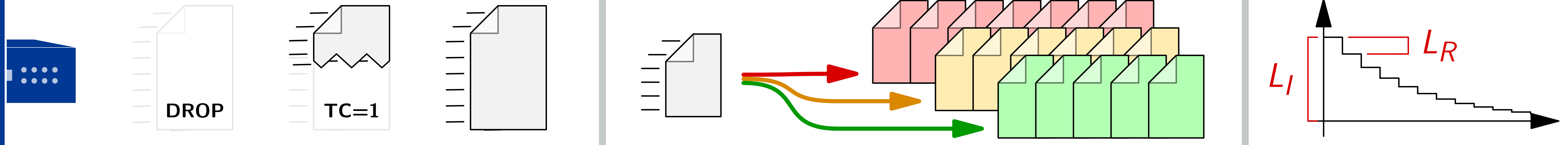
- hashing
  - buckets
  - two tables
- evicting
  - normalized limits
  - choosing minimal
  - keeping value
- lazy decay
- memory layout
  - hashed labels
  - prob. rounding
  - fit in cache-line
- **optimizations**
  - **prefetching**
  - **lock-free**
  - **vectorization**



cz.nic CZ DOMAIN REGISTRY

Summary

- rate-limiting
  - counting UDP queries
  - truncating or dropping
- prioritization
  - measuring time
  - reordering
- counters
  - instant/rate limit
  - exponential decay
  - higher limits for shorter prefixes
- implementation⇒



hash(172.16.96.1/32) = 1011110110000110101011000011010100111

rounding: 5098 . 7008

16-bit base value

16-bit probability of rounding up (70.08 %)

timestamp: 1:23:30.435	
1111010010000001	⌈11008.17⌉ $< 2^{16}$
1110000111101000	⌈16863.23⌉ $< 2^{16}$
1111100100011110	⌈19136.51⌉ $< 2^{16}$
⋮	

32 bits

15 rows

16 bits

16 bits

64-byte cache-line

timestamp: 1:23:30.435	
1000111110011000	⌈8410.453⌉ $< 2^{16}$
0100111010011101	⌈5098.701⌉ $< 2^{16}$
0110111011110110	⌈9895.936⌉ $< 2^{16}$
1010010110110001	⌈10677.25⌉ $< 2^{16}$
⋮	

cz.nic CZ DOMAIN REGISTRY