Collective Communication: Theory and Practice

Robert van de Geijn

Acknowledgements

This presentation is based on work in the mid-1990s that was sponsored in part by the Intel Research Council and Intel Scalable Systems Division. At that time, David Payne, Lance Shuler, and Jerrell Watts contributed to the research

Outline

Part I: Theory

- **Model of parallel computation**
- **Collective communications**
- **A building block approach to library implementation**

Part II: Practice

- **Implementation on the Paragon**
- **Performance results**

Outline

Part I: Theory

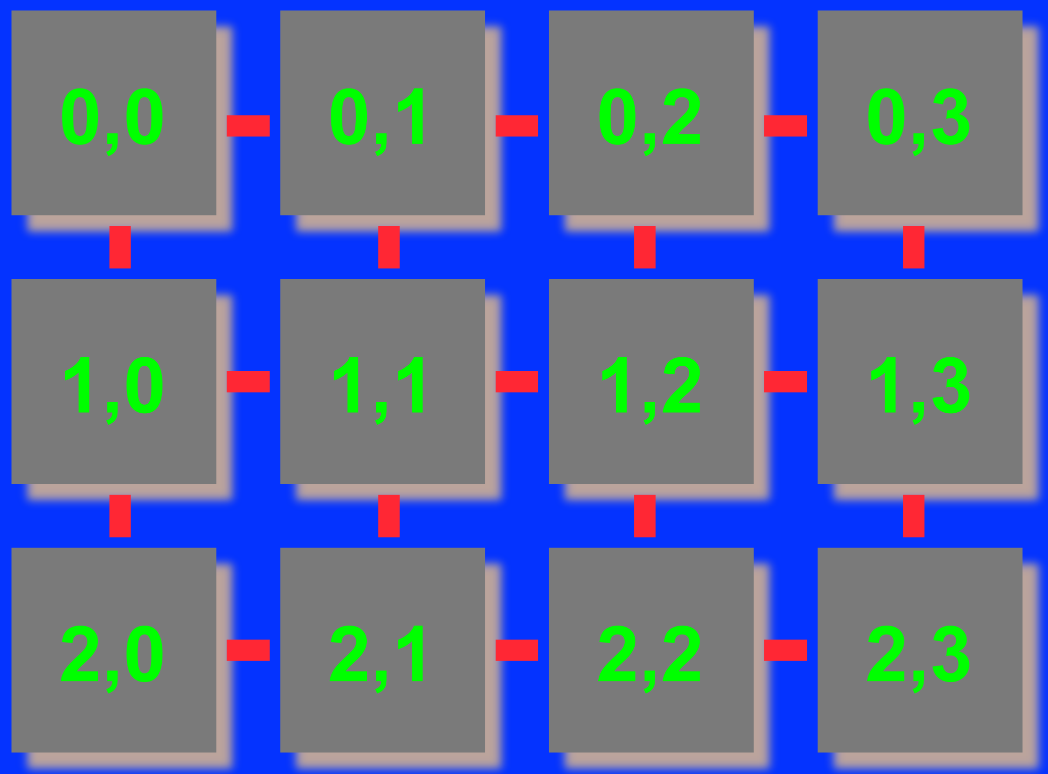
- Model of parallel computation
- Collective communications
- A building block approach to library implementation

Part II: Practice

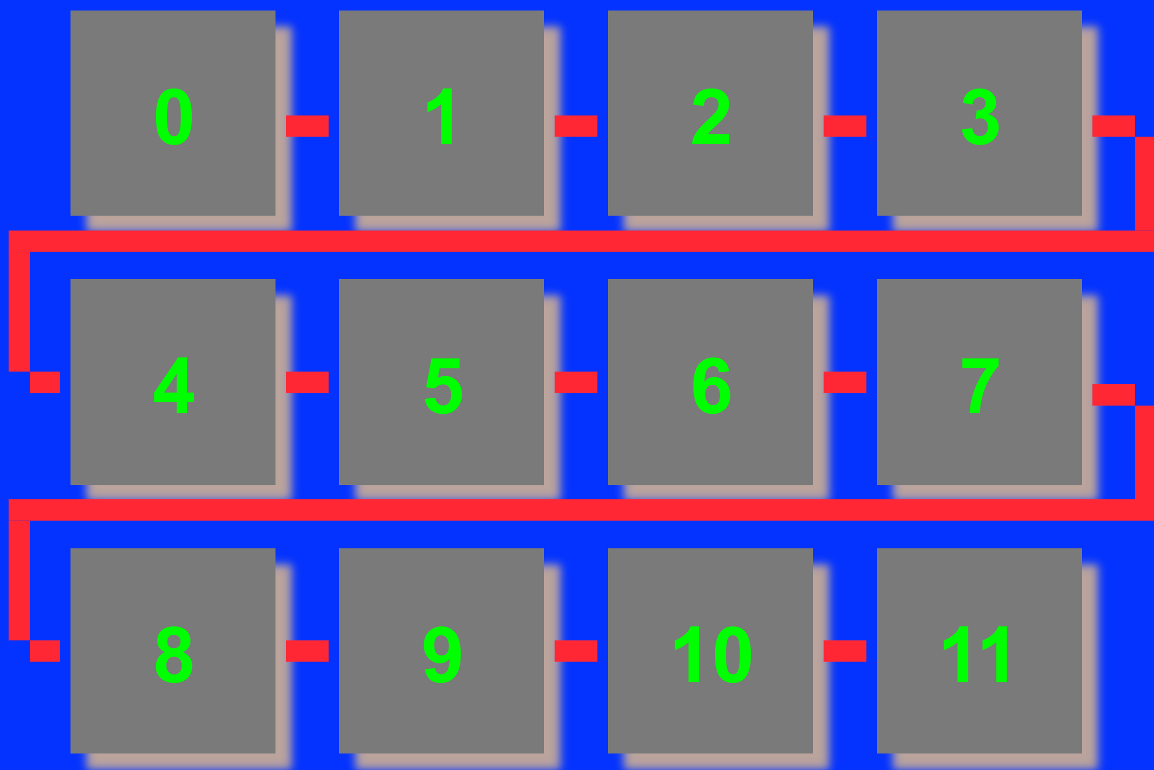
- Implementation on the Paragon
- Performance results
- Applications

Model of Parallel Computation

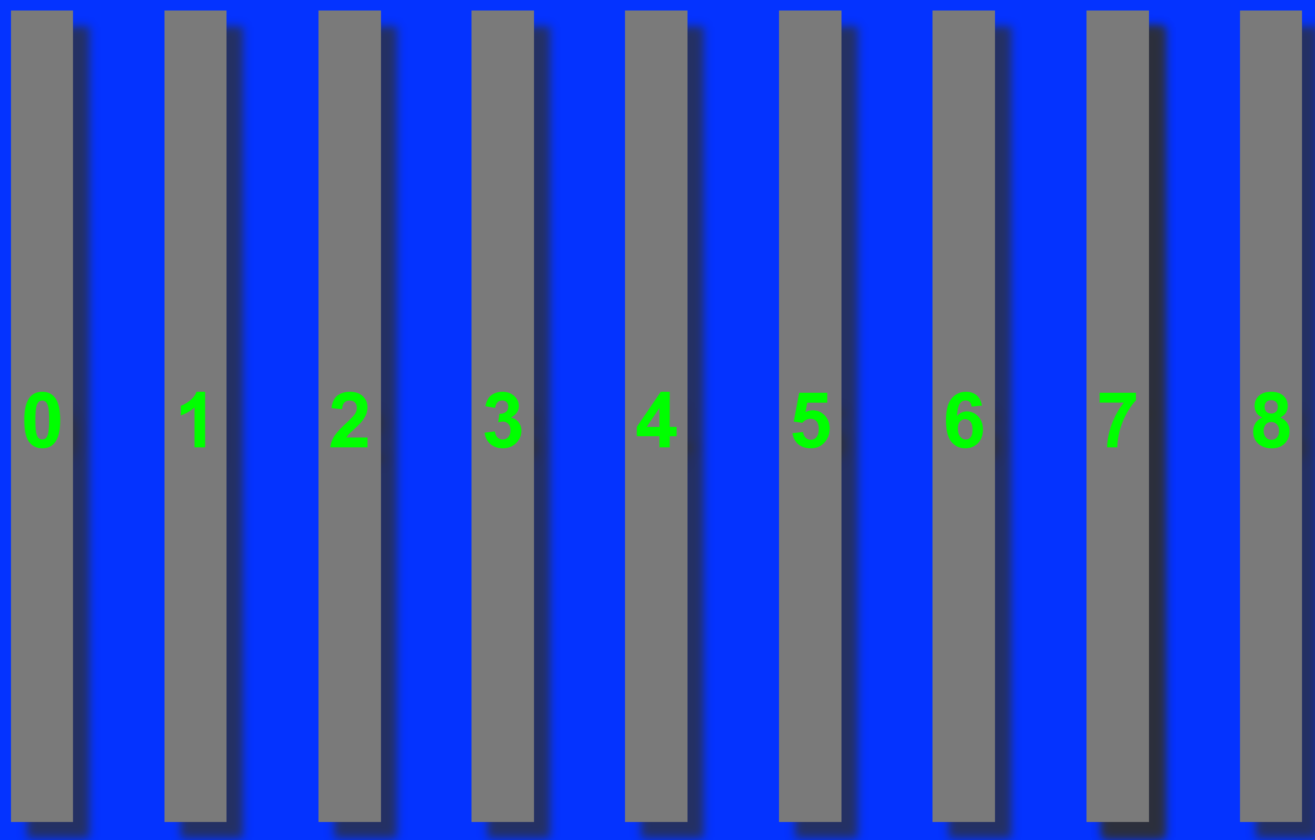
- *p* nodes
- physical two dimensional mesh
  - *r* rows, *c* columns
  - nodes have physical indices *(i,j)*
- often logically viewed as a linear array
  - indexed *0, ... , p-1*
  - nodes are numbered in row-major order



- **physical two dimensional mesh**
  - *r* rows, *c* columns
  - nodes have physical indices *(i,j)*



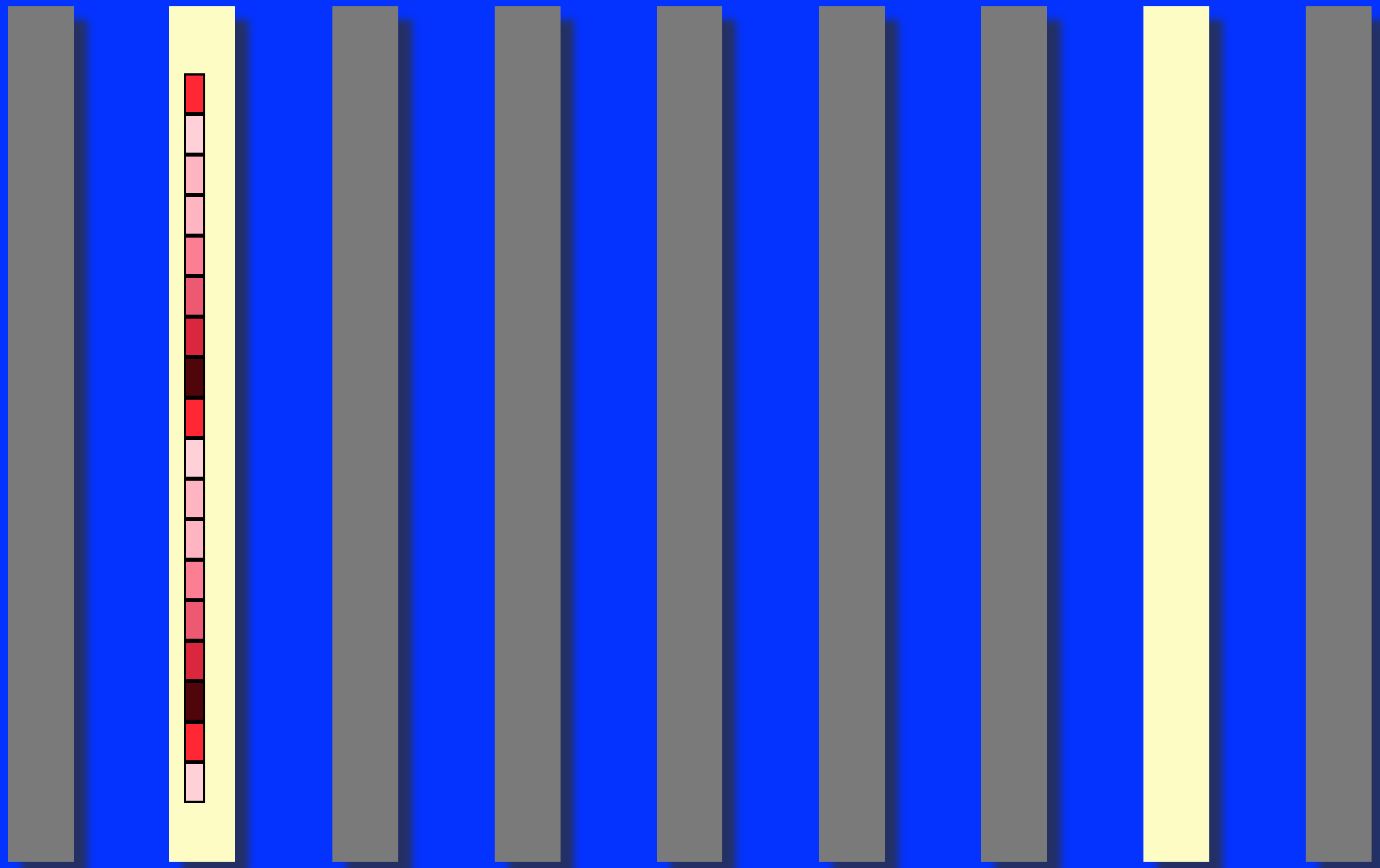
- often logically viewed as a linear array
  - indexed *0, ... , p-1*
  - nodes are numbered in row-major order

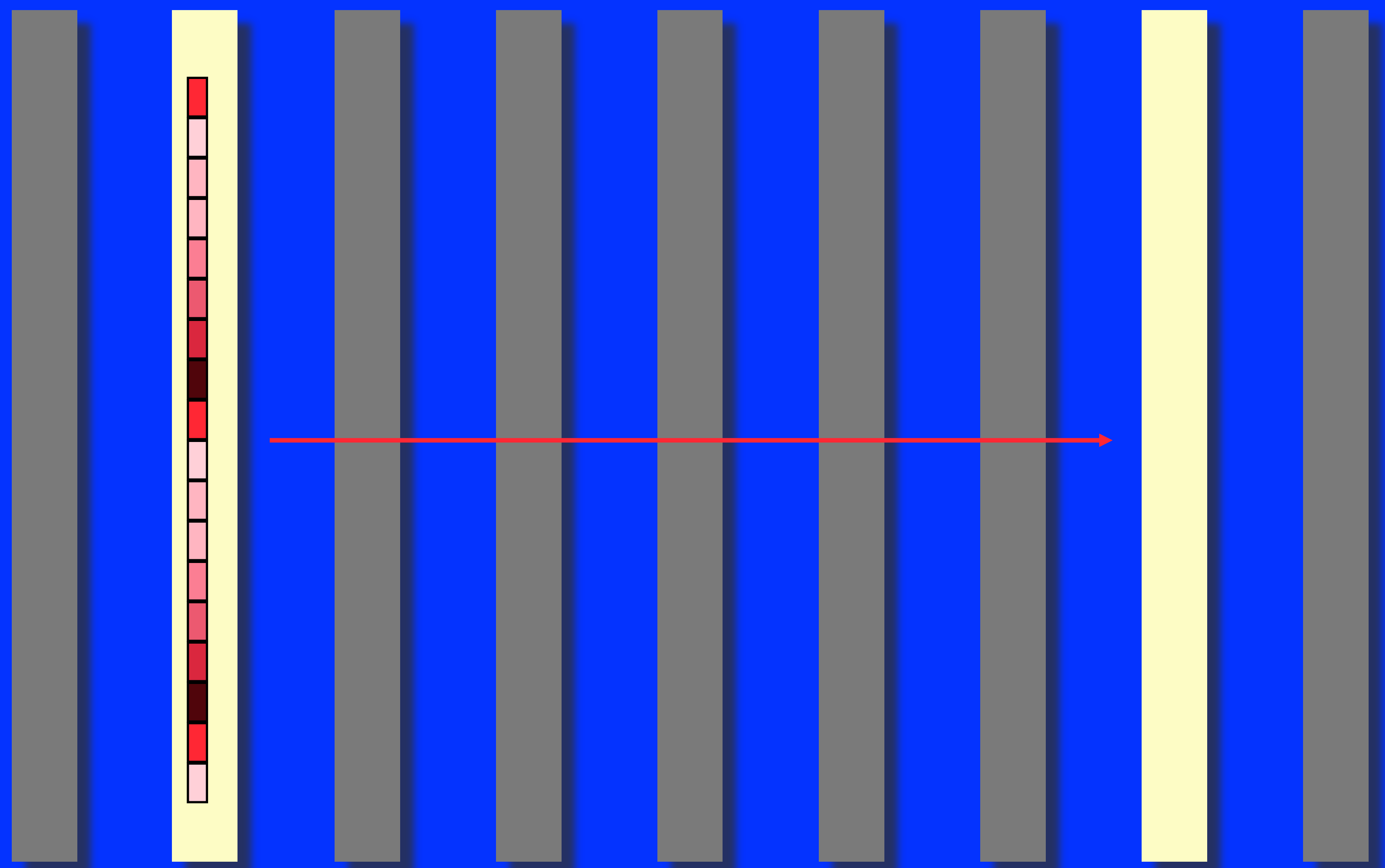


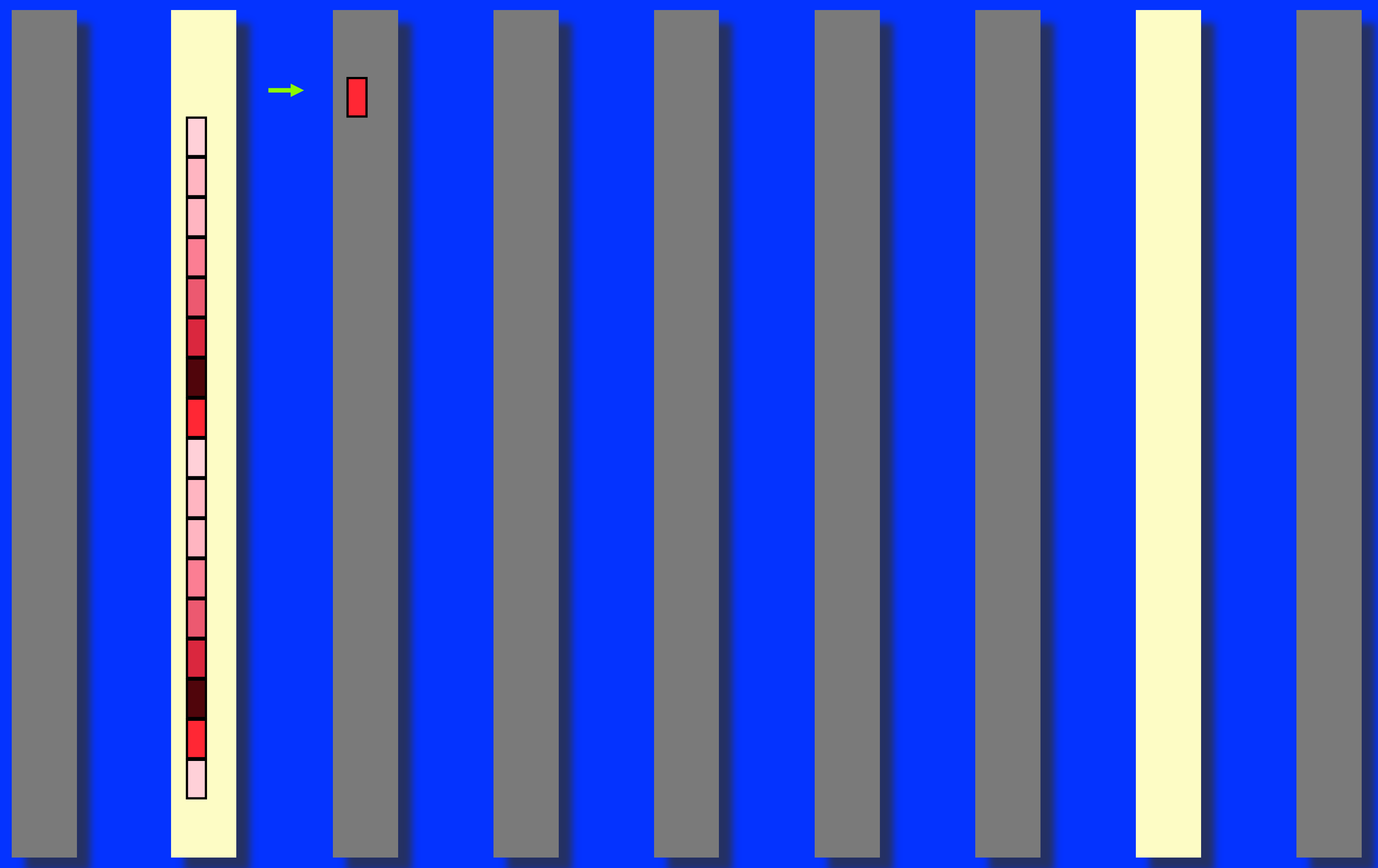
- **often logically viewed as a linear array**
  - **indexed** ***0, … , p-1***

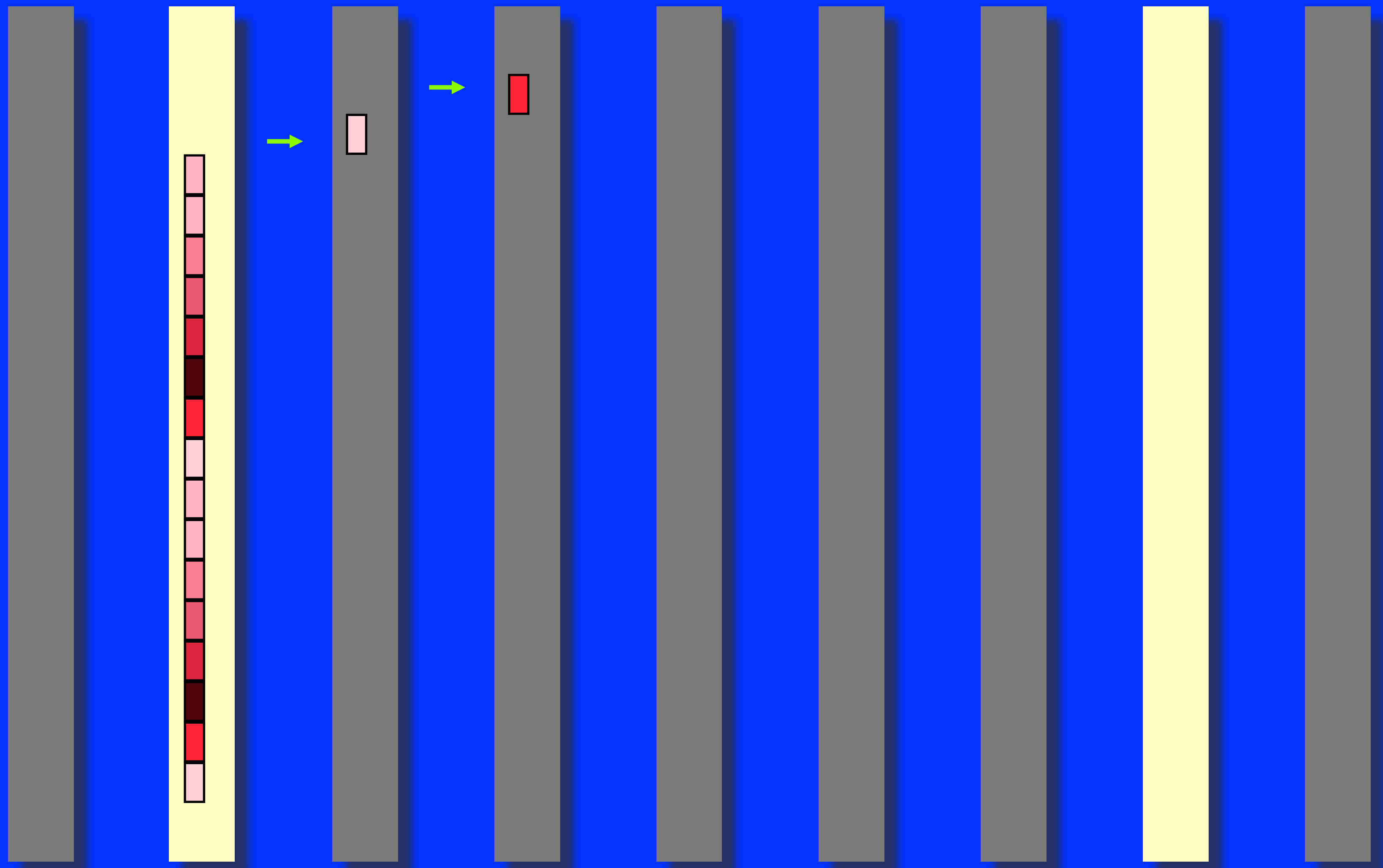
The Cost of Communication

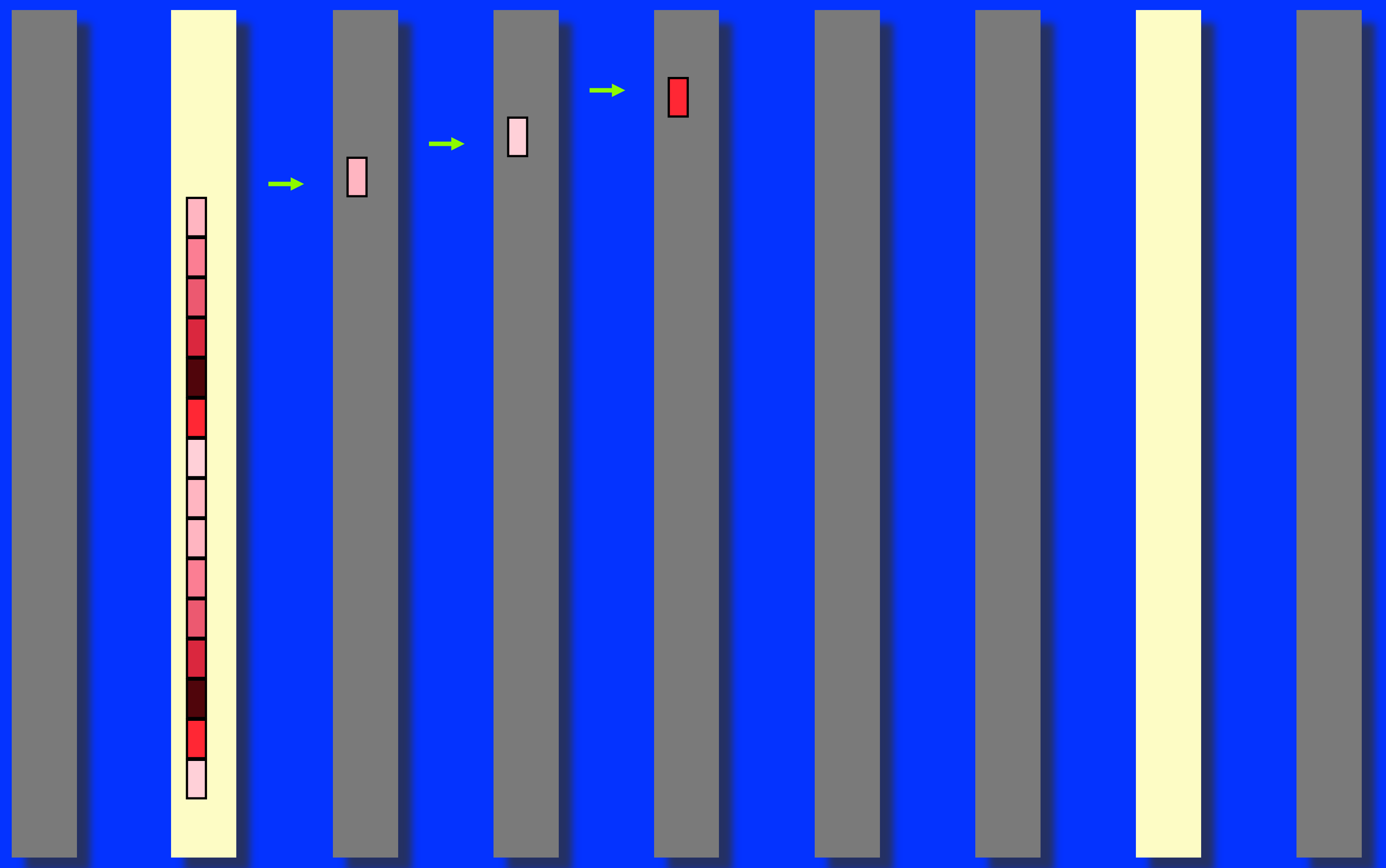
- send a message of length n over d links
- packetize the message
- Example: $d=6$

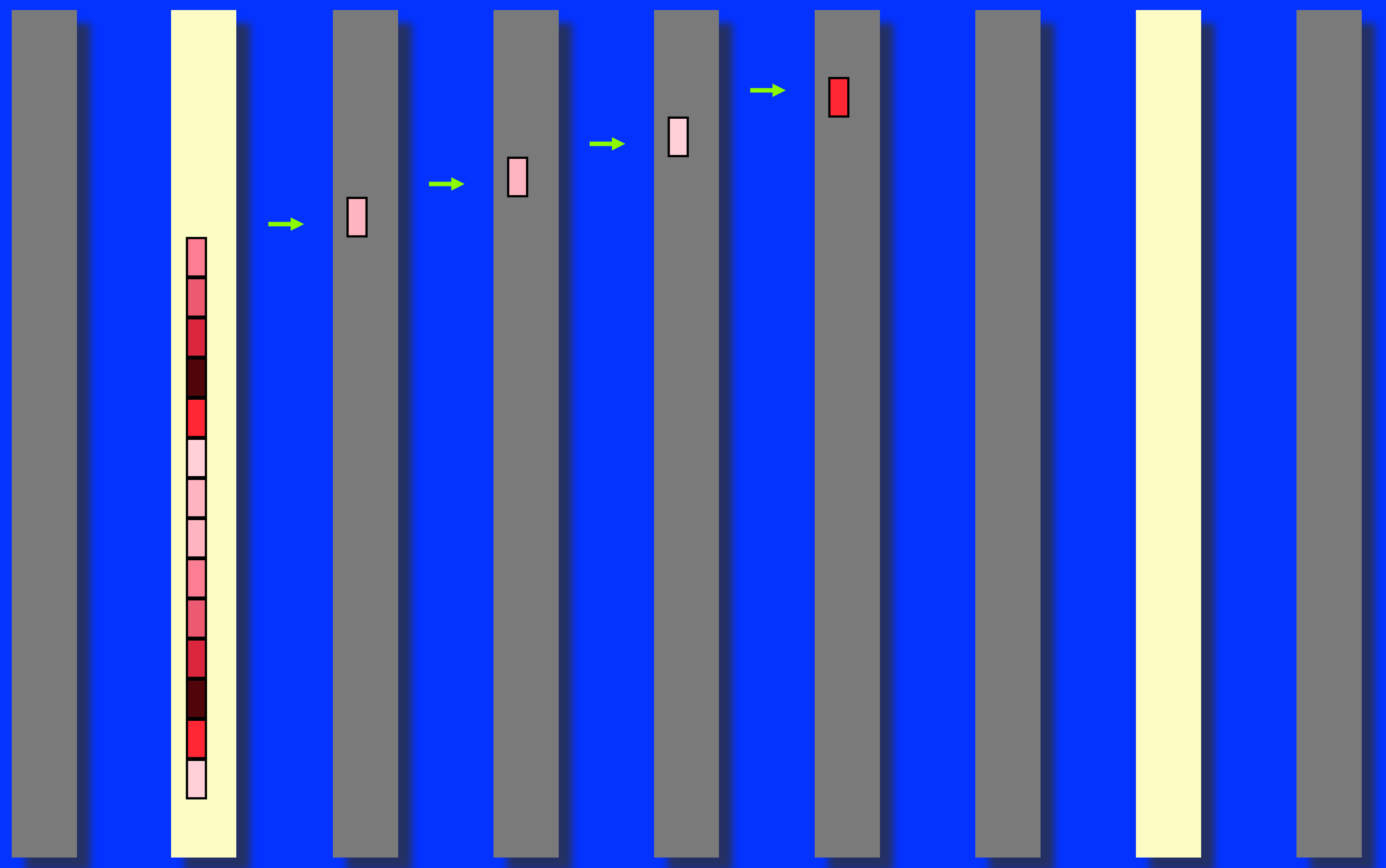


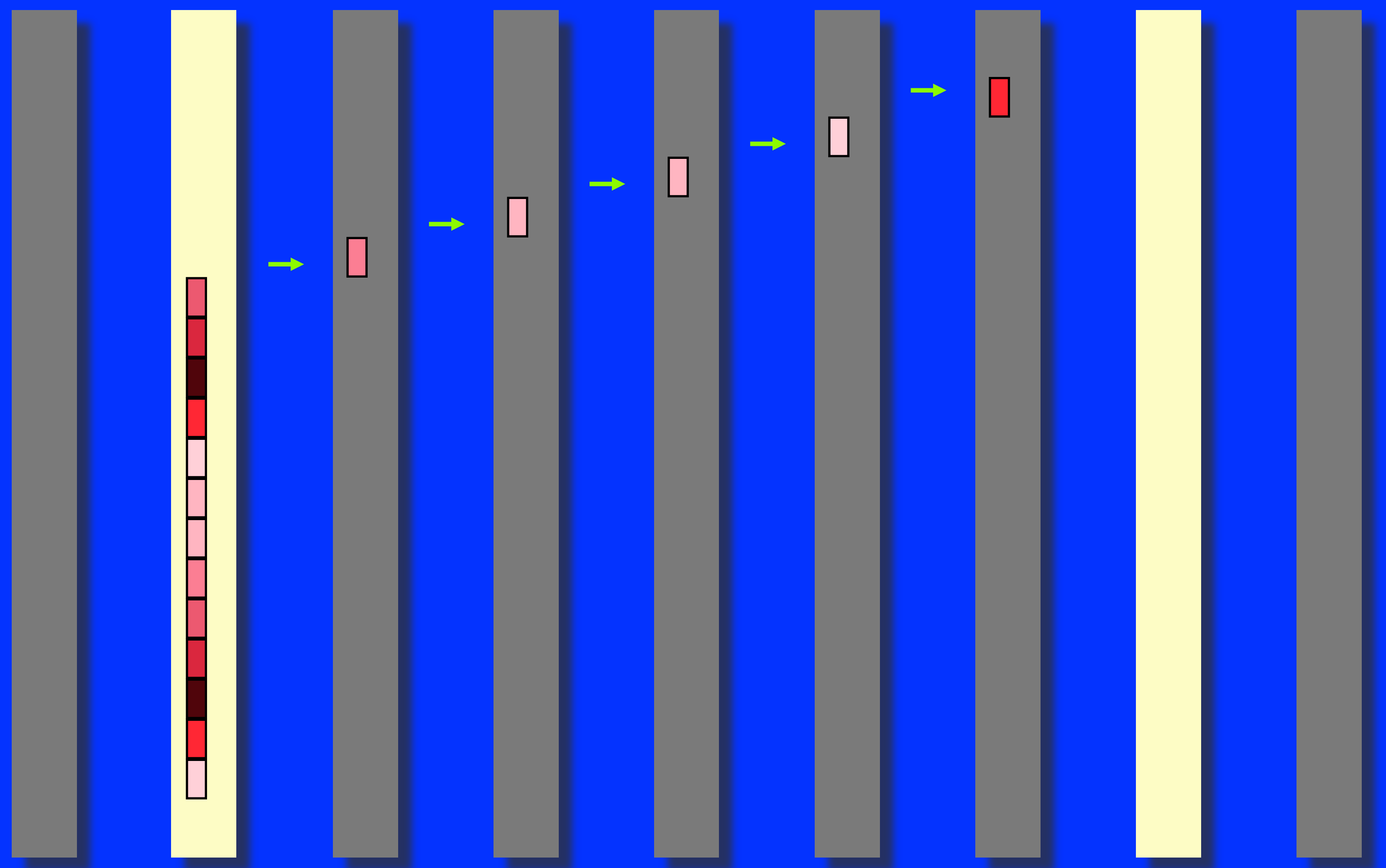


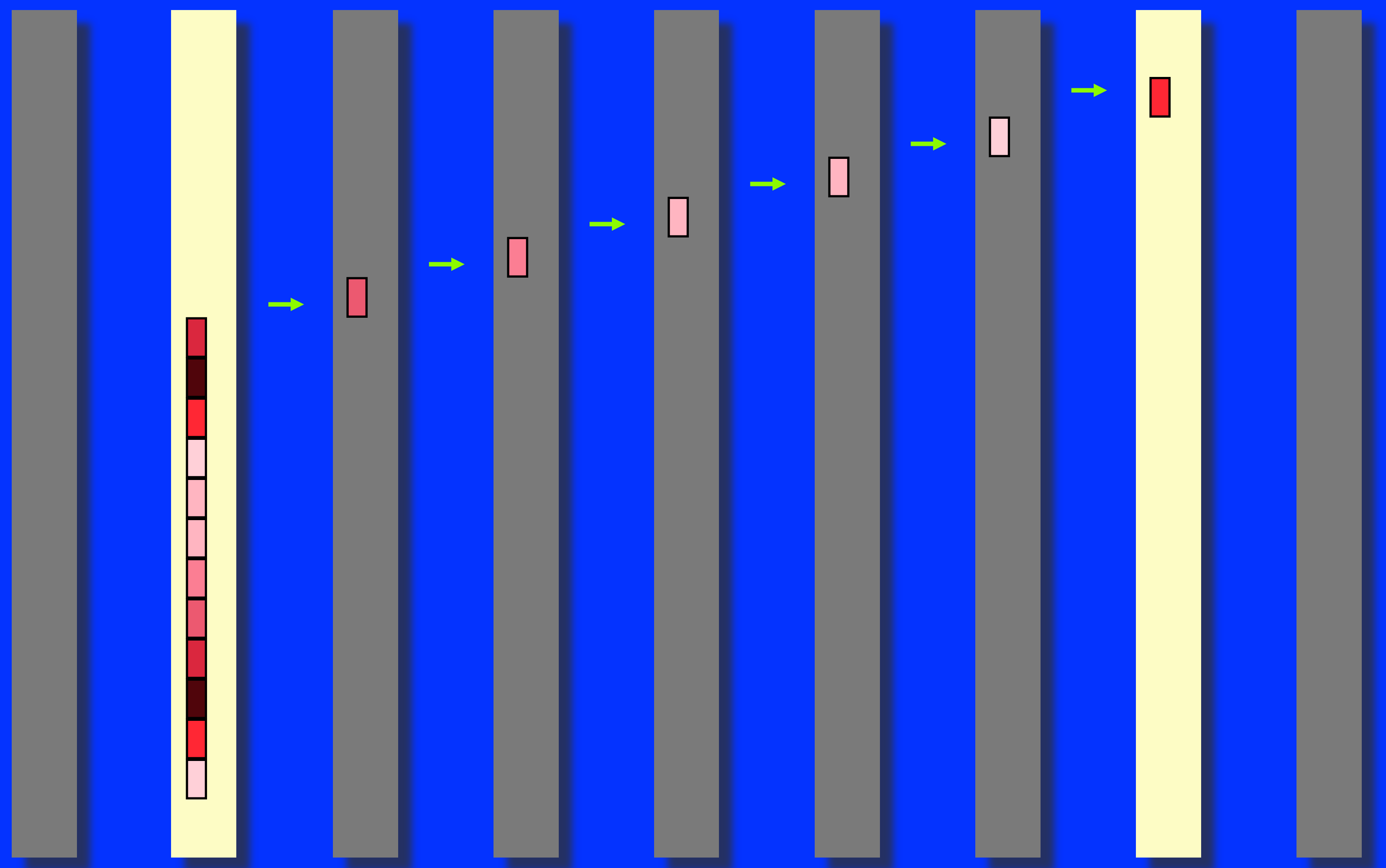


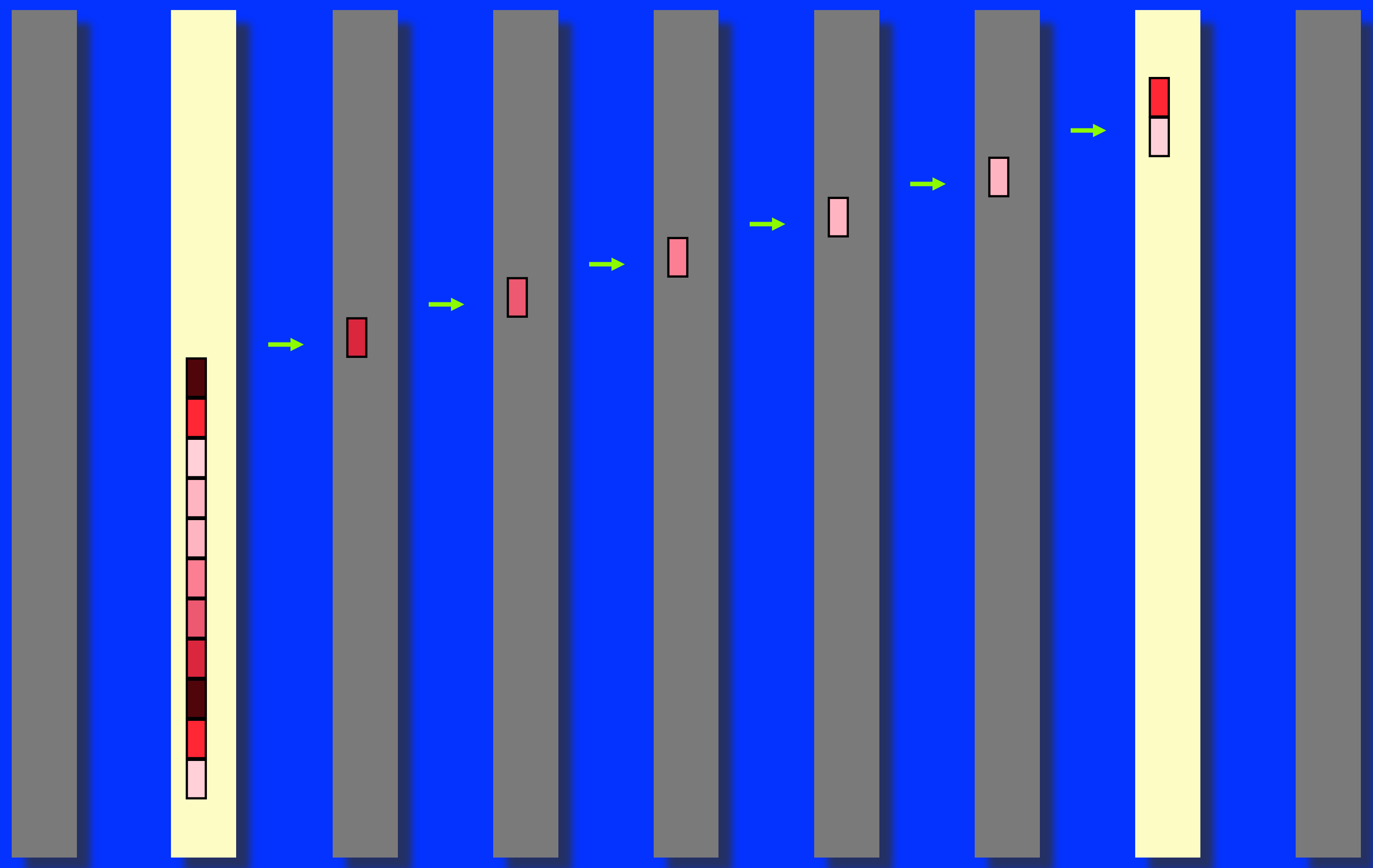


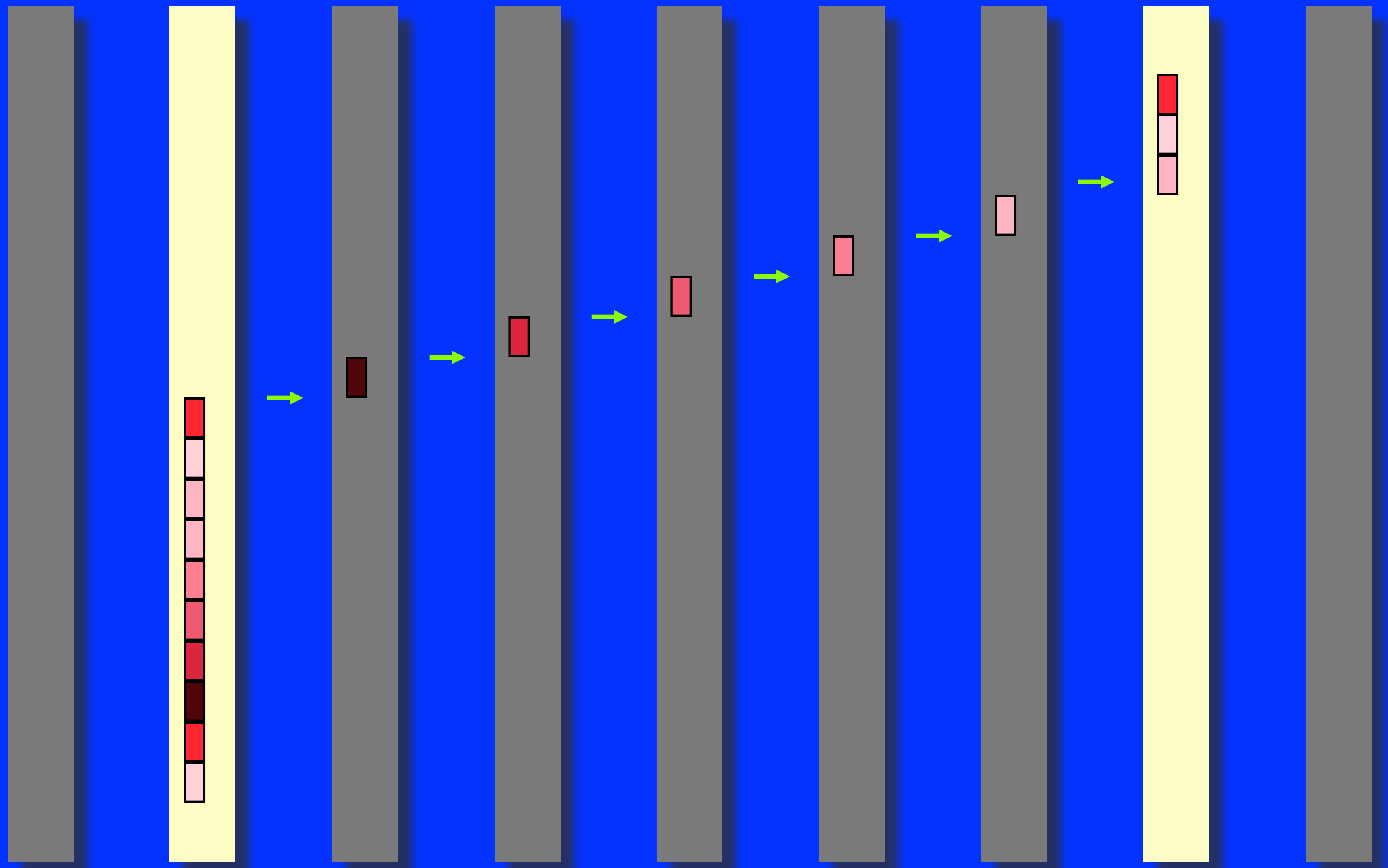


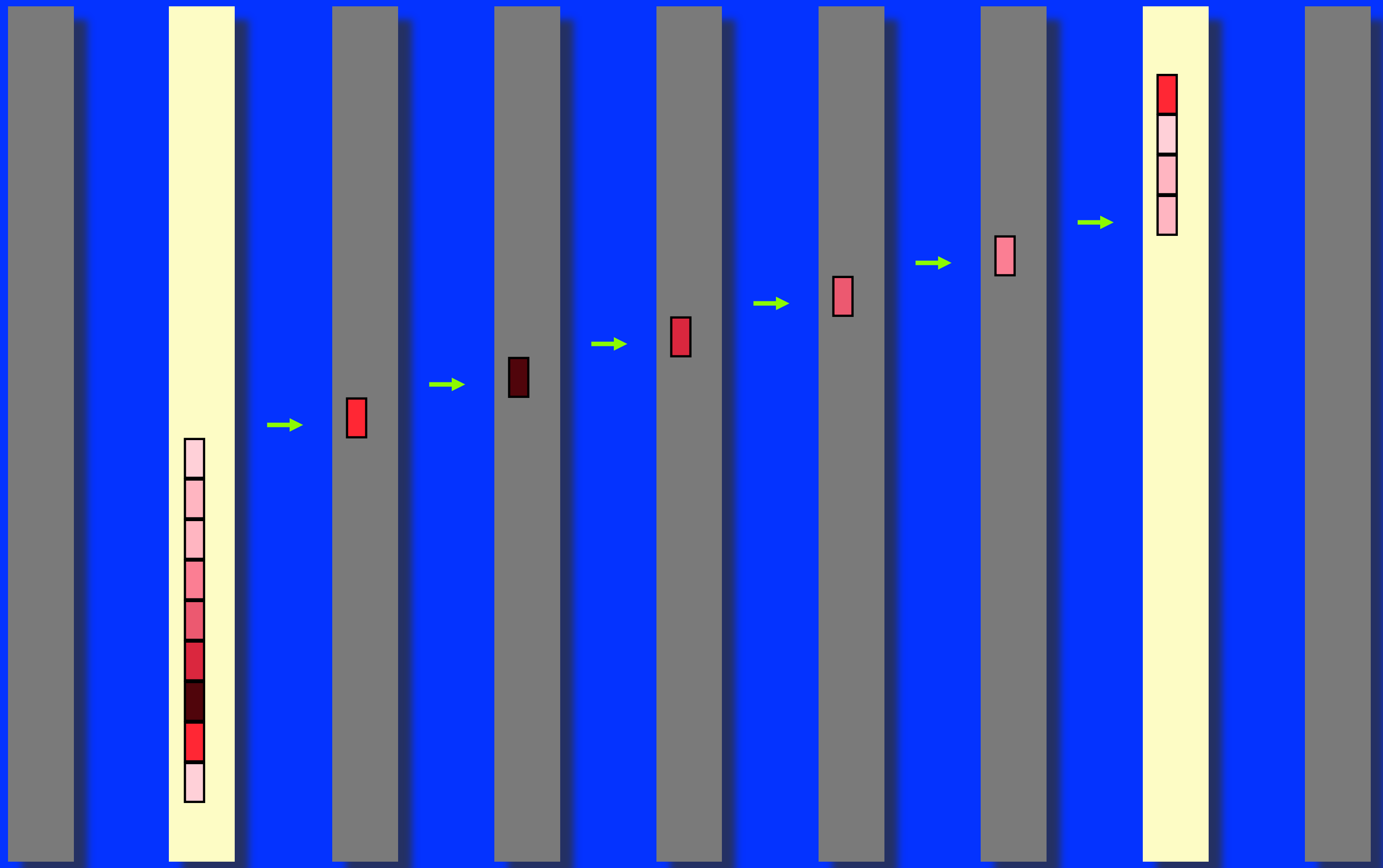


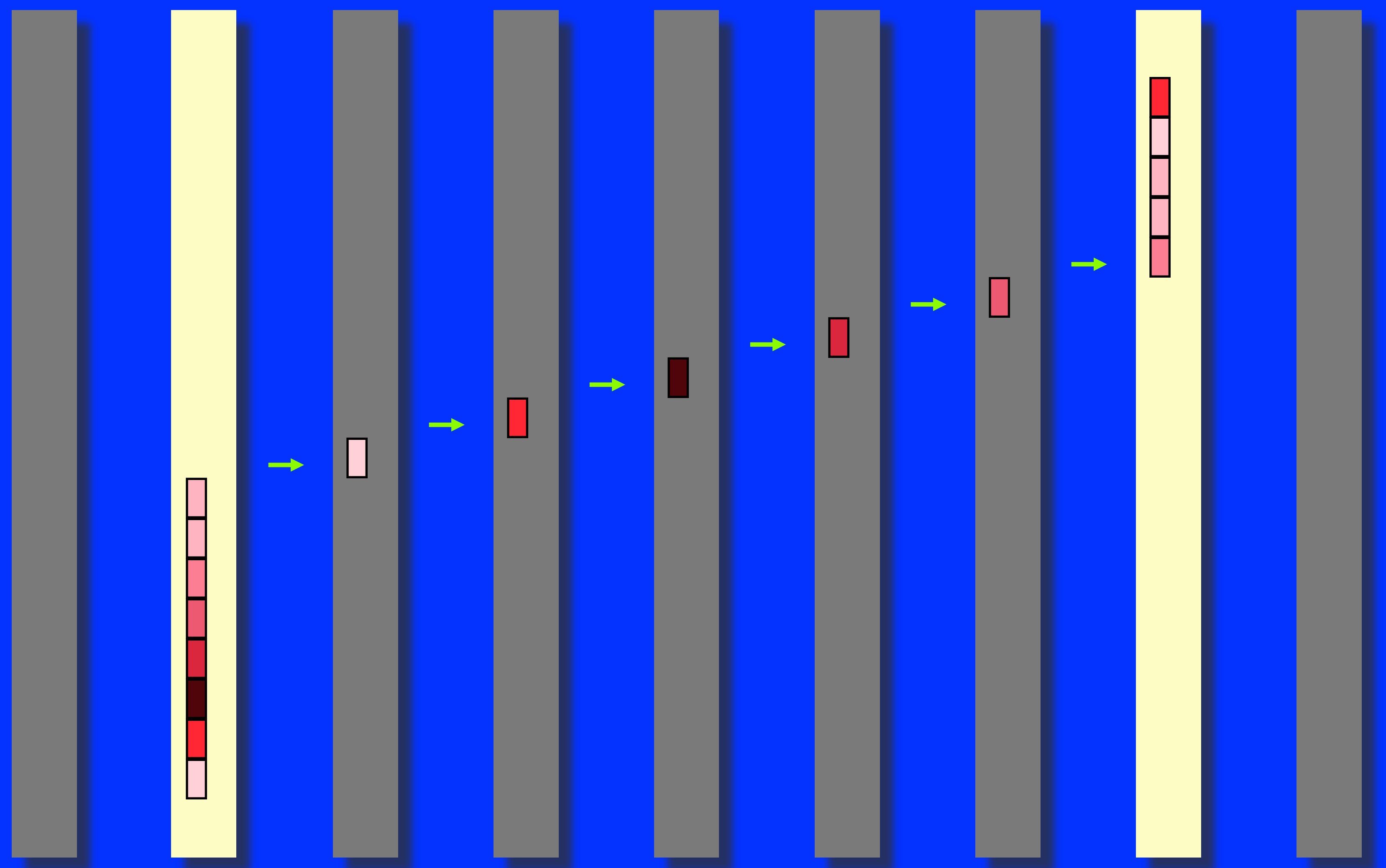


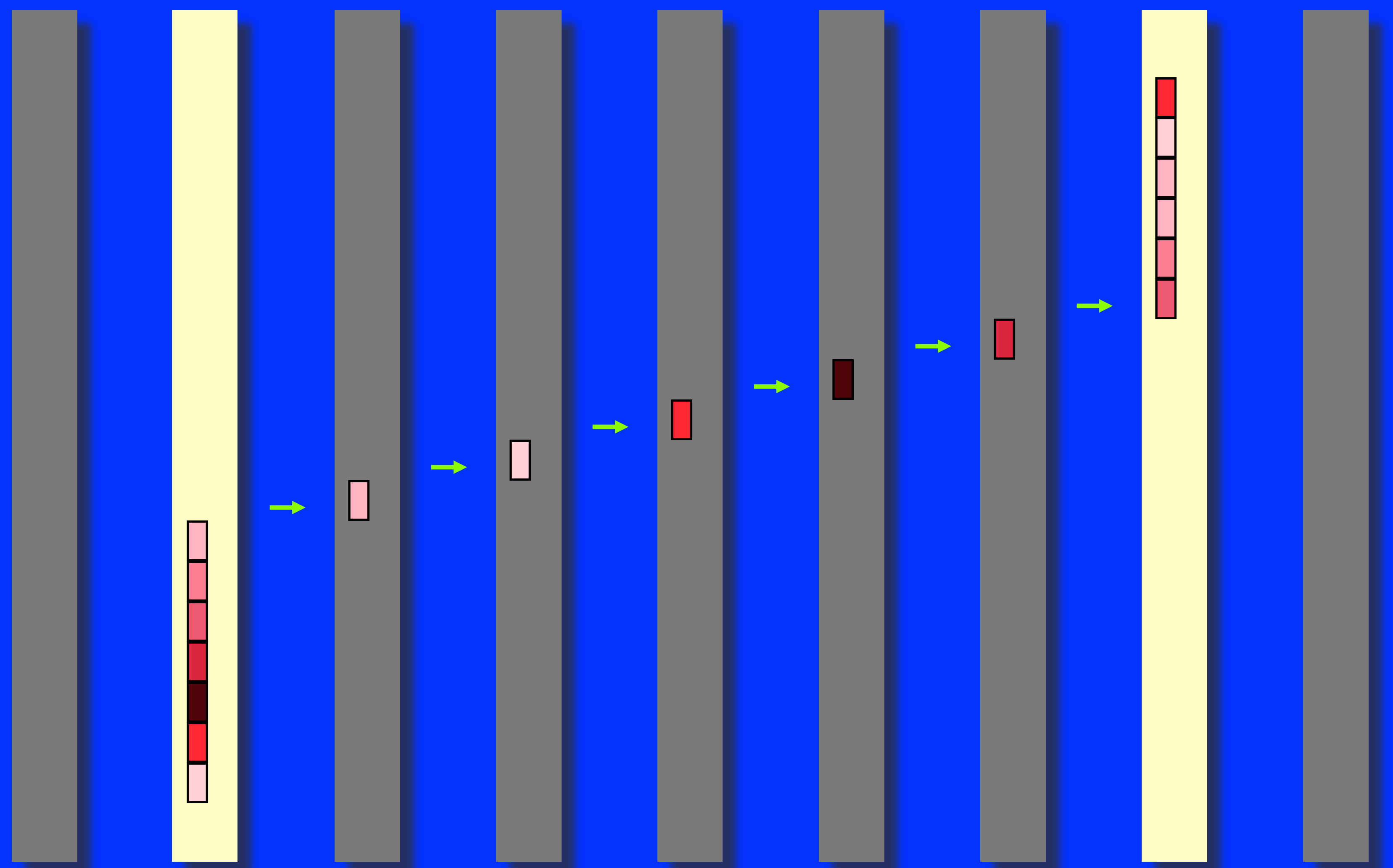


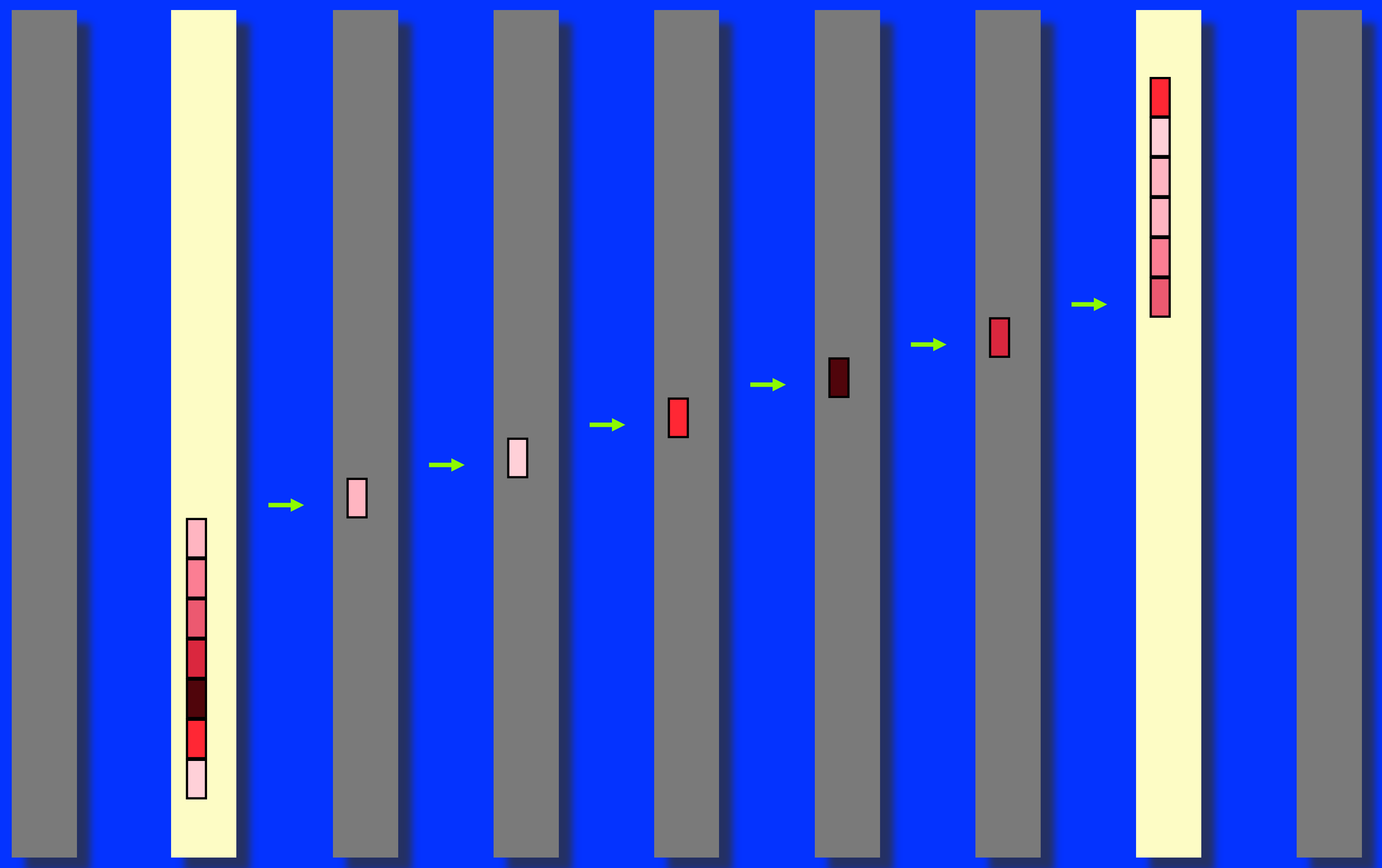


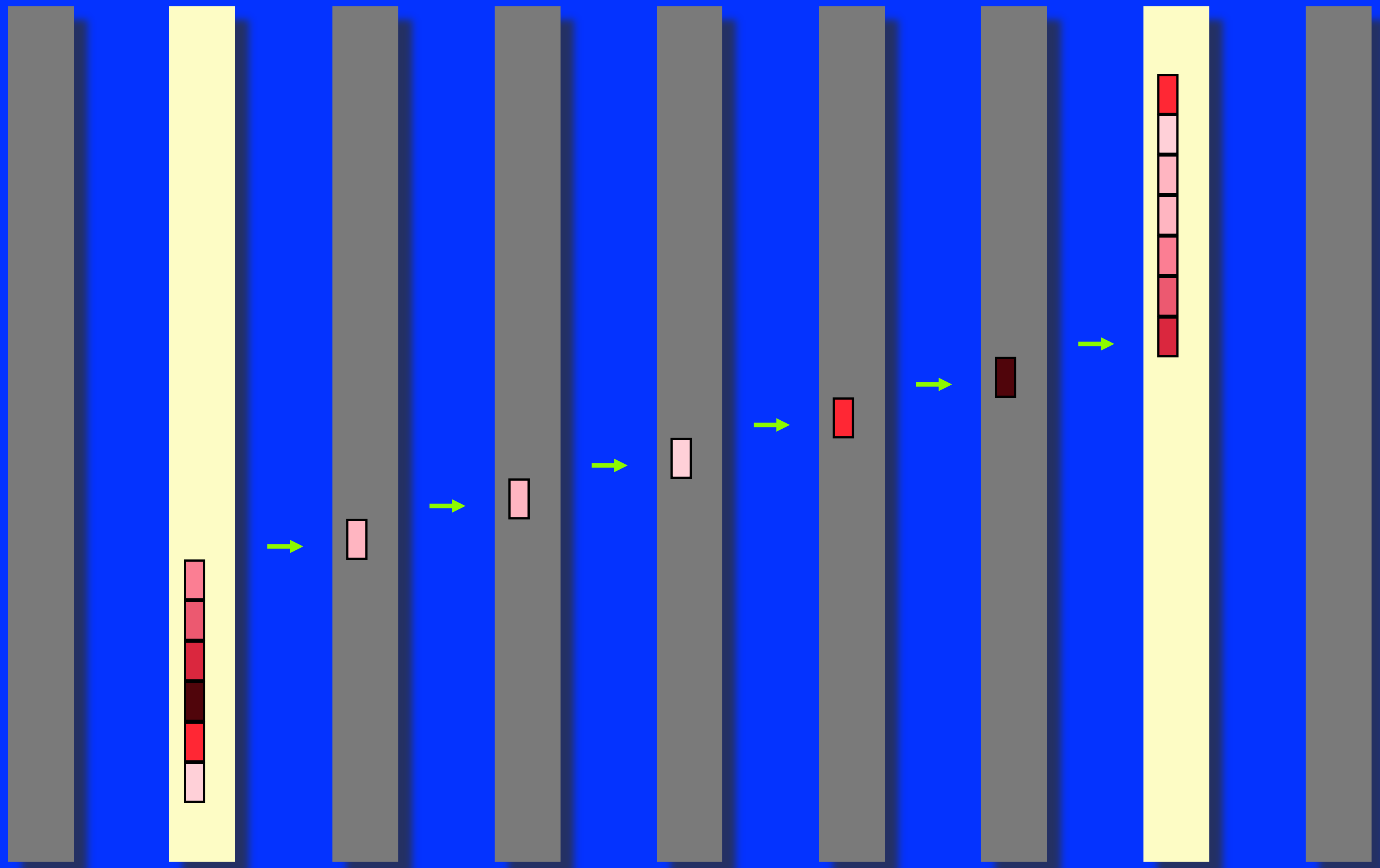


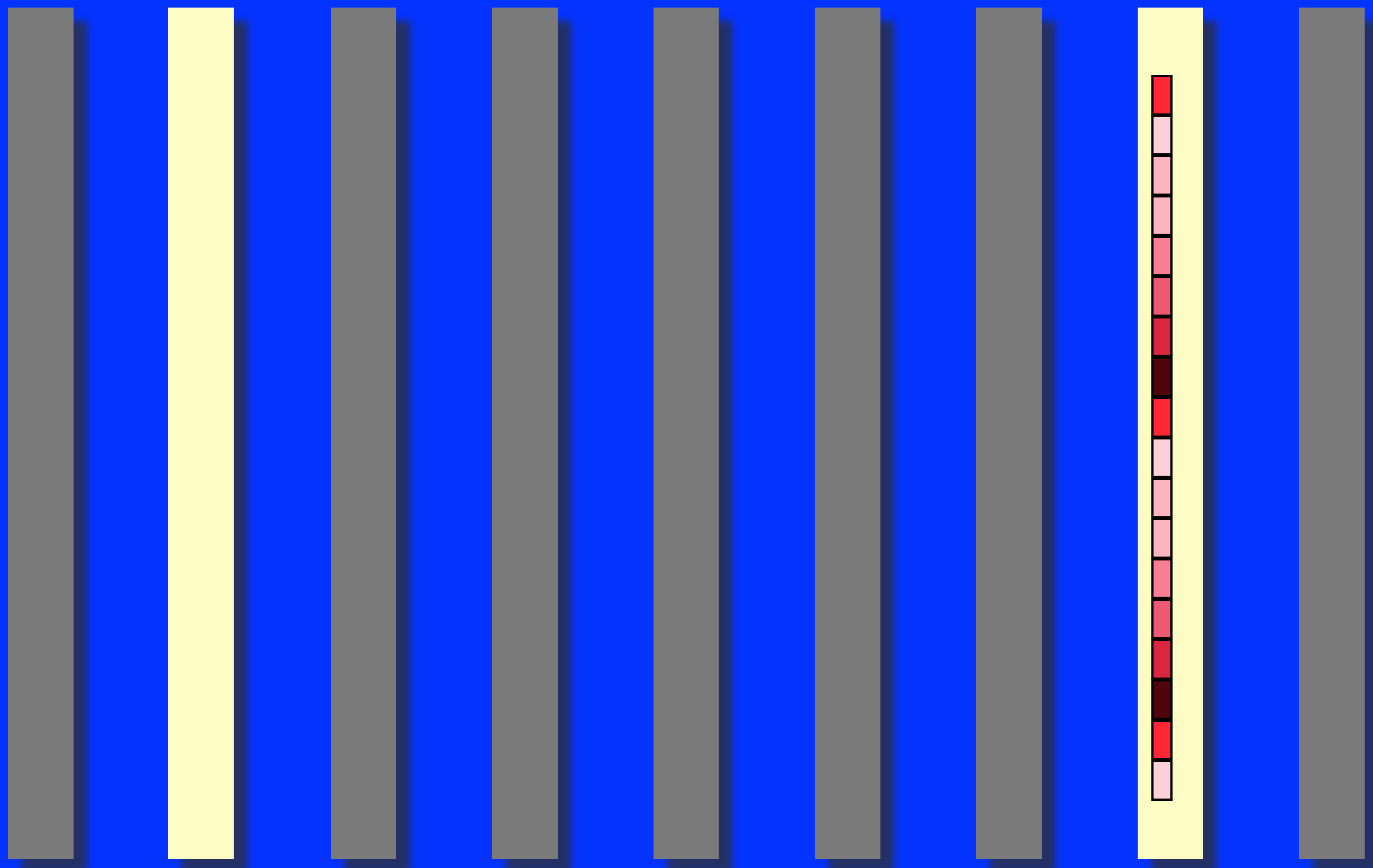












The Cost of Communication

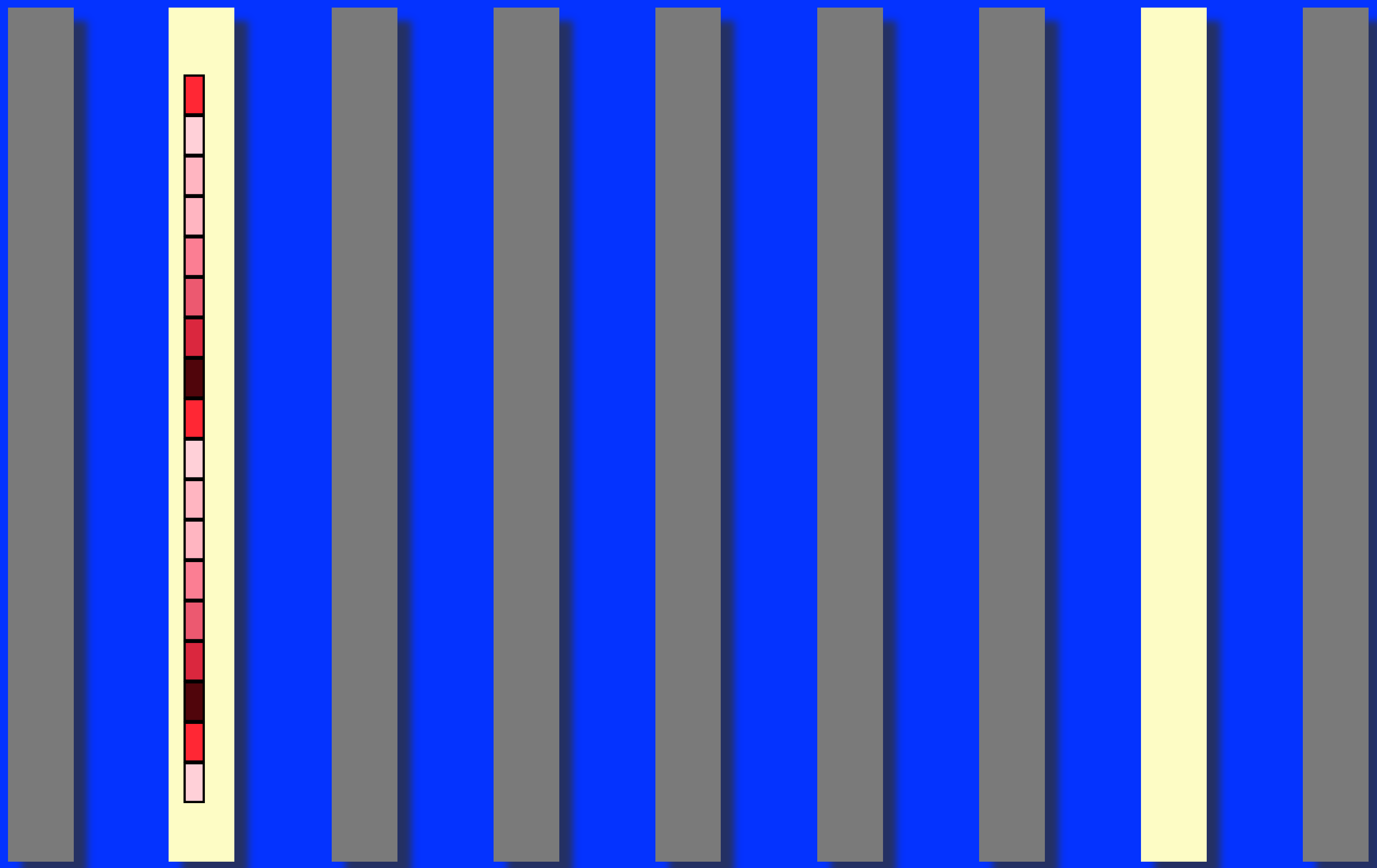
- send a message of length n over d links
- k packets
- Cost:

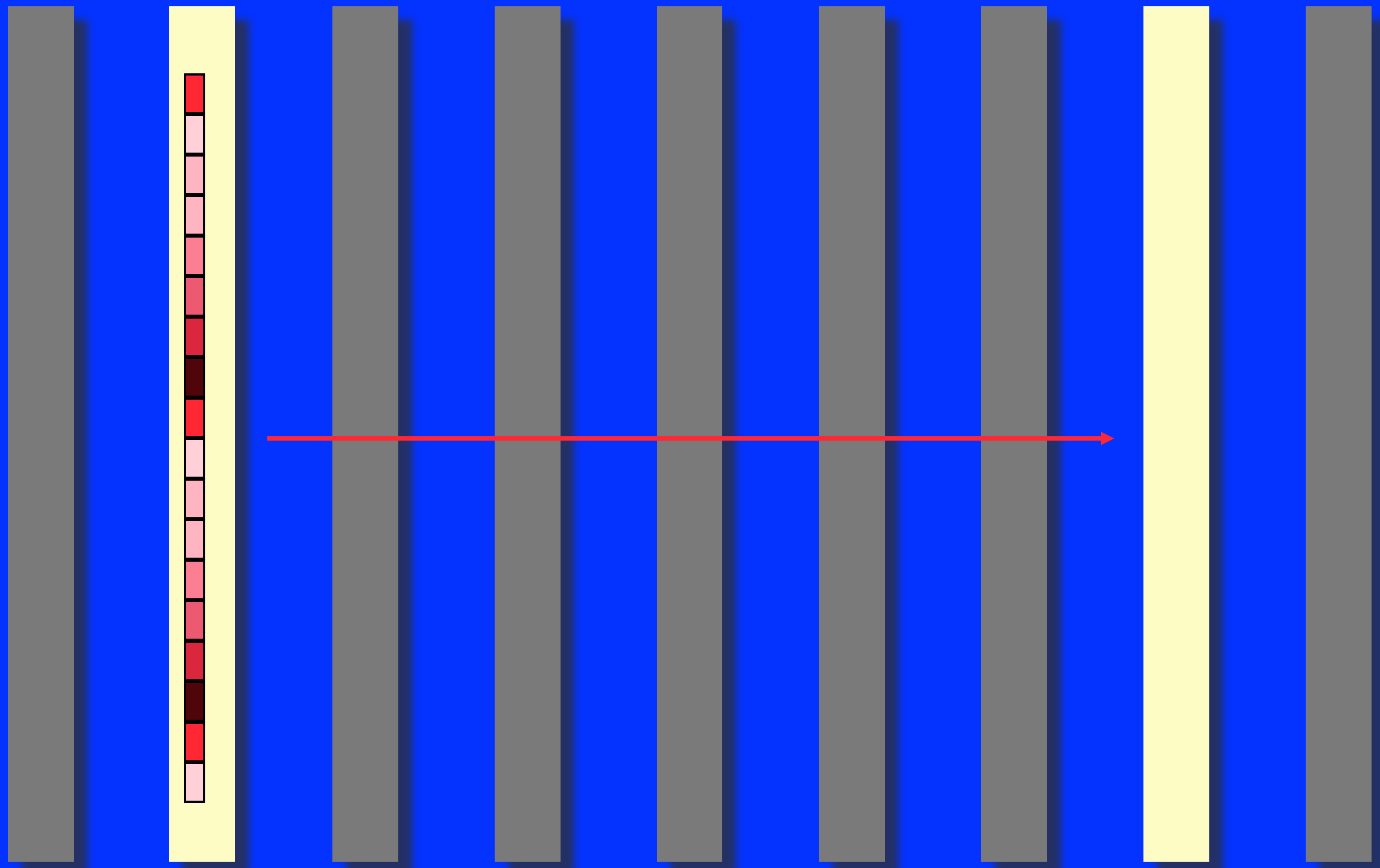
The Cost of Communication

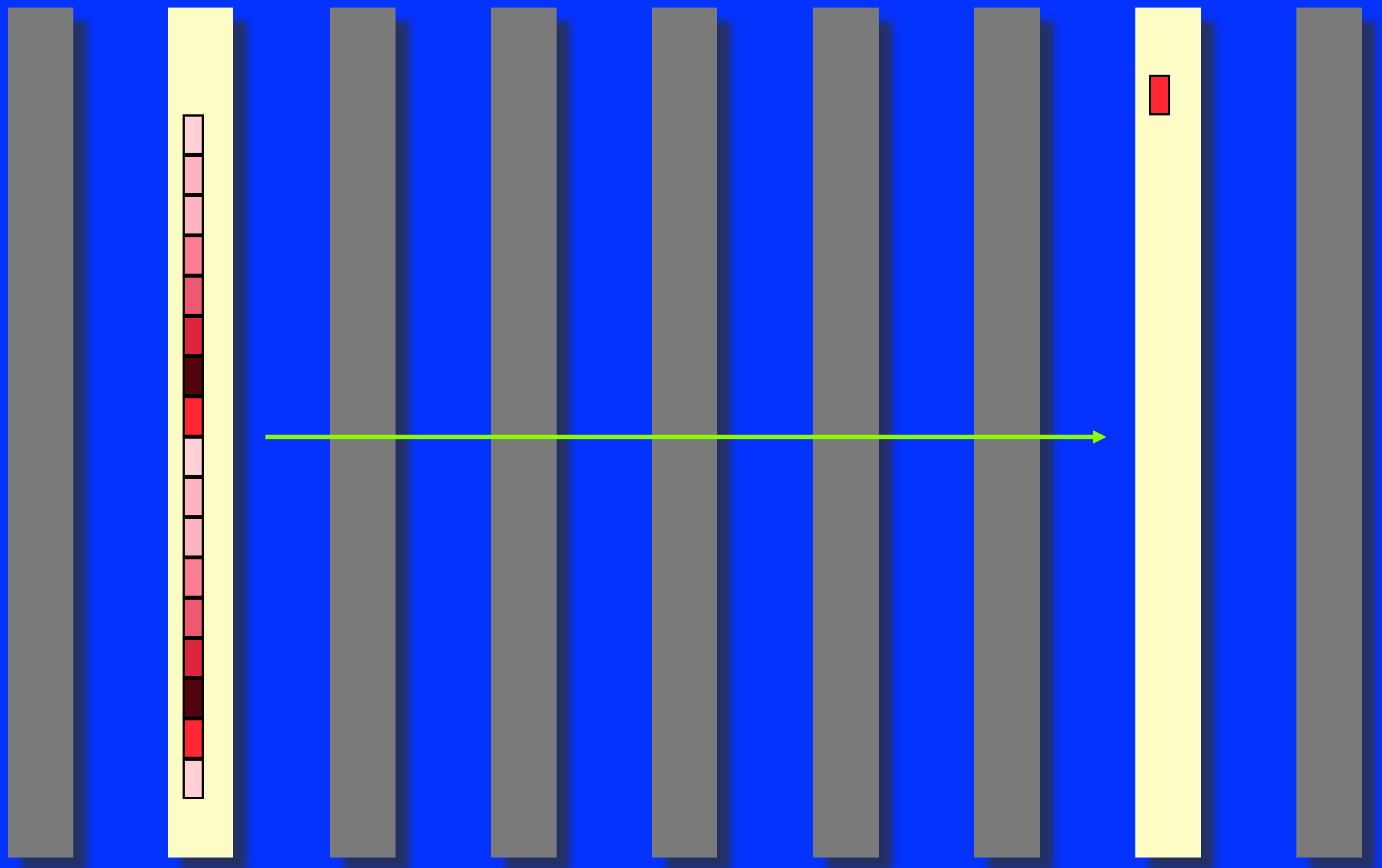
- **send a message of length n over d links**
- **k packets**
- **Cost:**

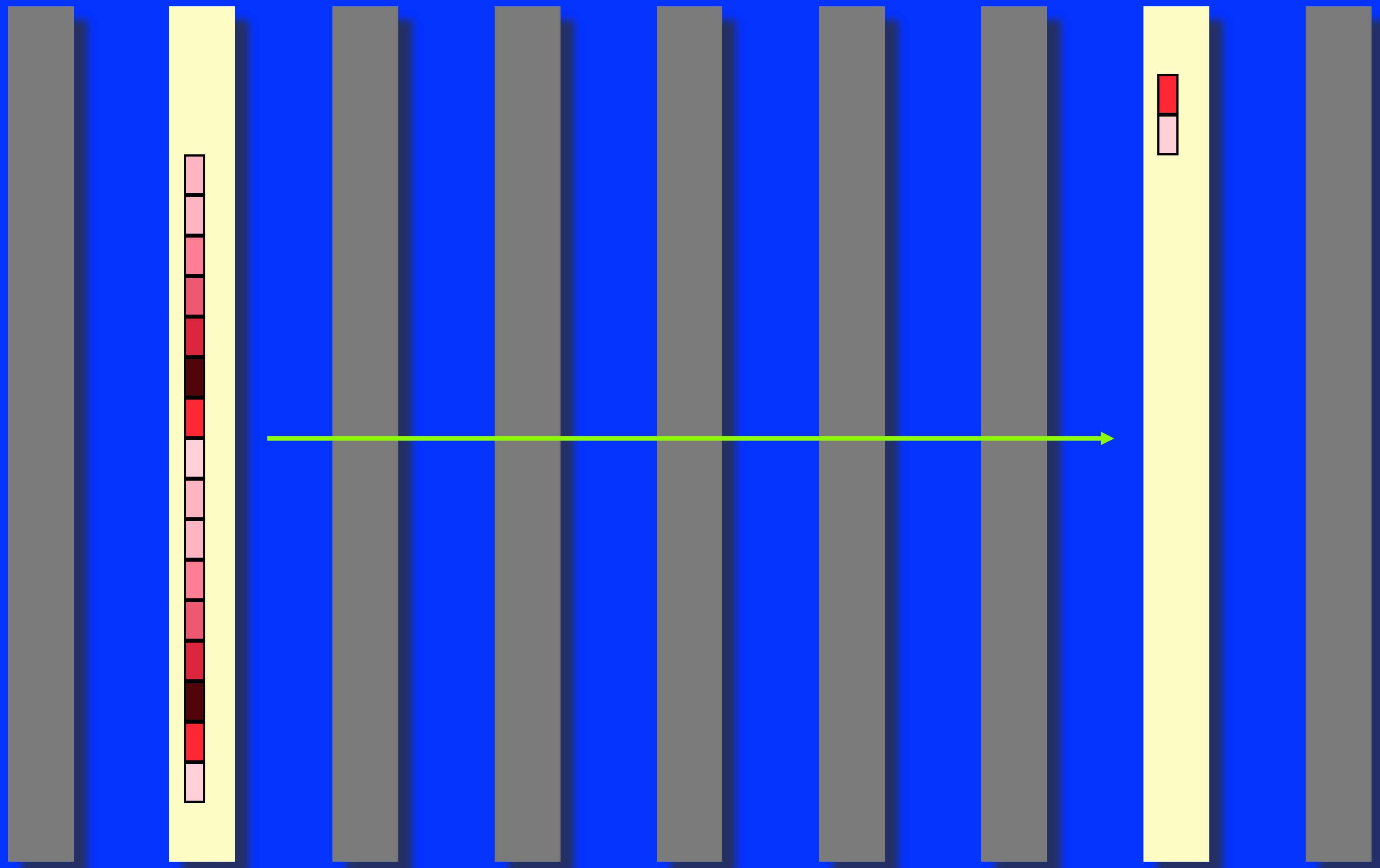
$$\alpha + d\left(\alpha_{net} + \frac{n}{k}\beta\right) + (k-1)\left(\alpha_{net} + \frac{n}{k}\beta\right)$$
$$=$$
$$\alpha + n\beta + (d + k - 1)\alpha_{net} + \frac{d-1}{k}\, n\beta$$
$$\approx$$
$$\alpha + n\beta$$

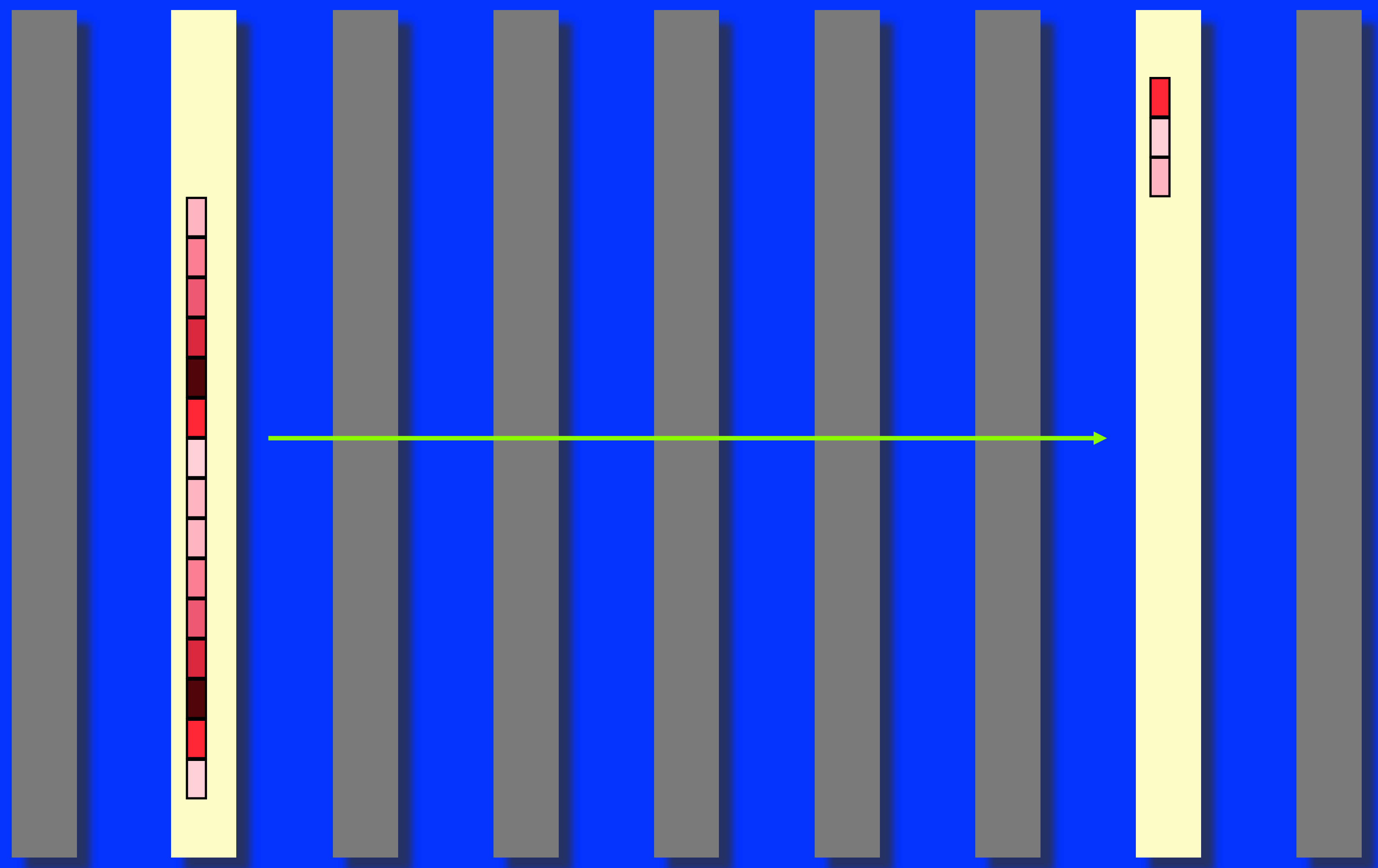
- **Example revisited ...**

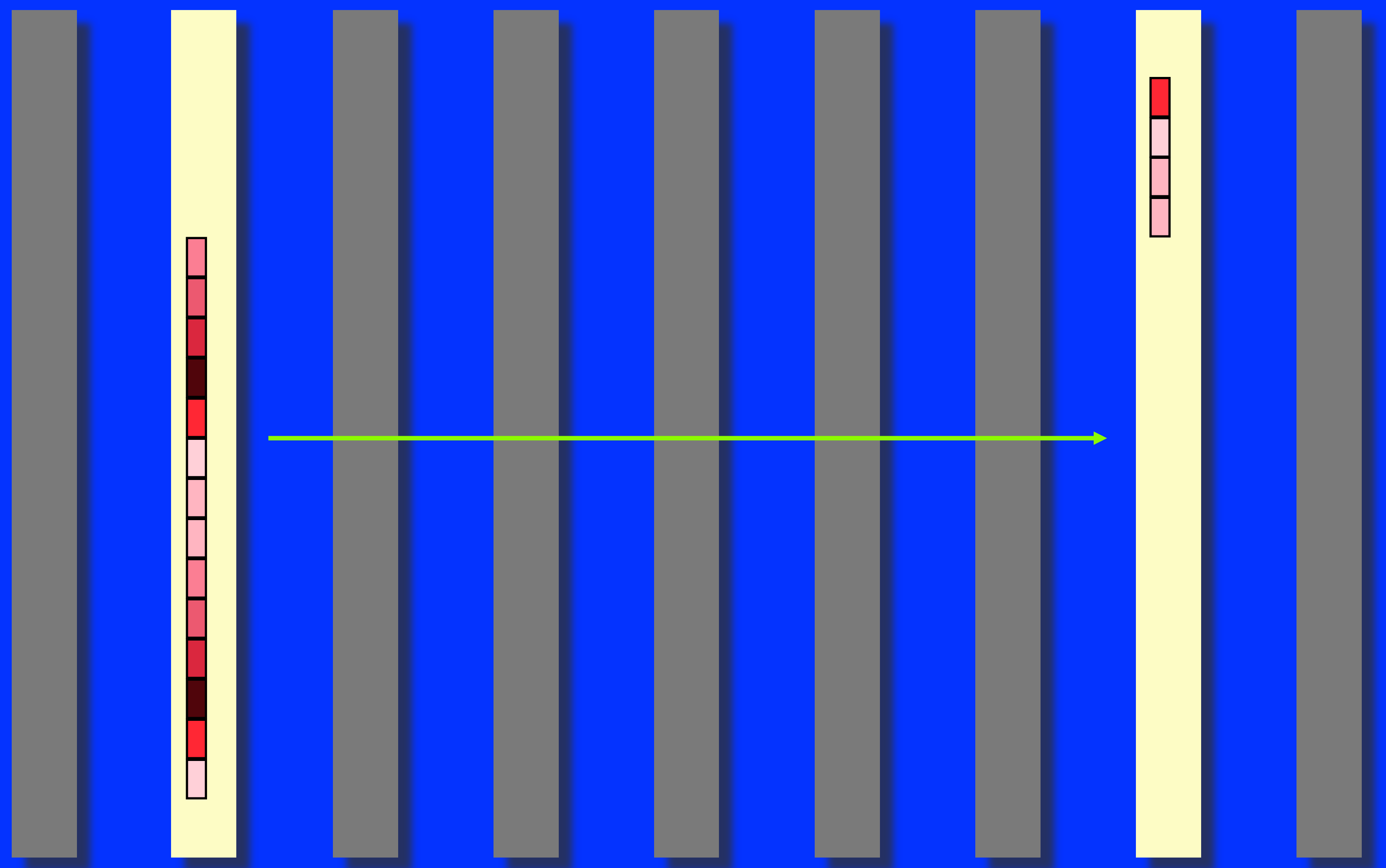


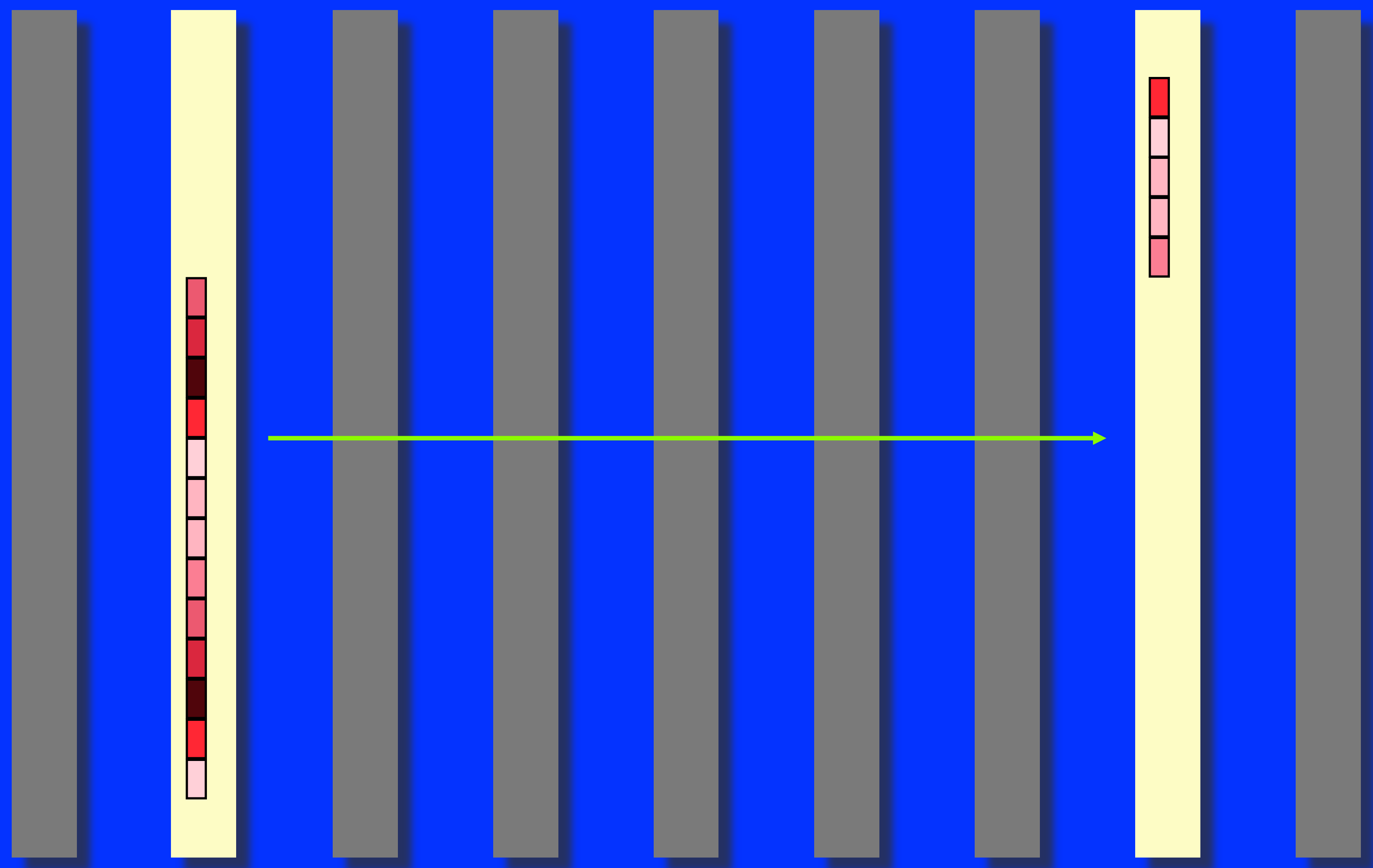


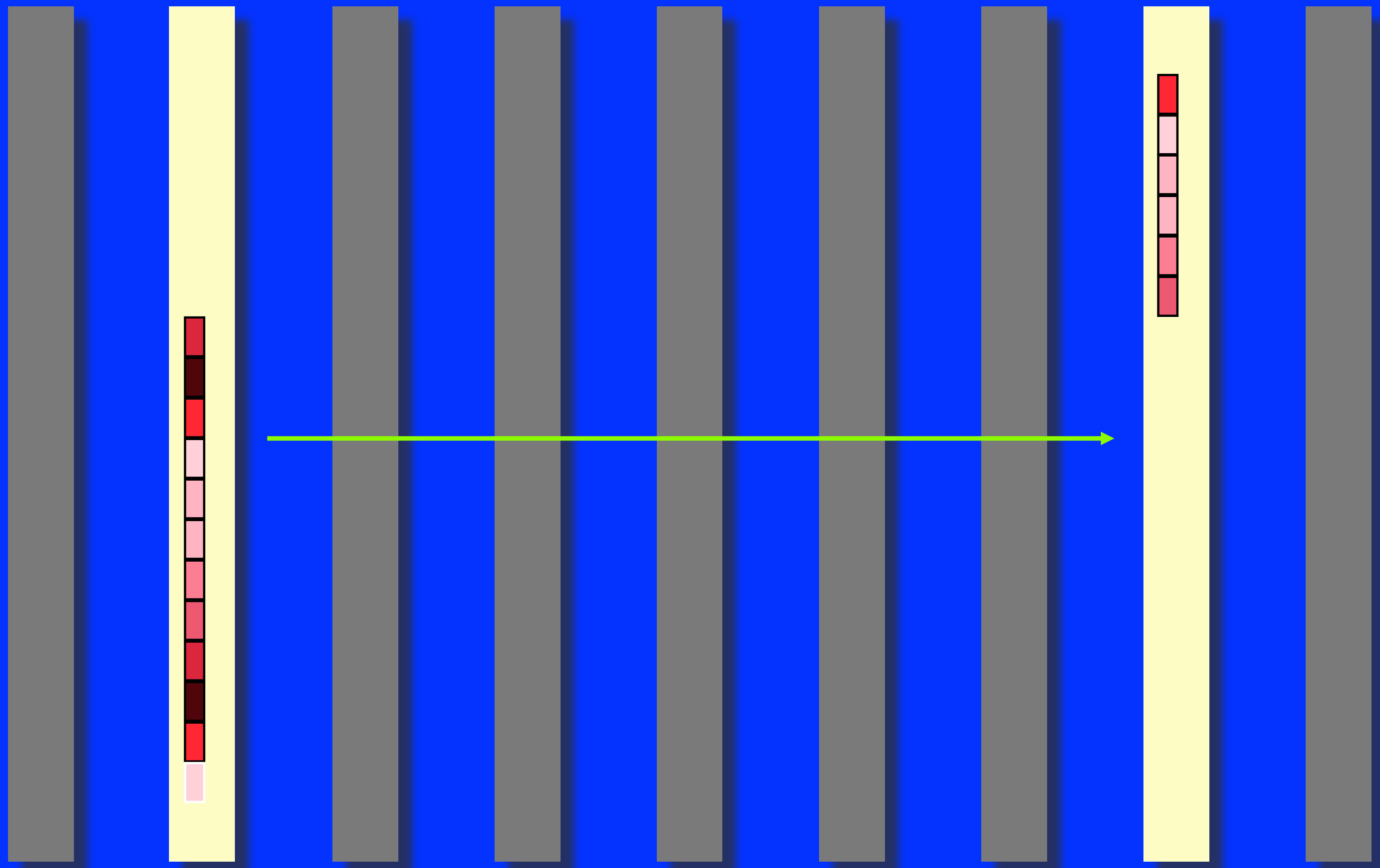


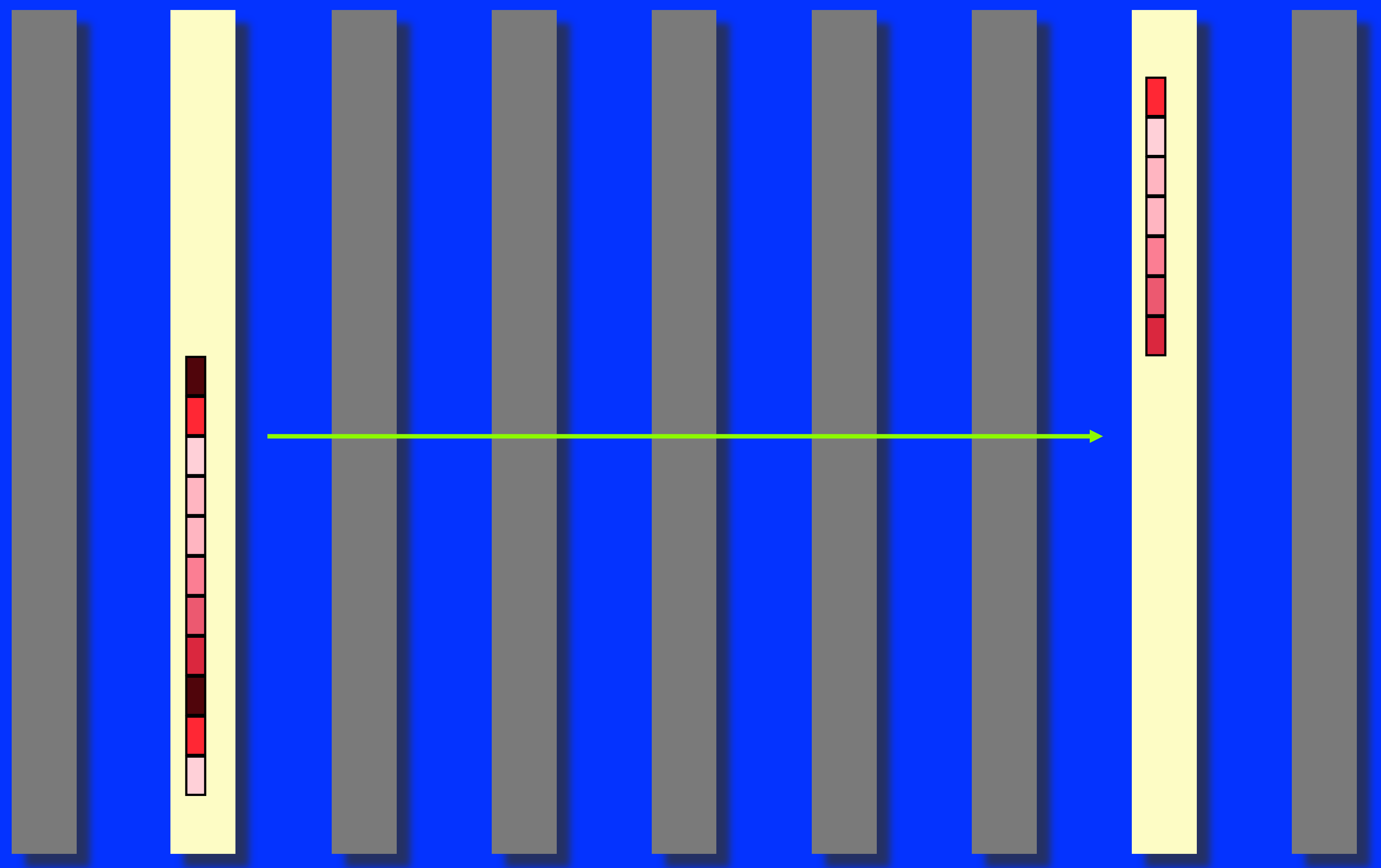


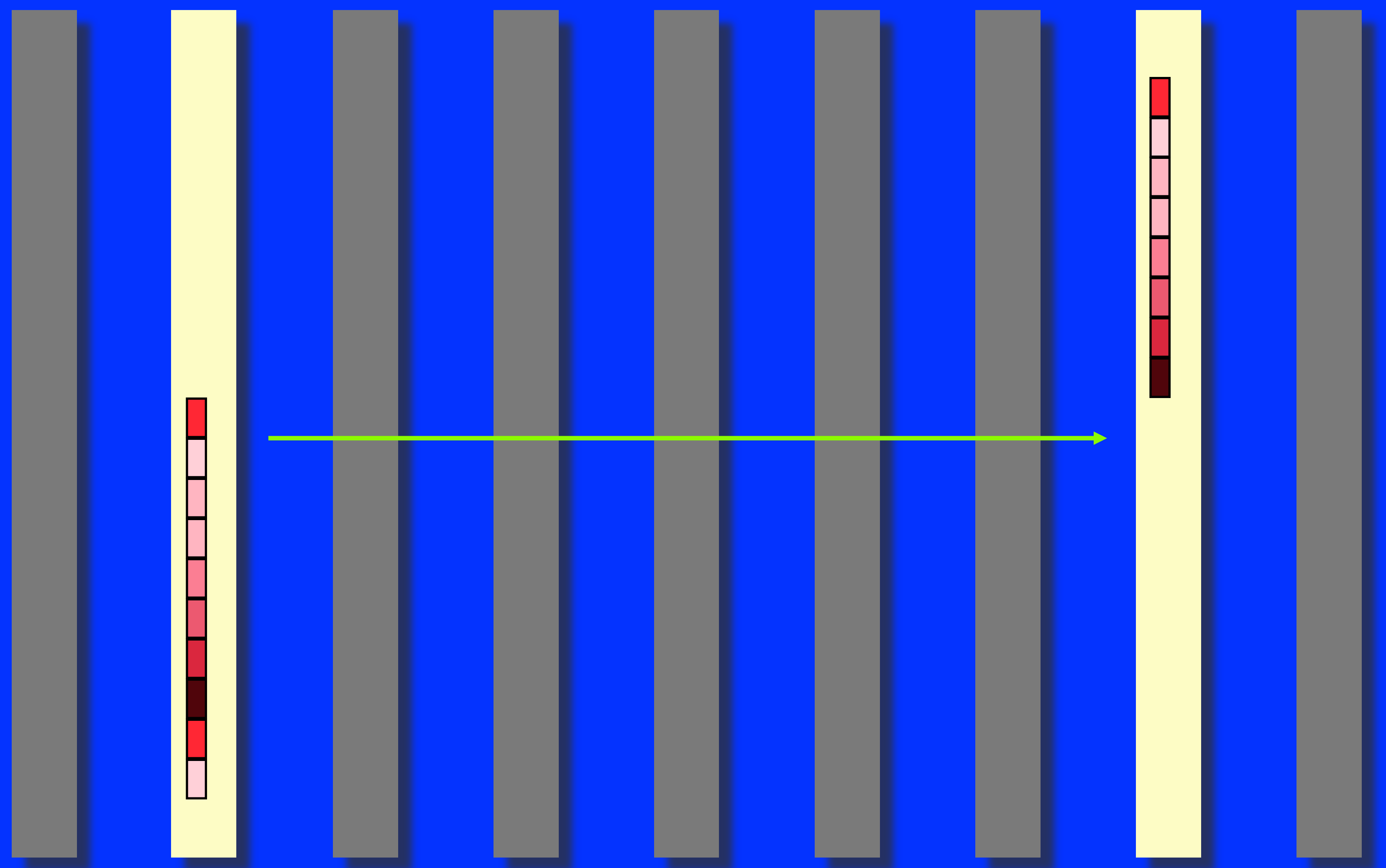


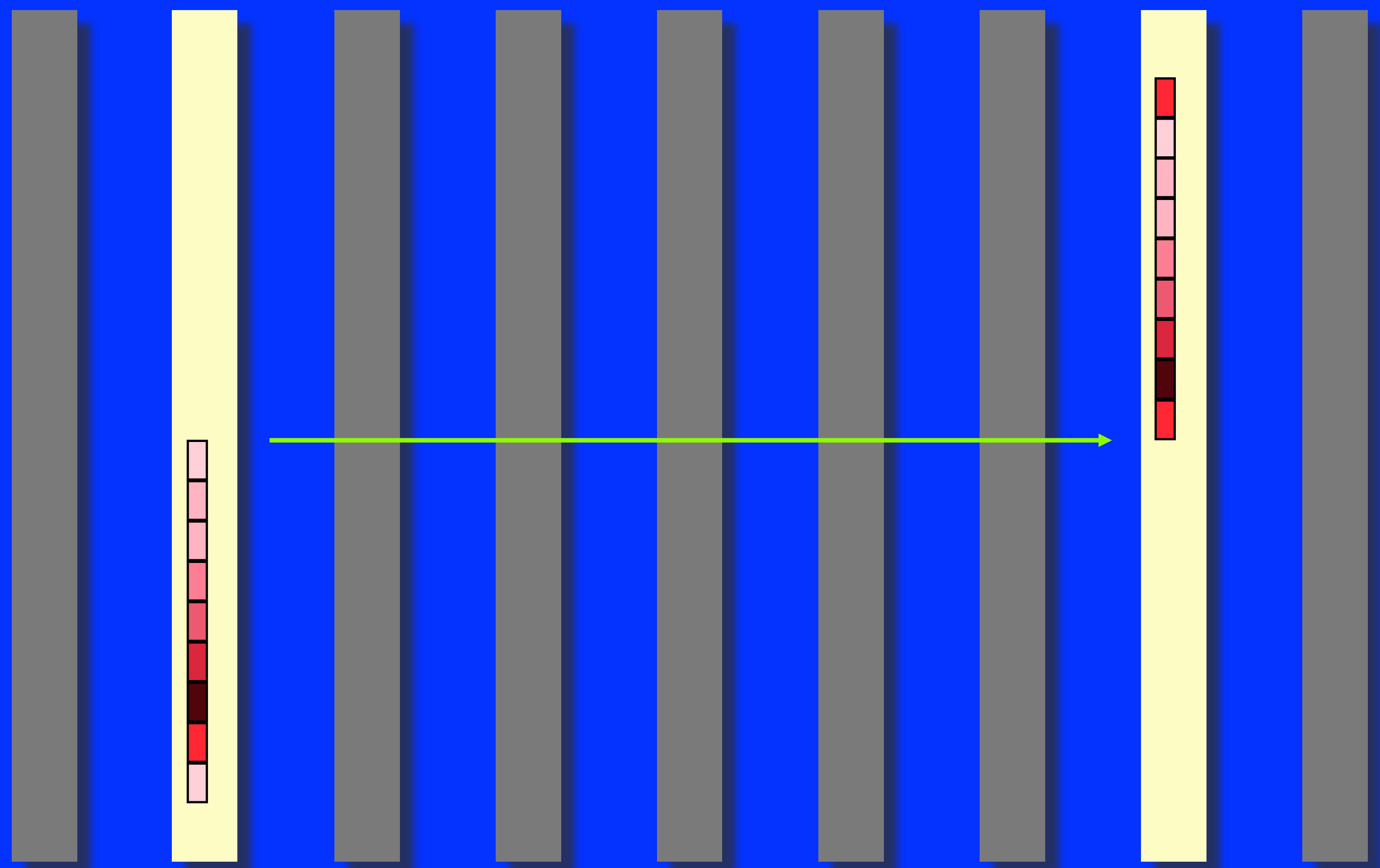


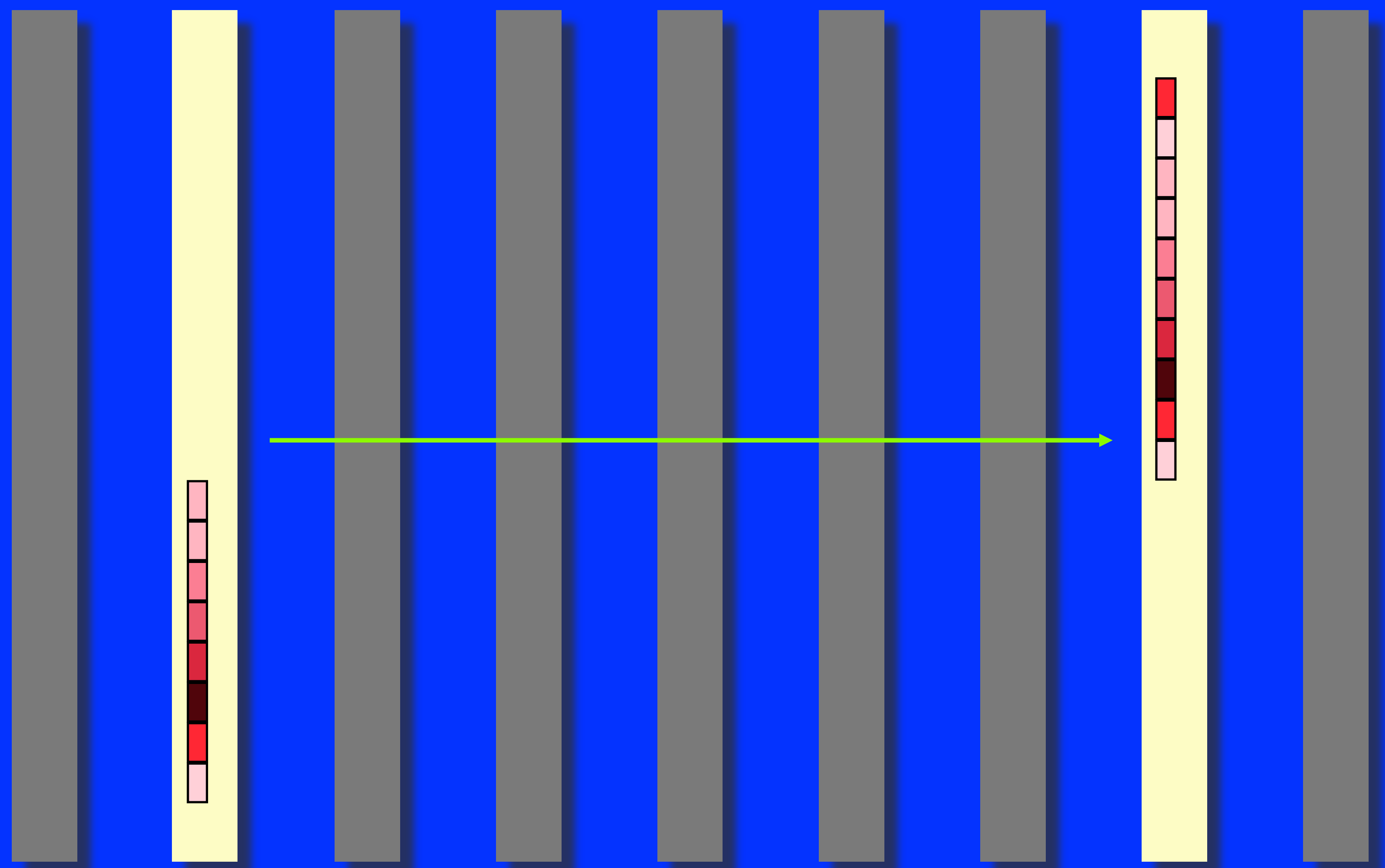


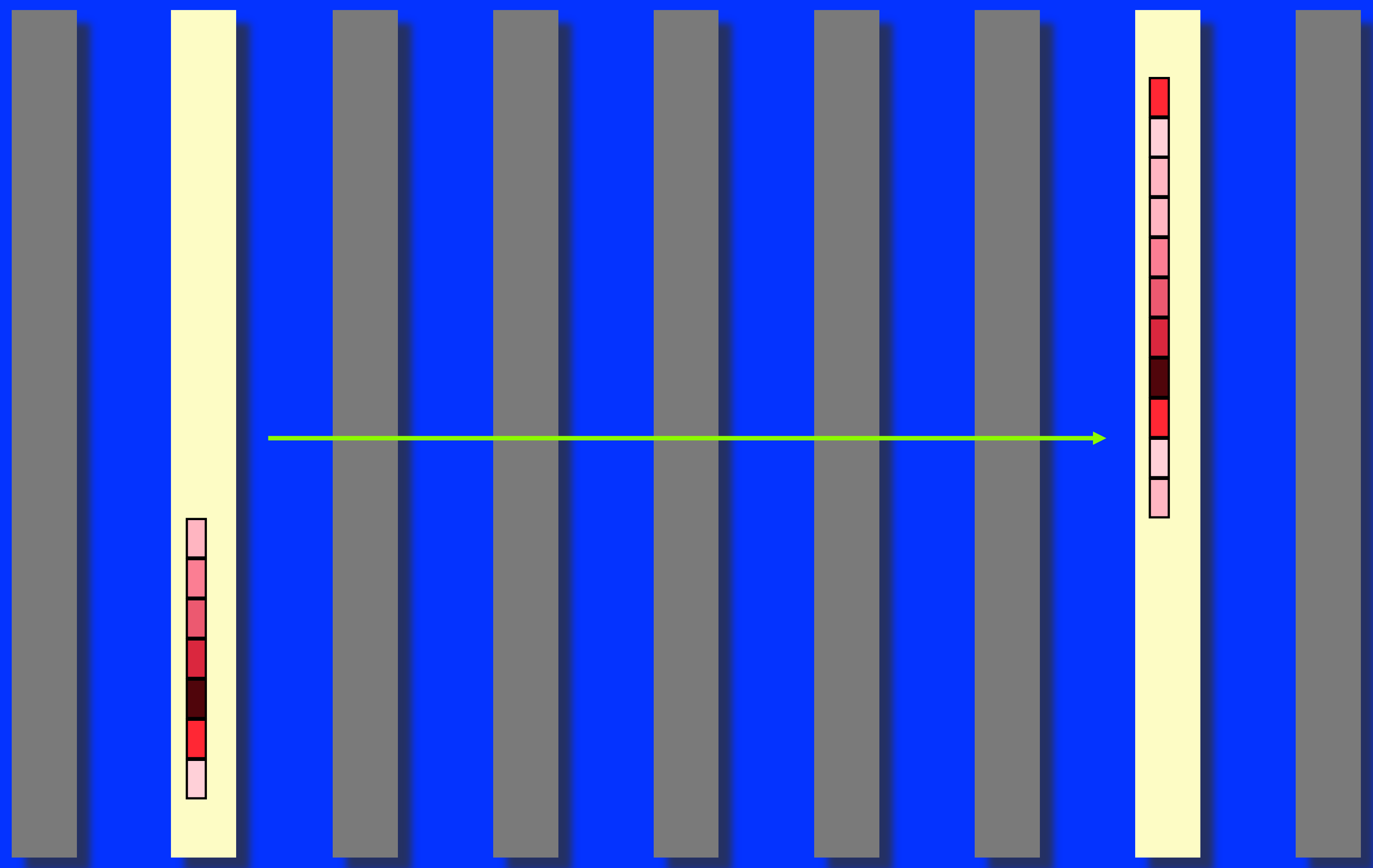


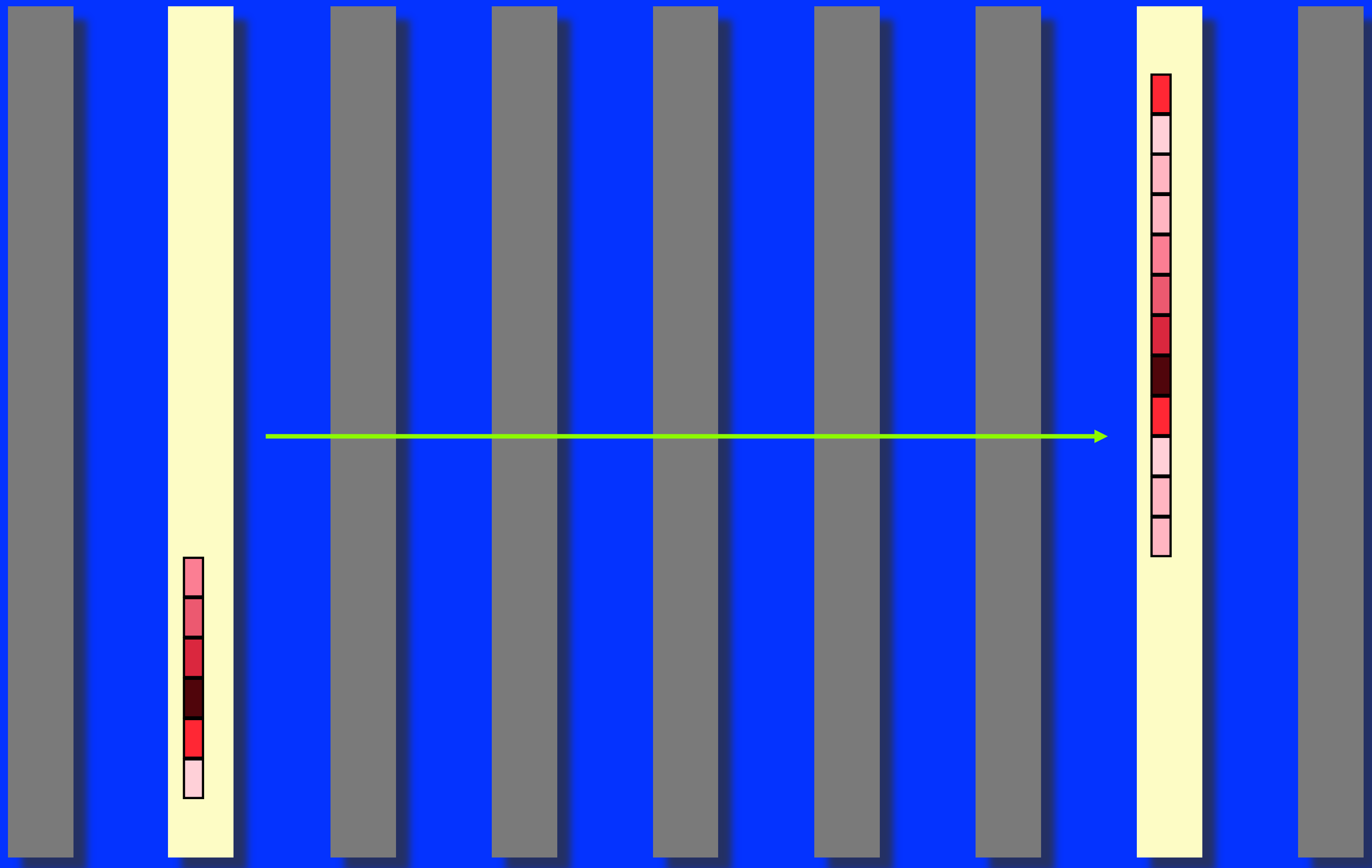


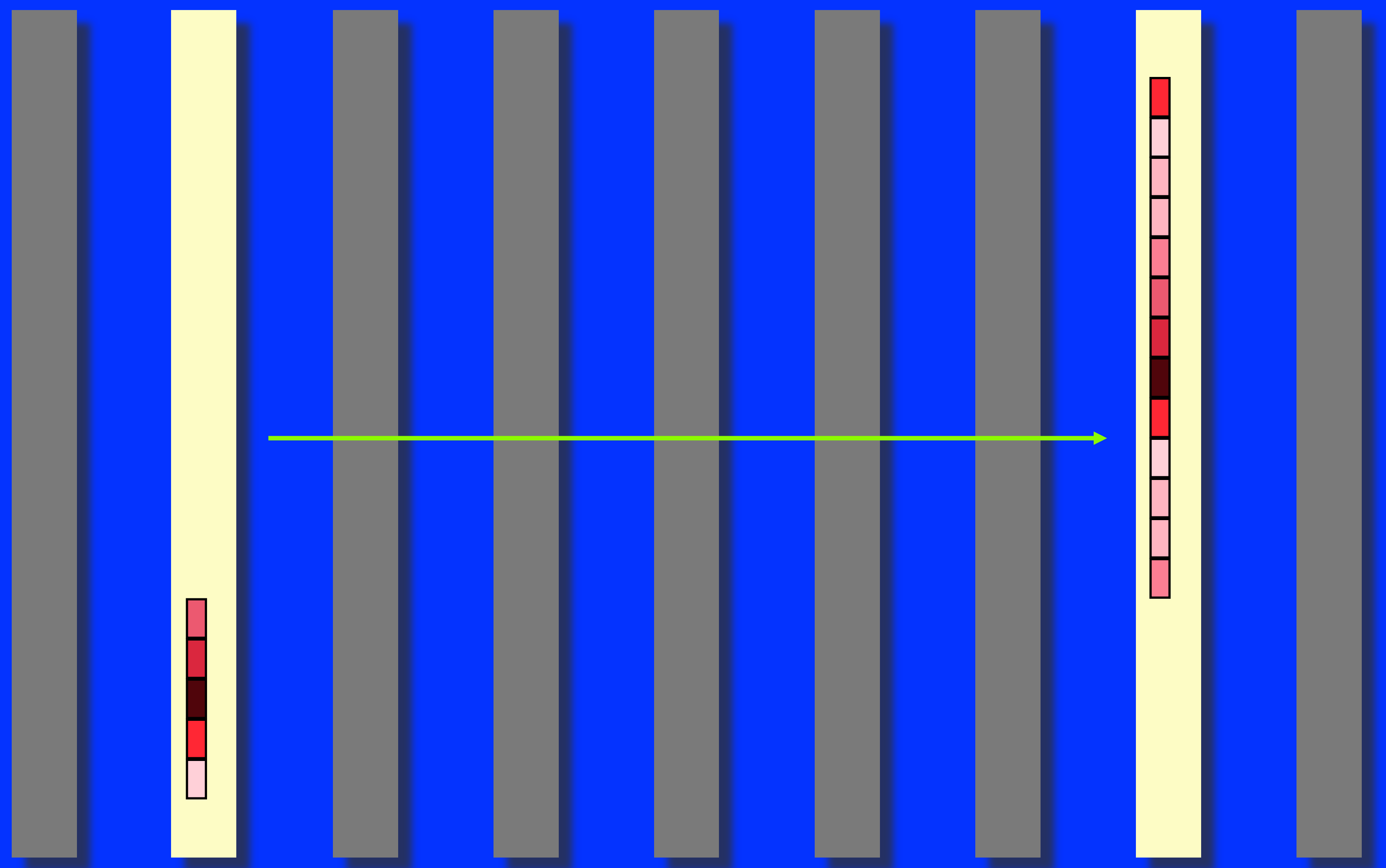


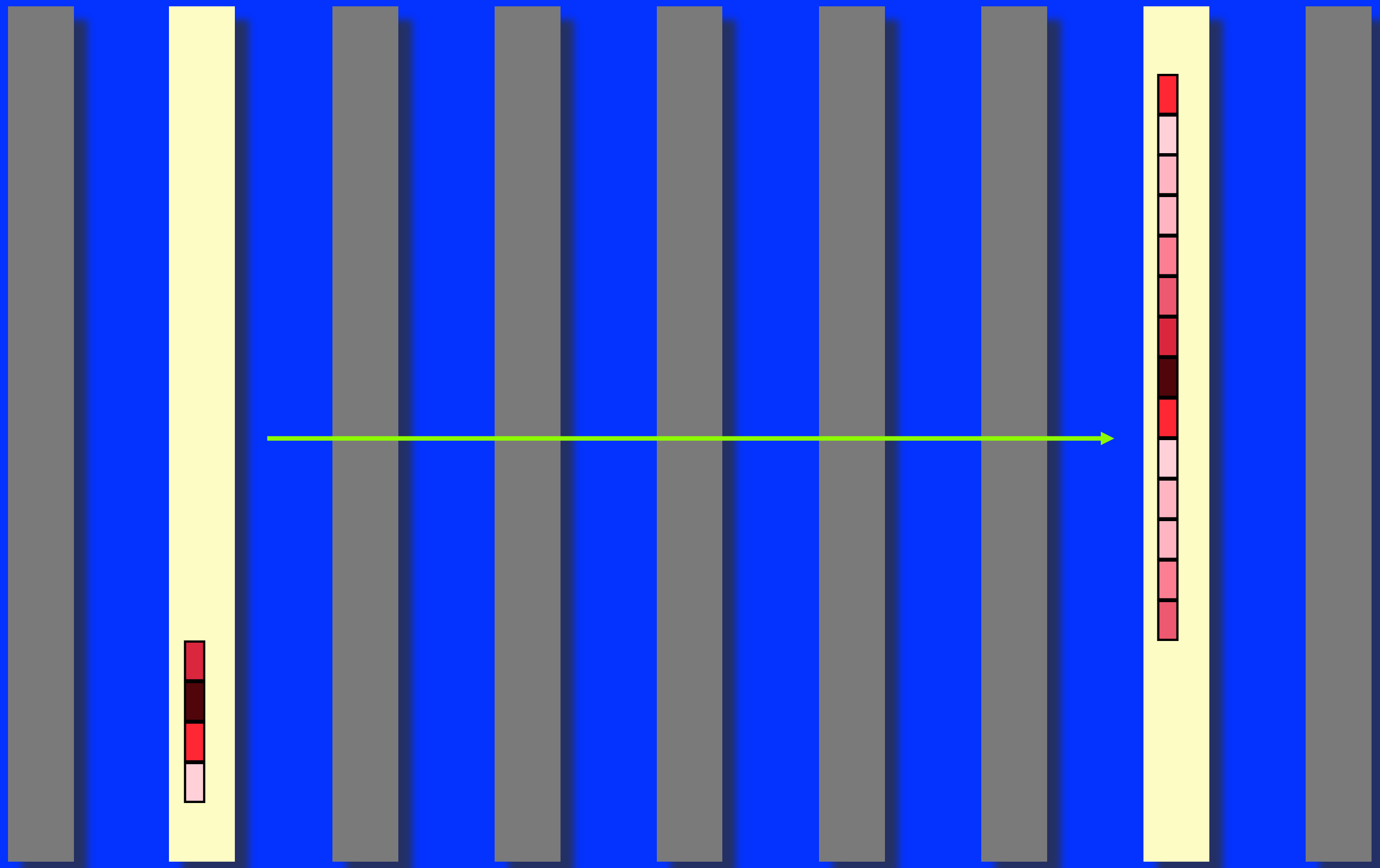


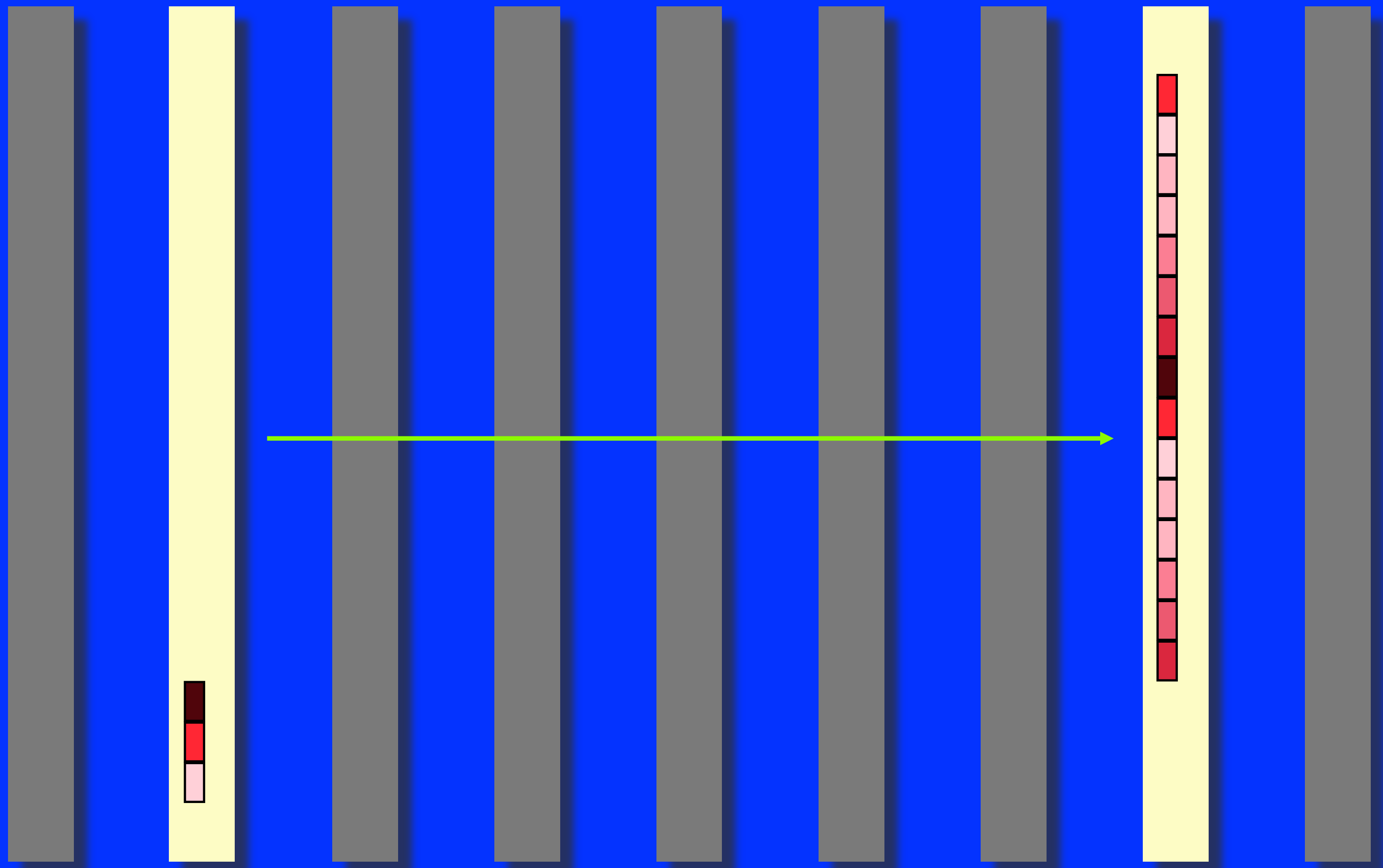


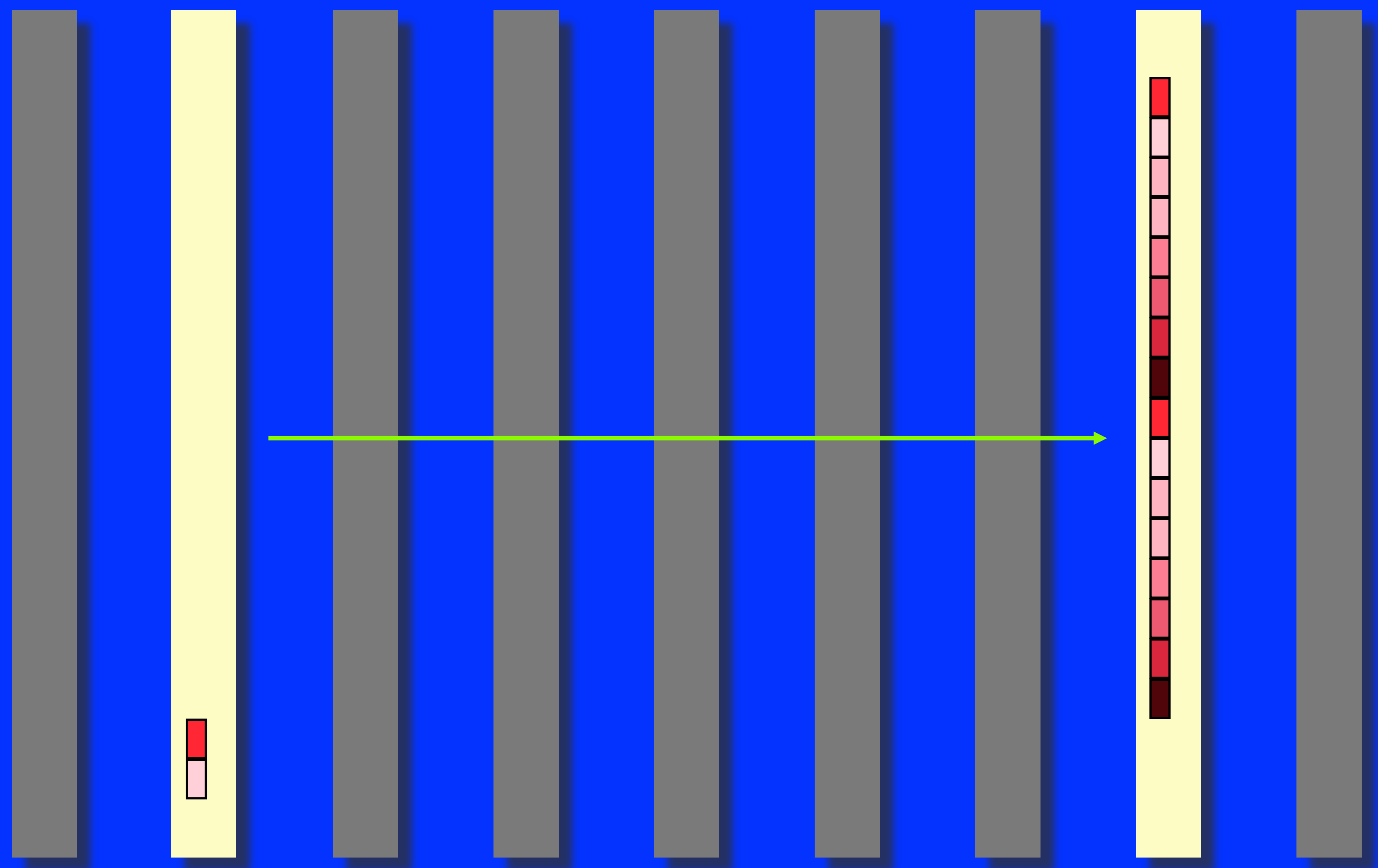


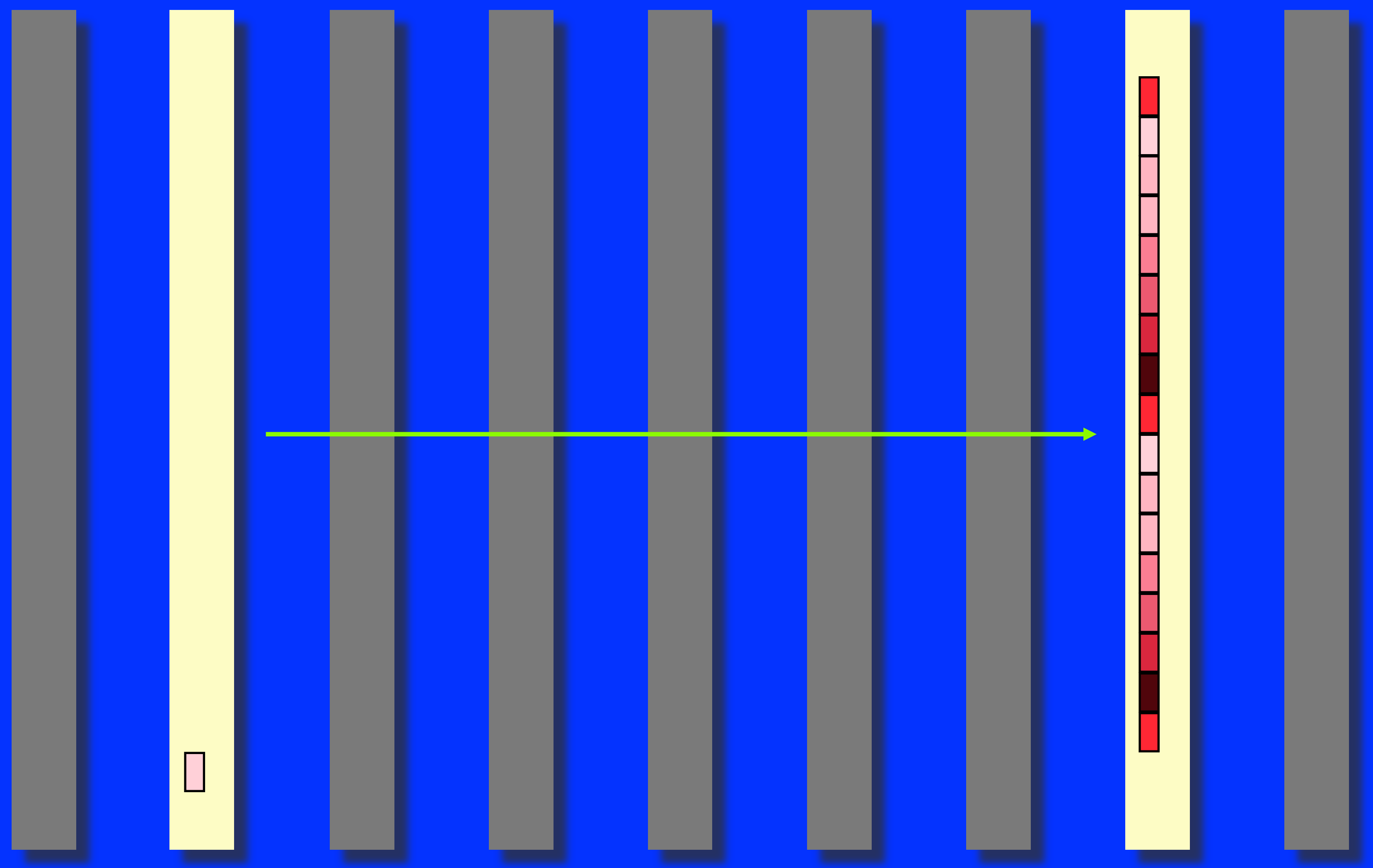


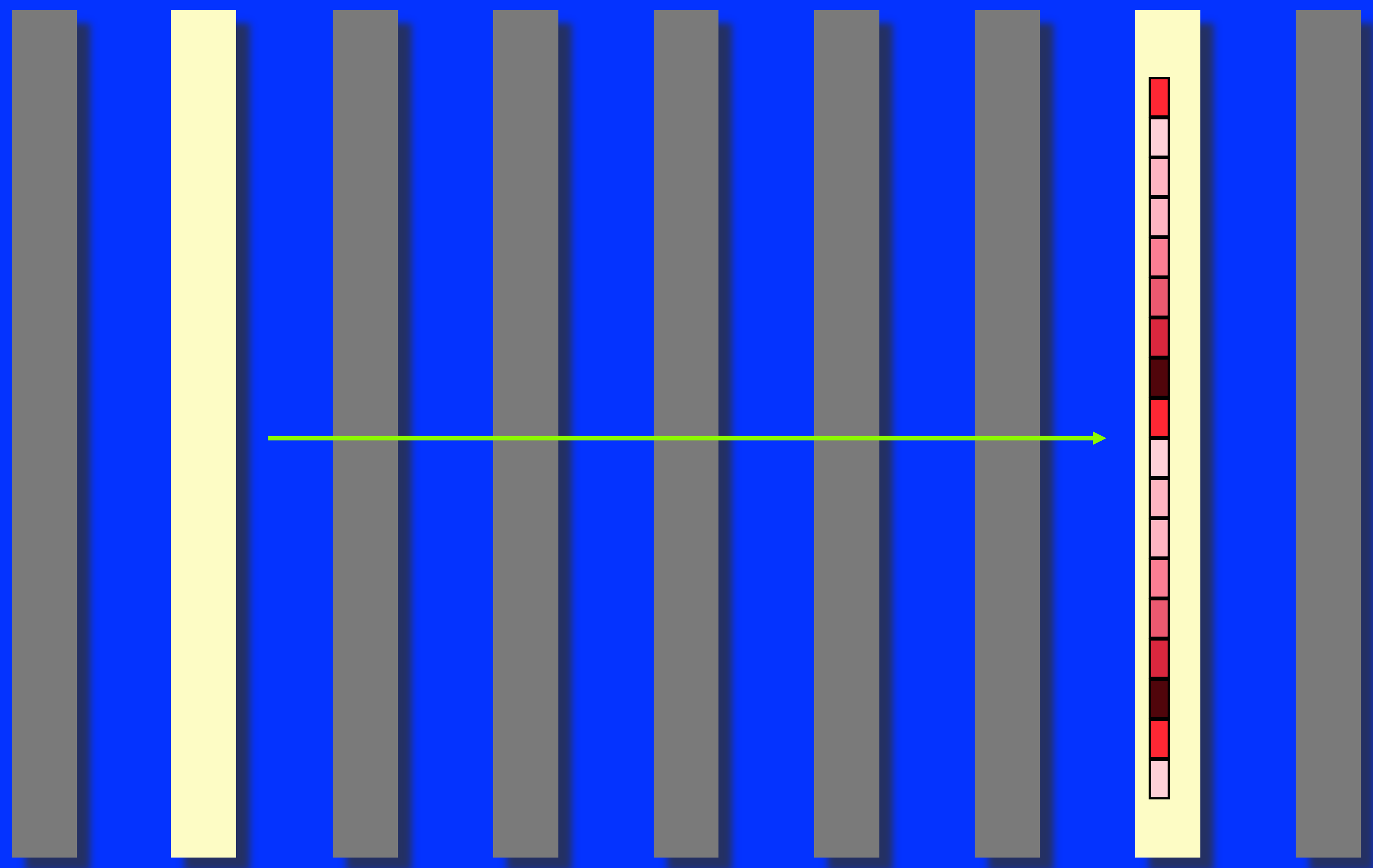


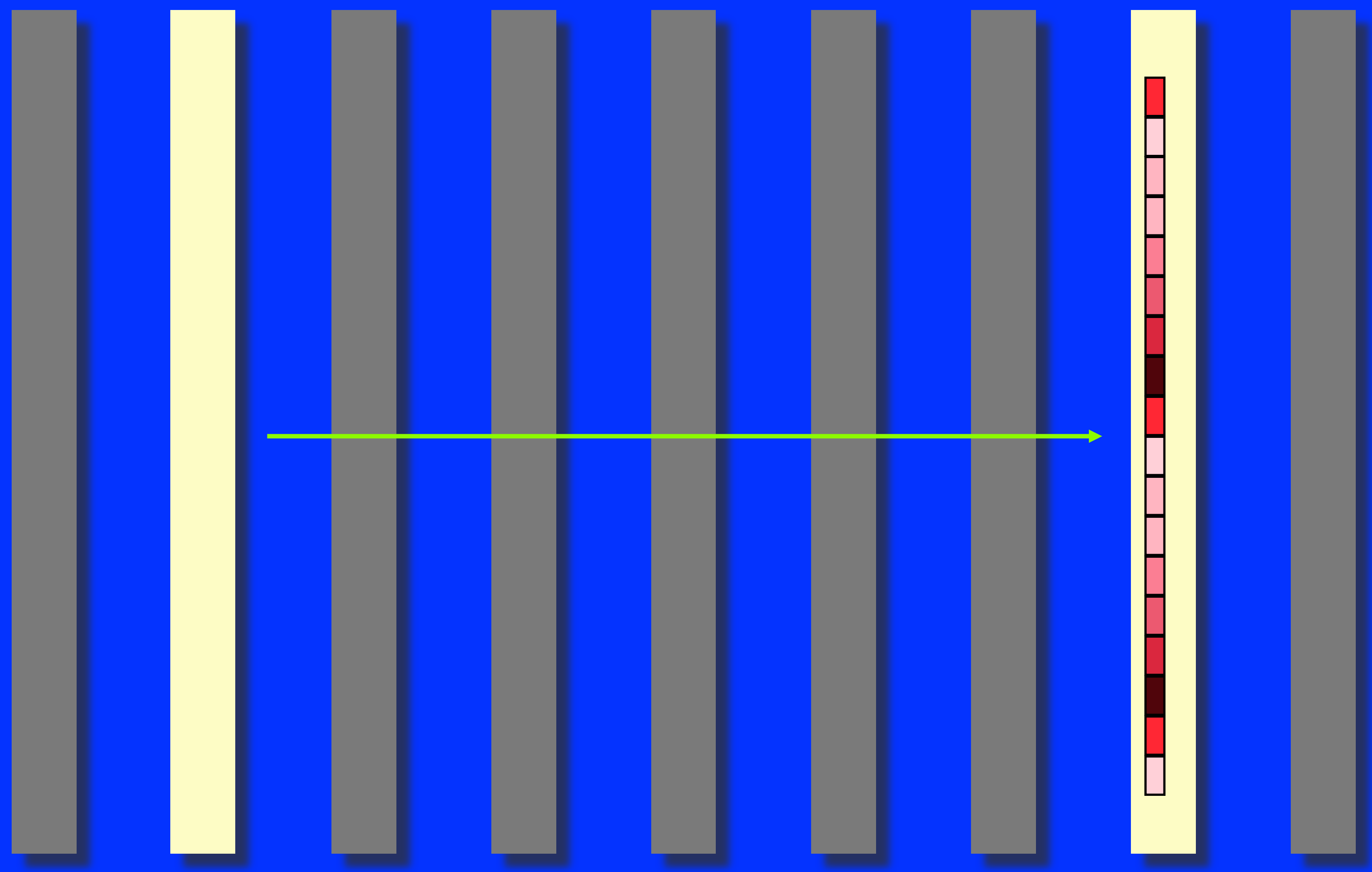












Model of Parallel Computation

- a node can send directly to any other node
- a node can simultaneously receive and send
- cost of communication
  - sending a message of length n between any two nodes
  - if a message encounters a link that simultaneously accomodates M messages, the cost becomes

Model of Parallel Computation

- a node can send directly to any other node
- a node can simultaneously receive and send
- cost of communication
  - sending a message of length n between any two nodes

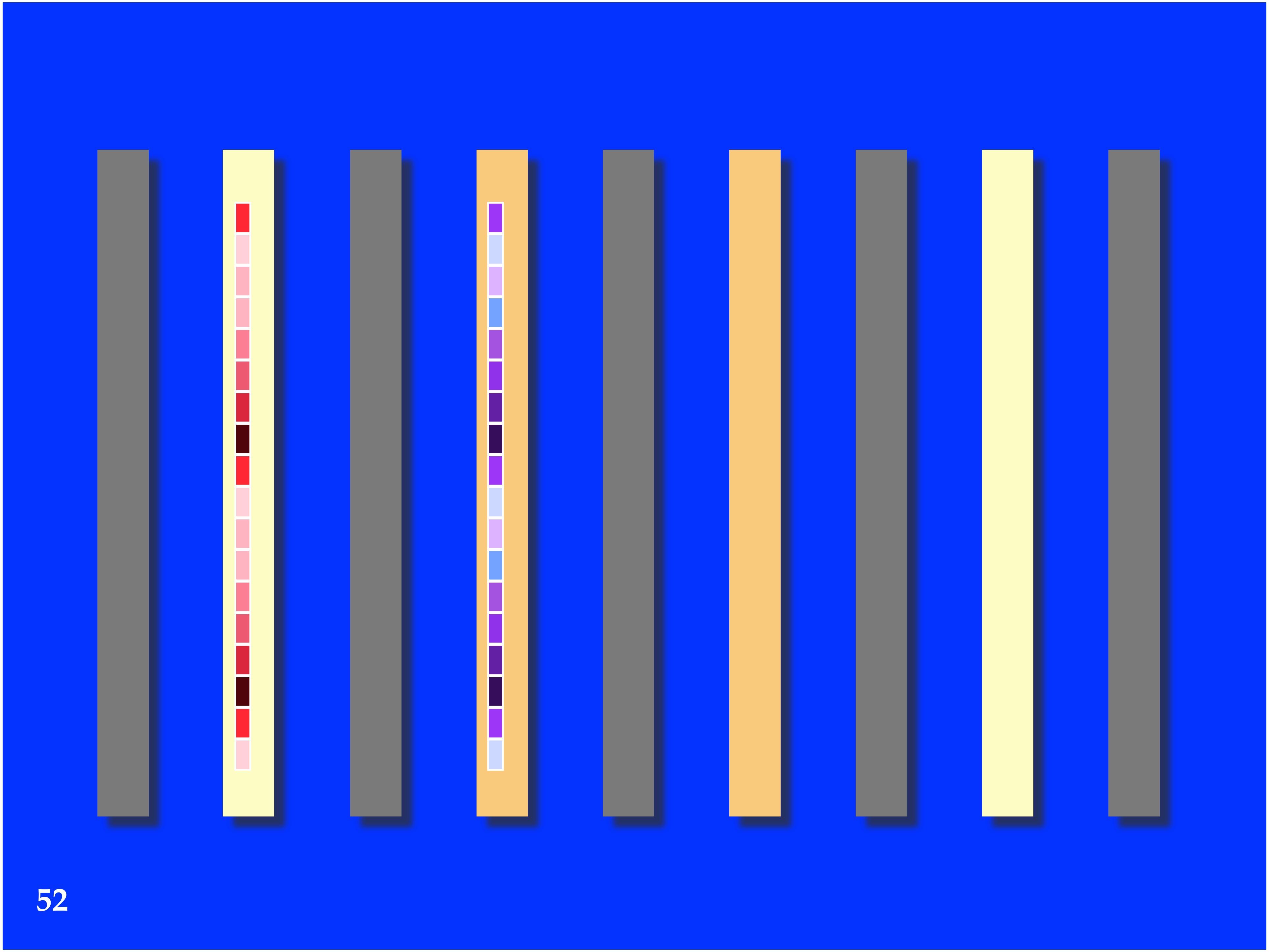
$$\alpha + n\beta$$

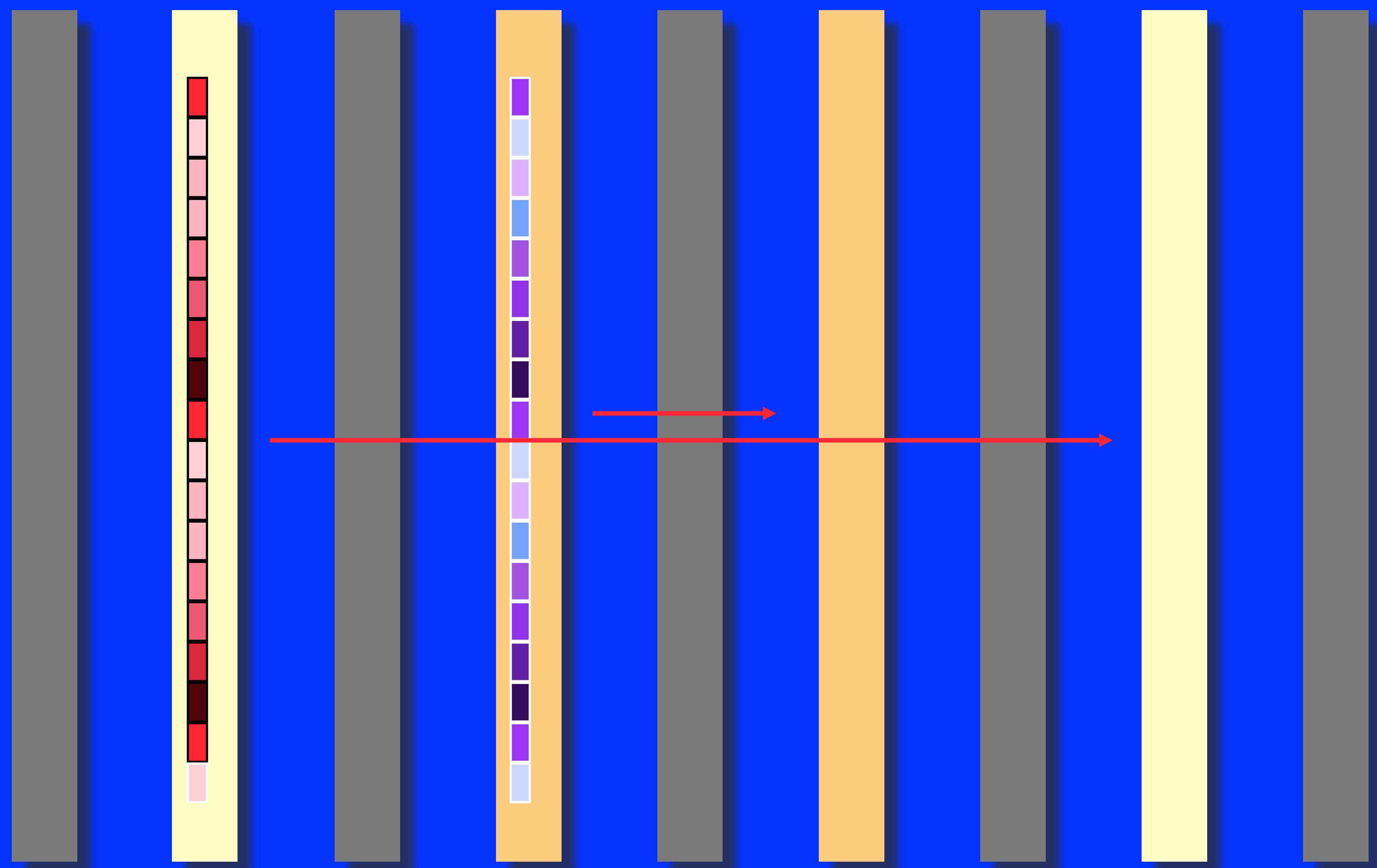
  - if a message encounters a link that simultaneously accomodates M messages, the cost becomes

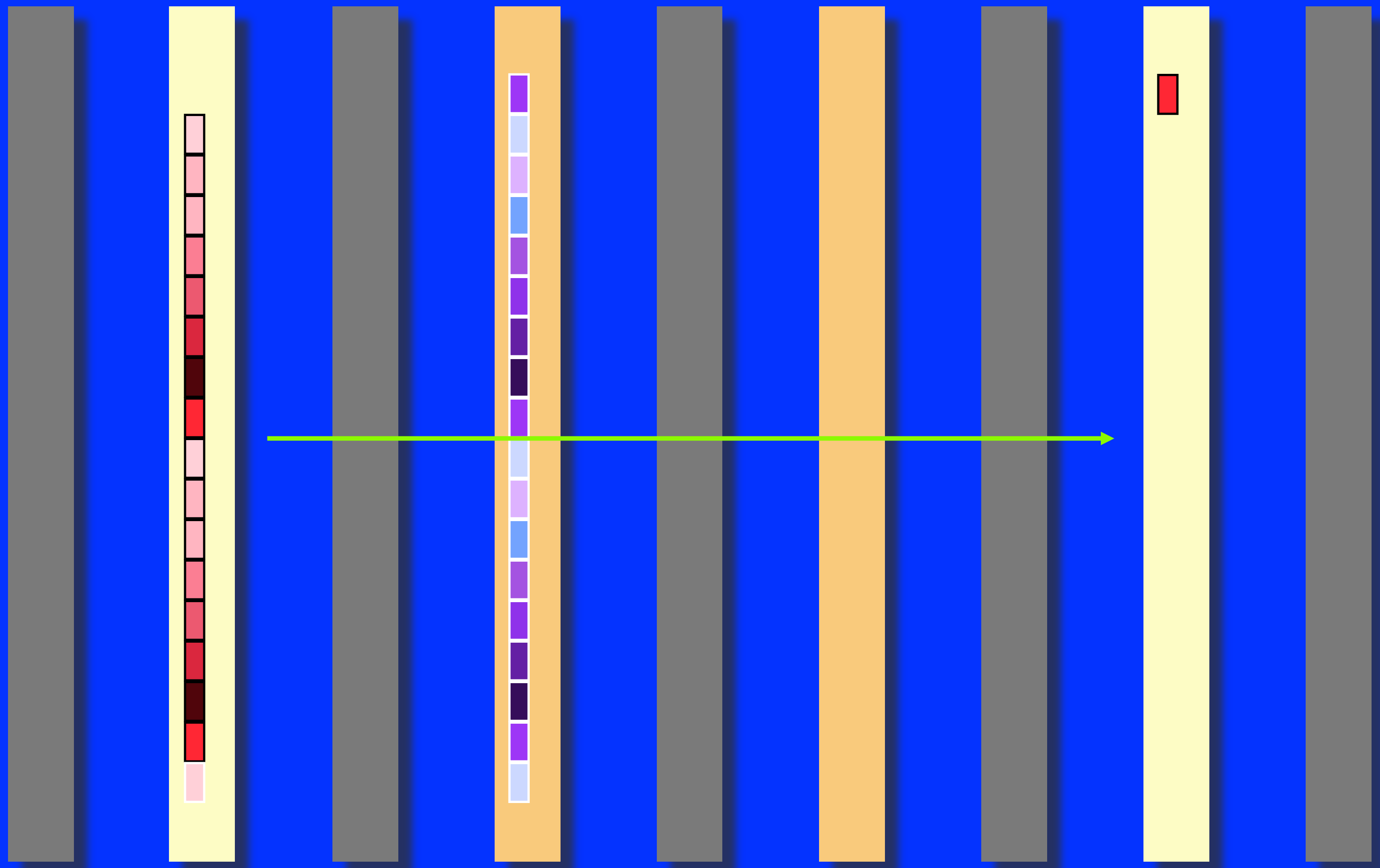
$$\alpha + Mn\beta$$

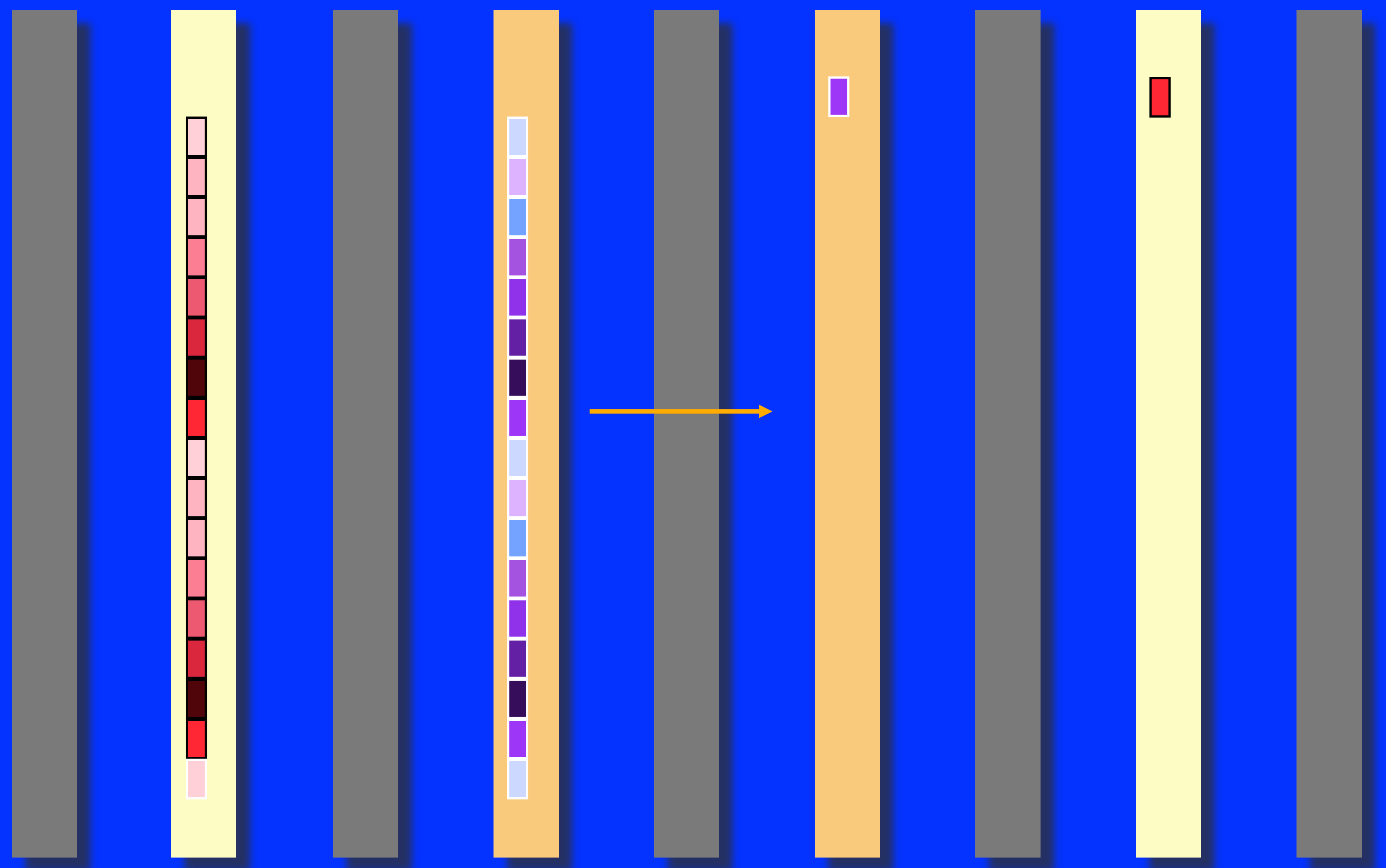
Interfering messages

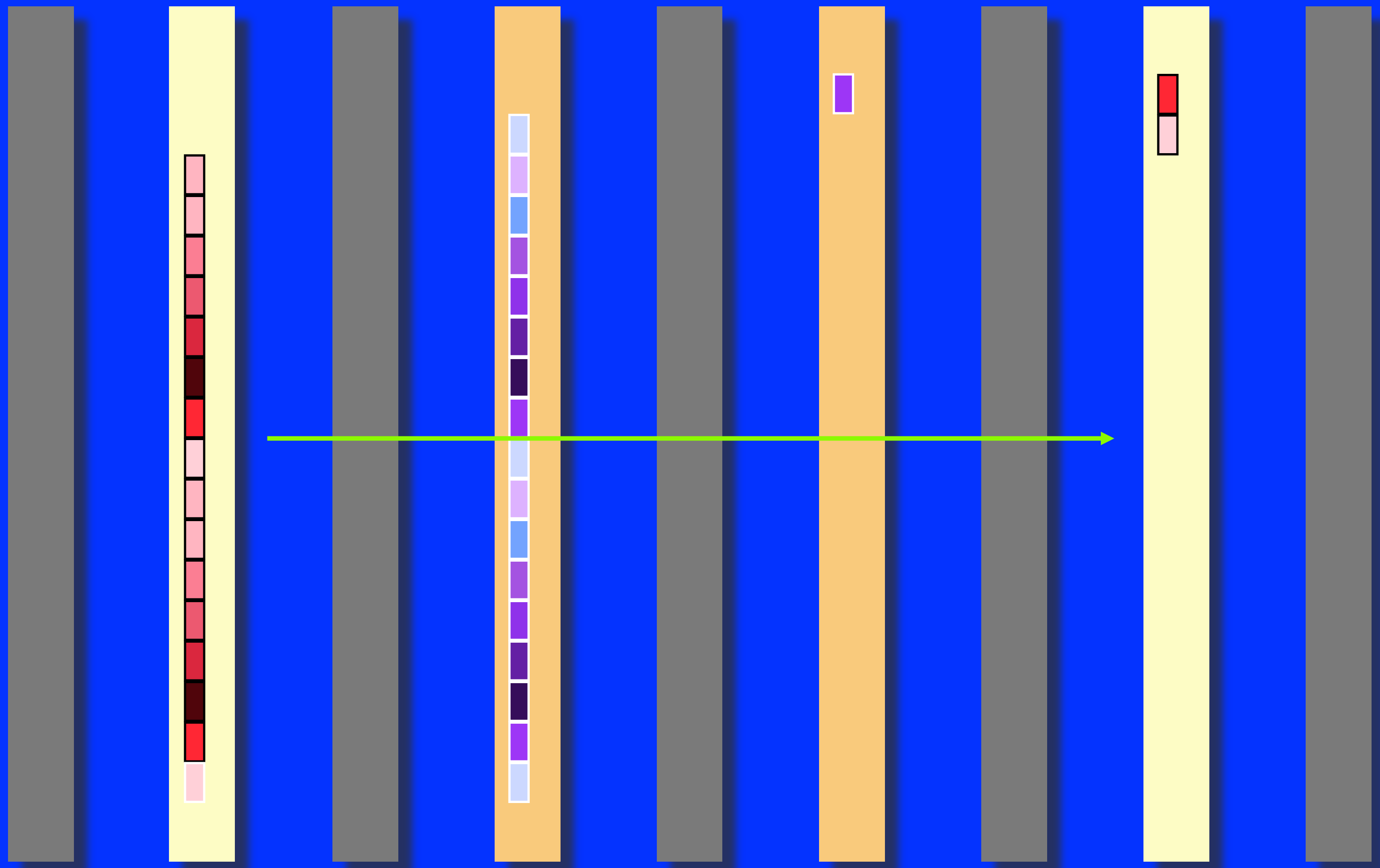
- Example: two messages of length n which share at least one link

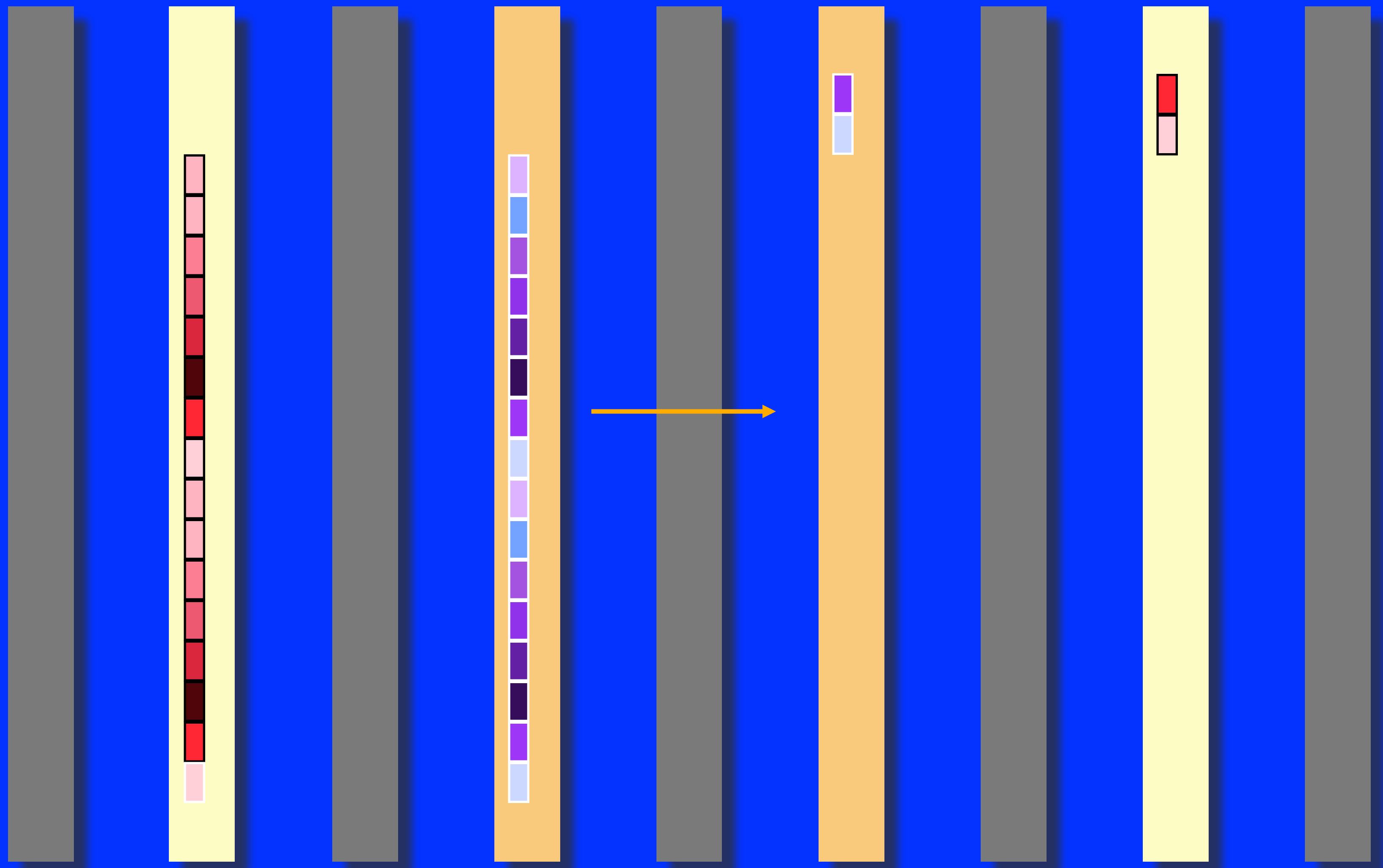


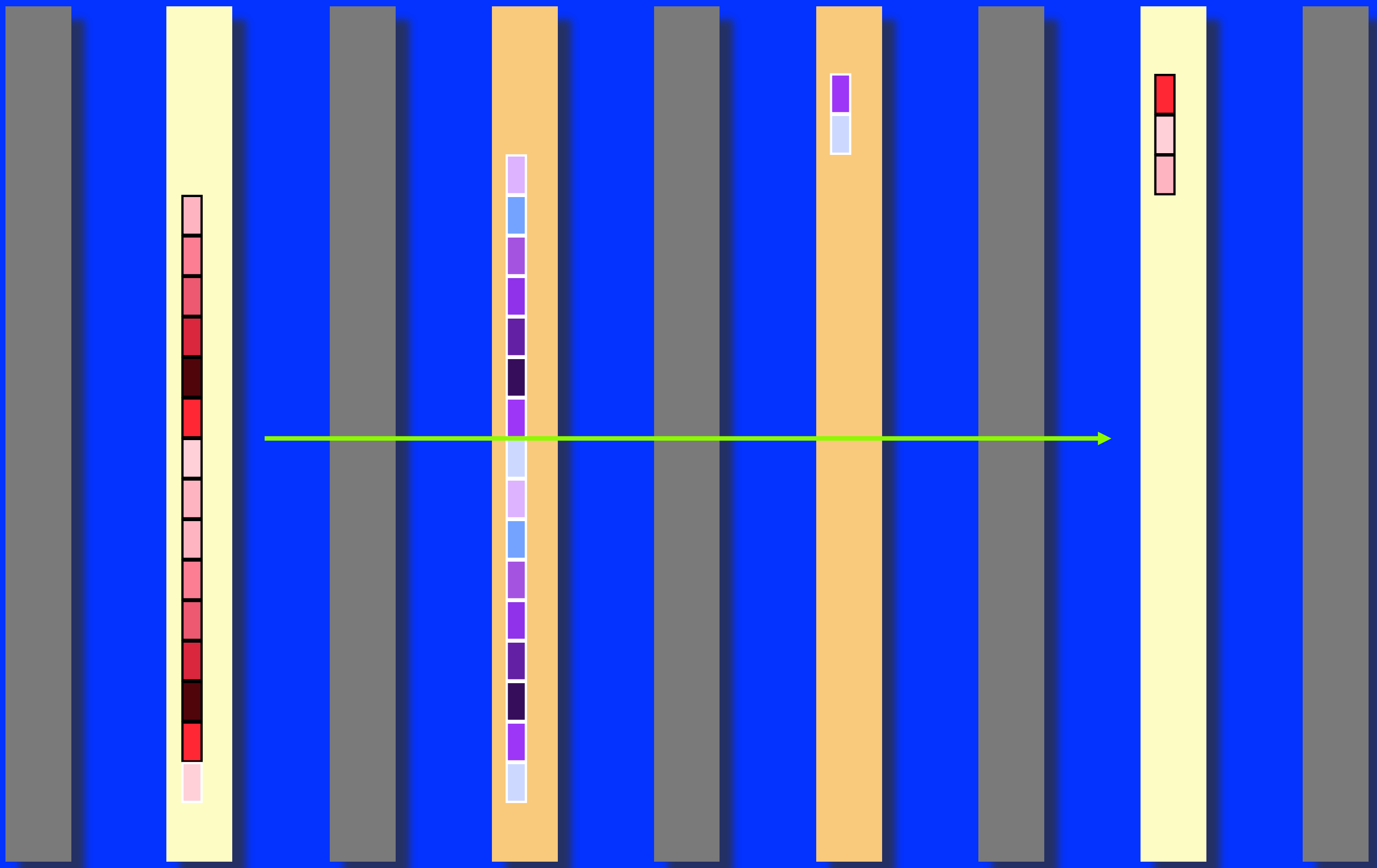


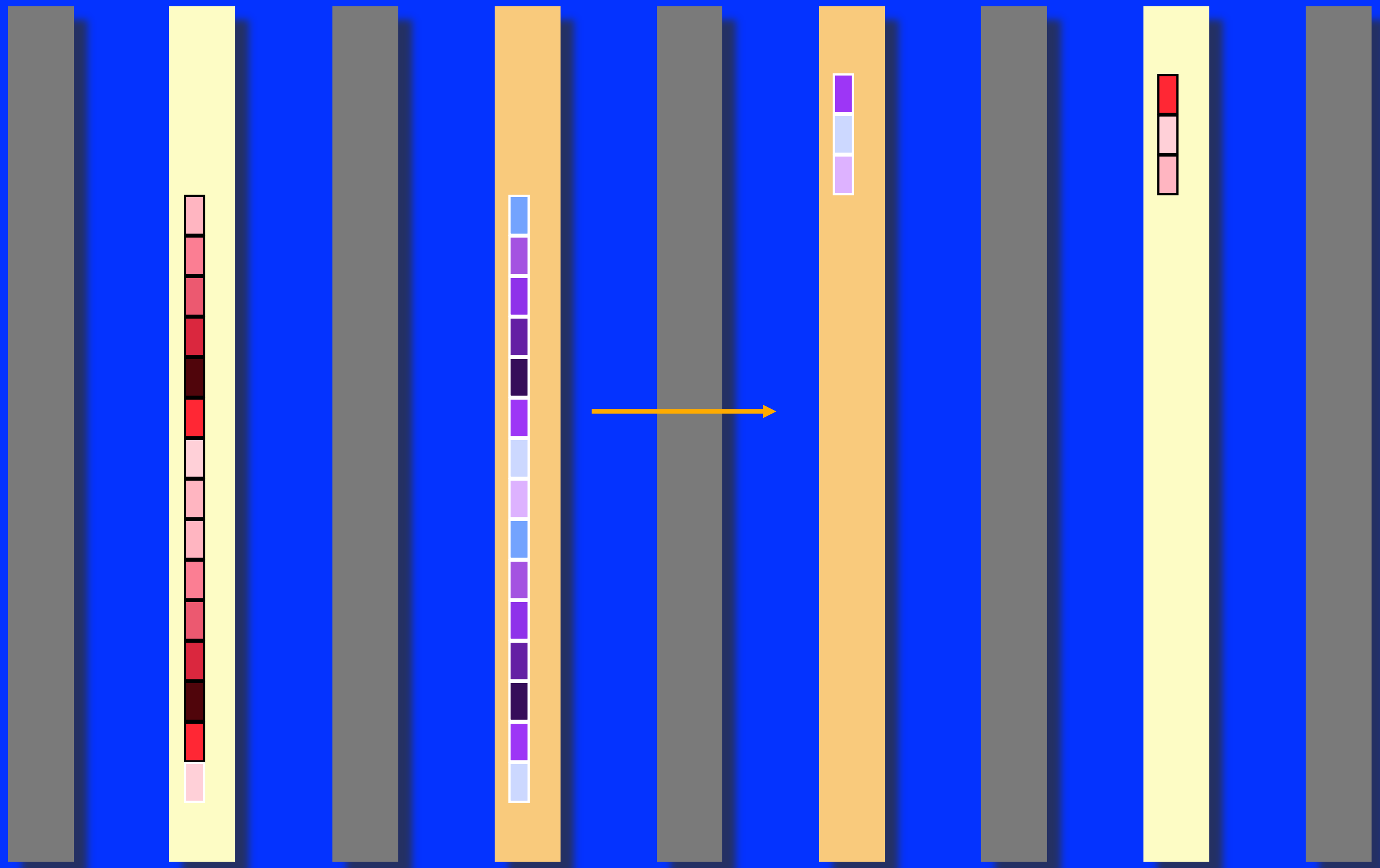


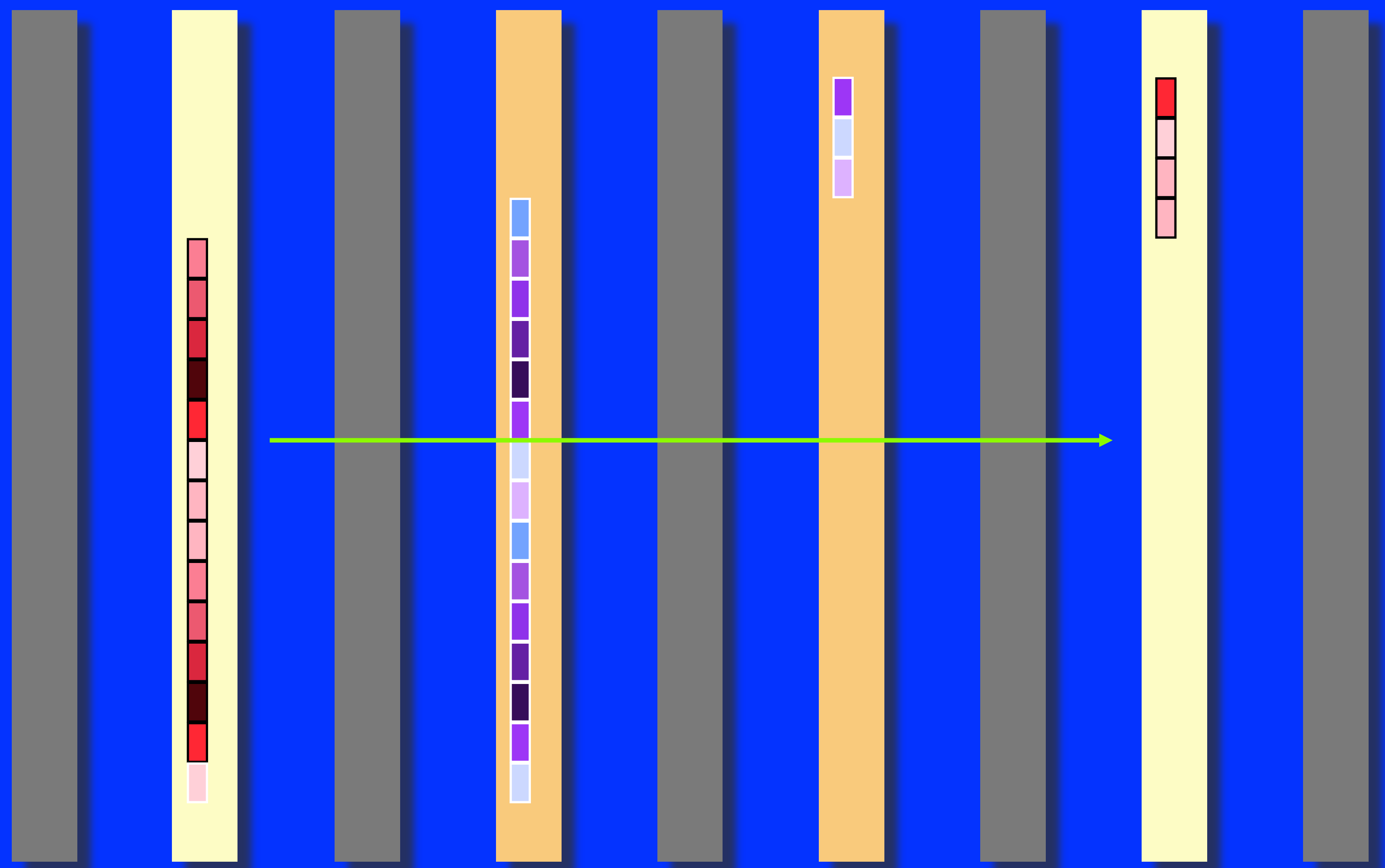


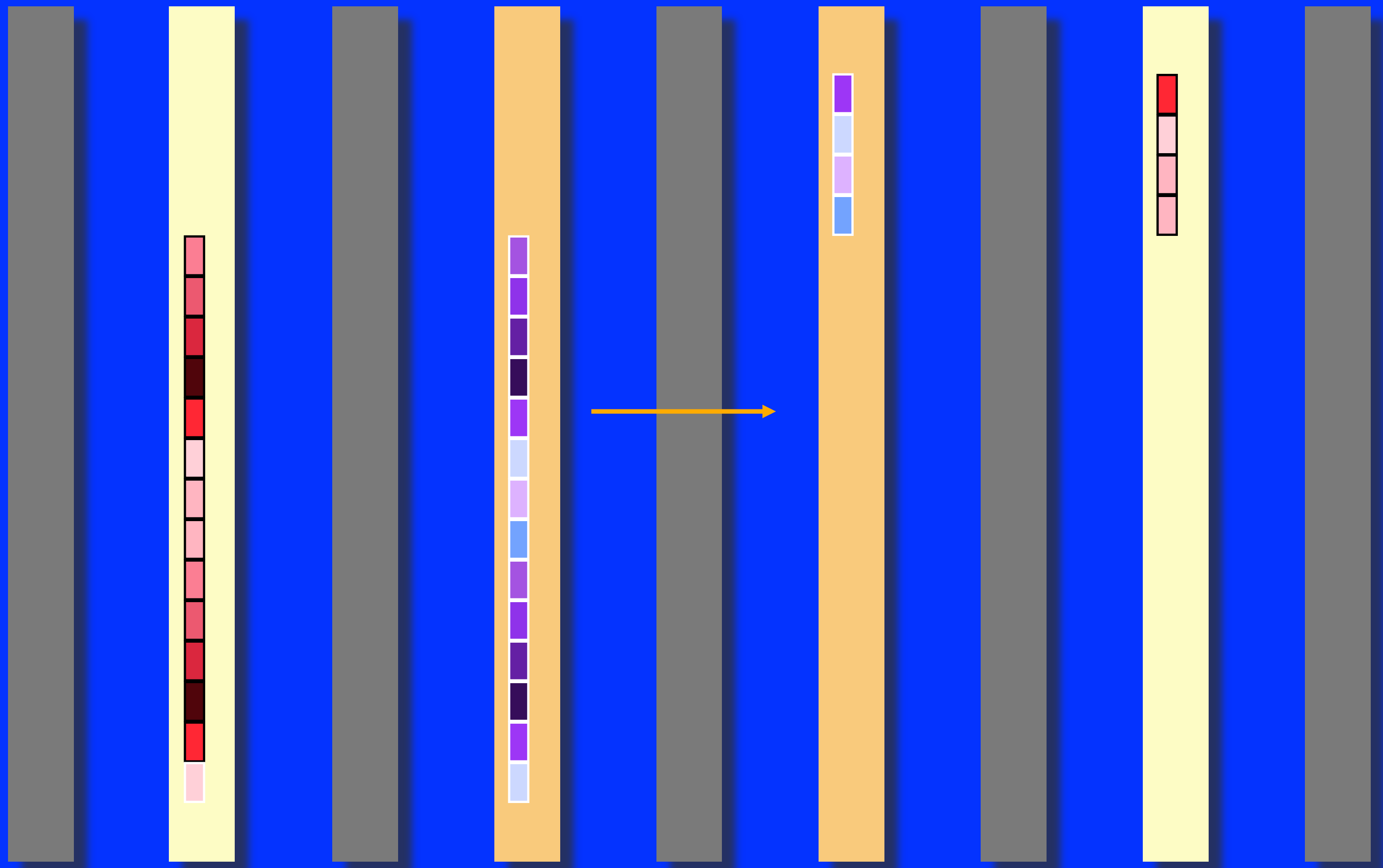


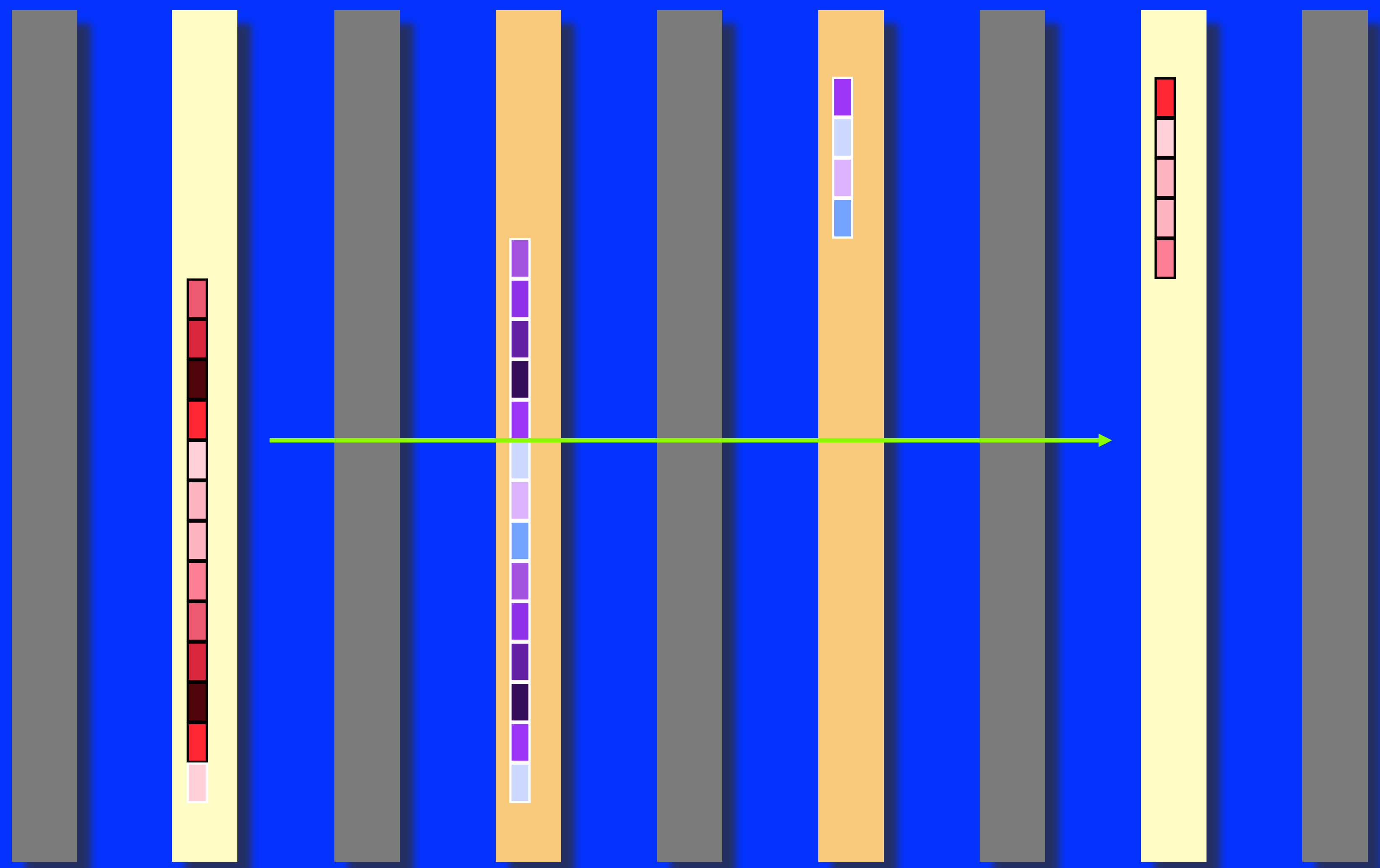


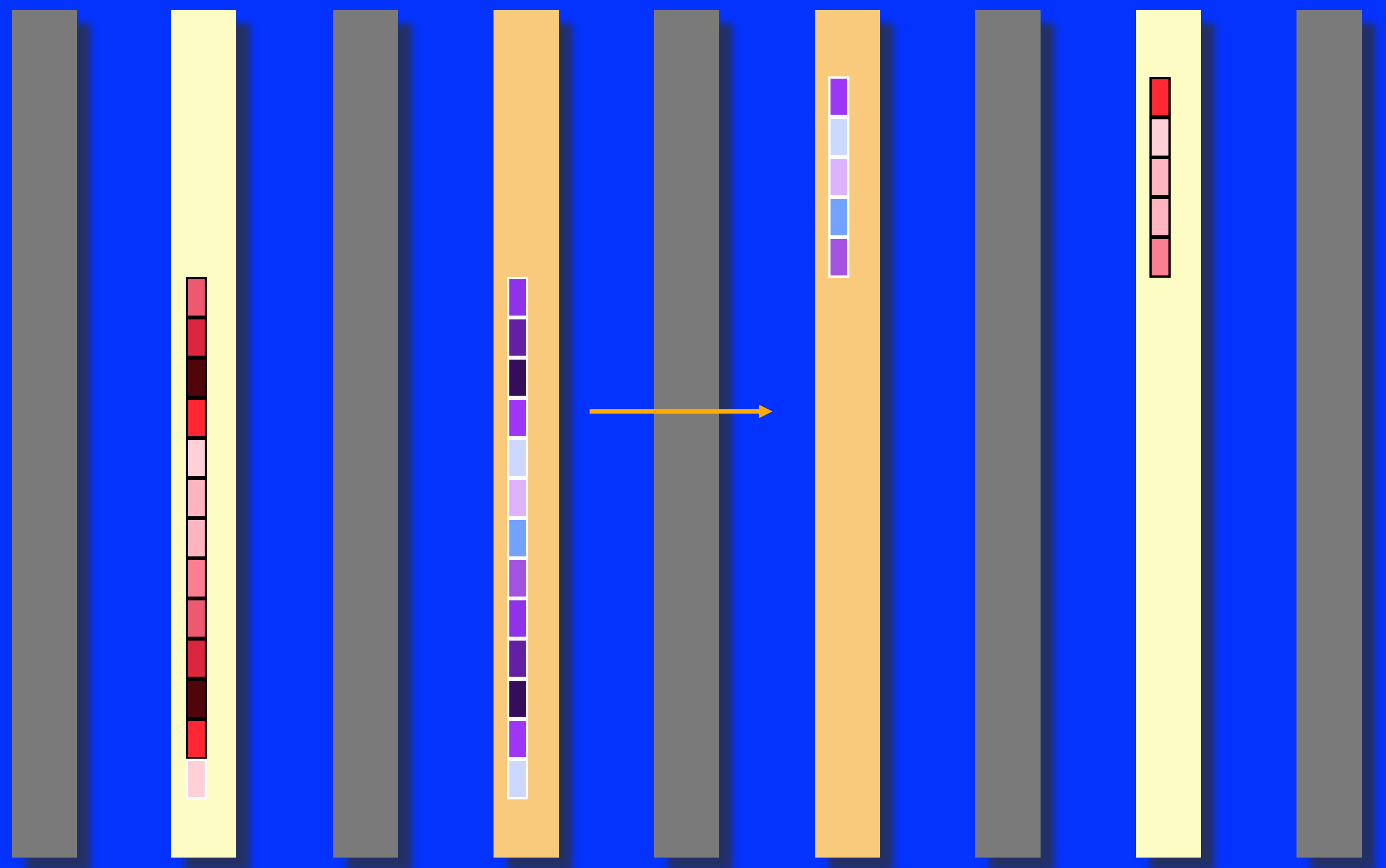


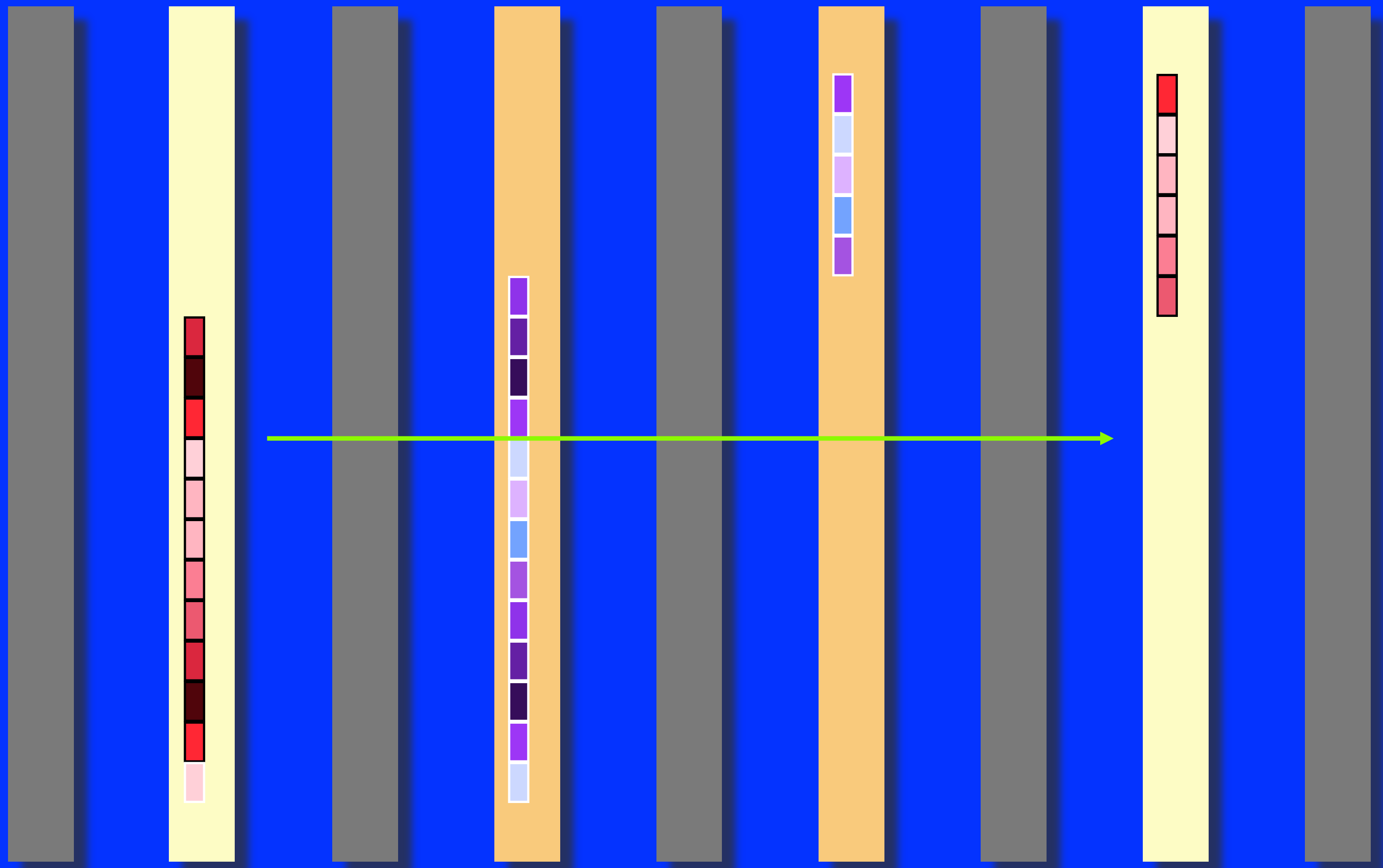


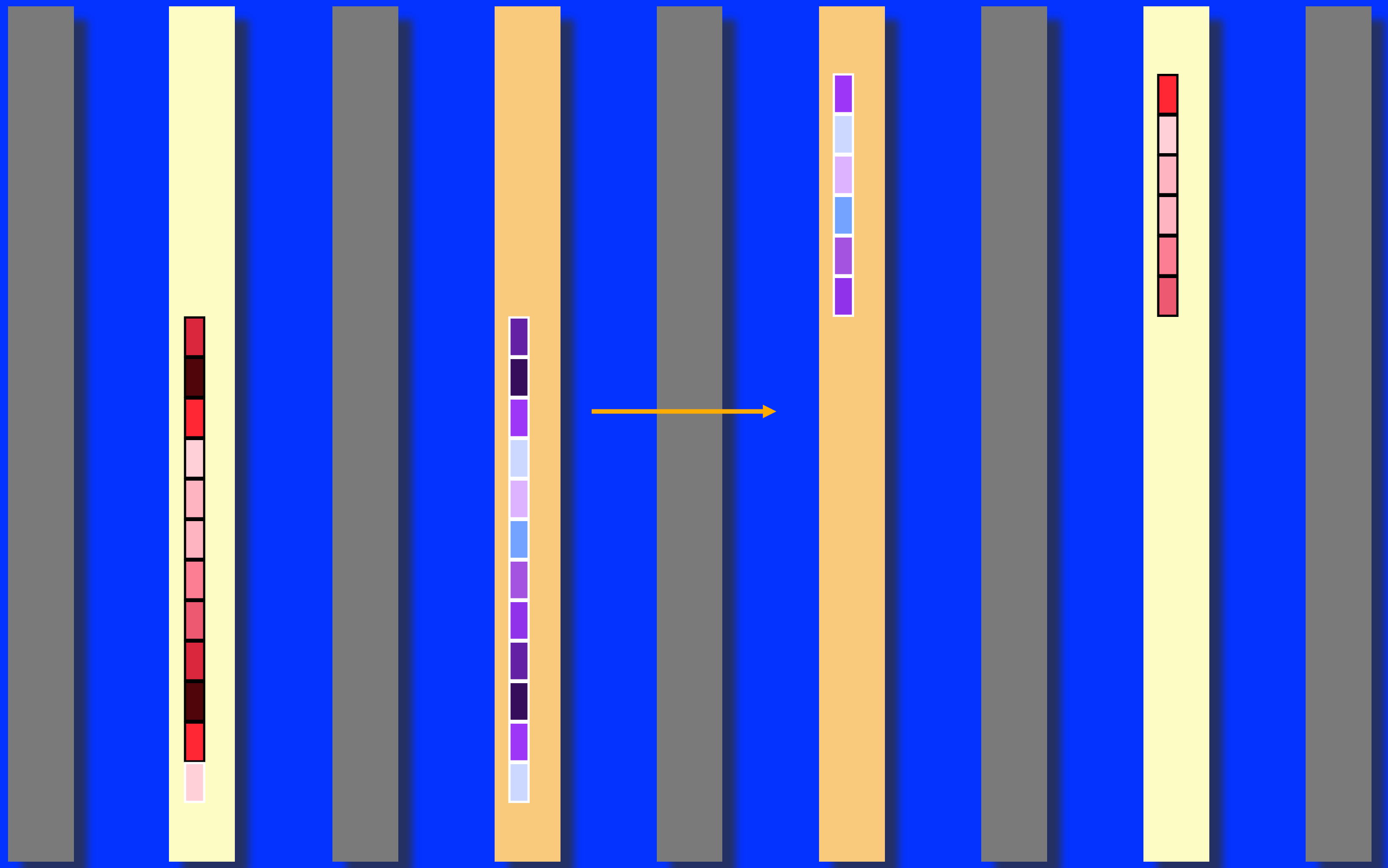


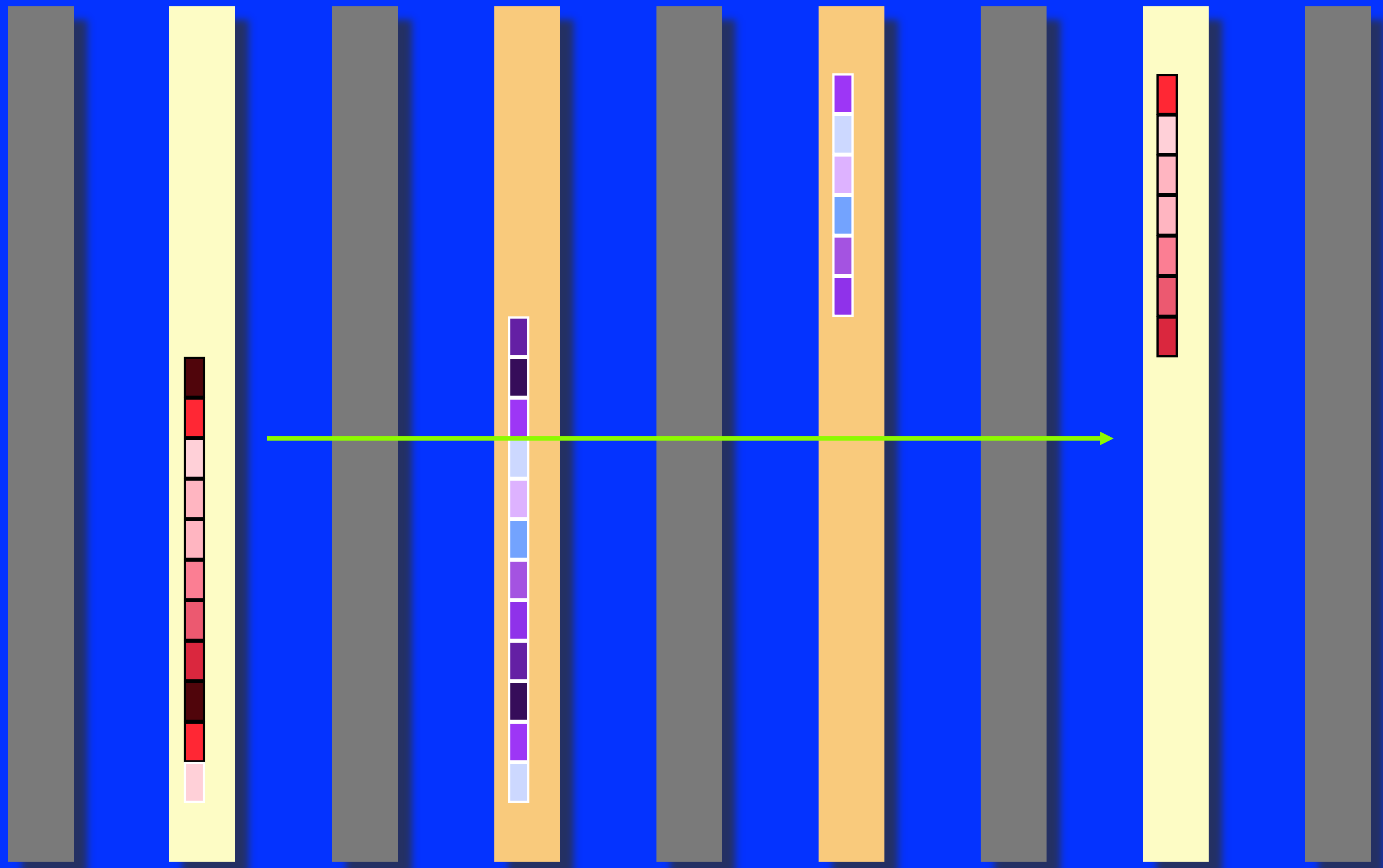


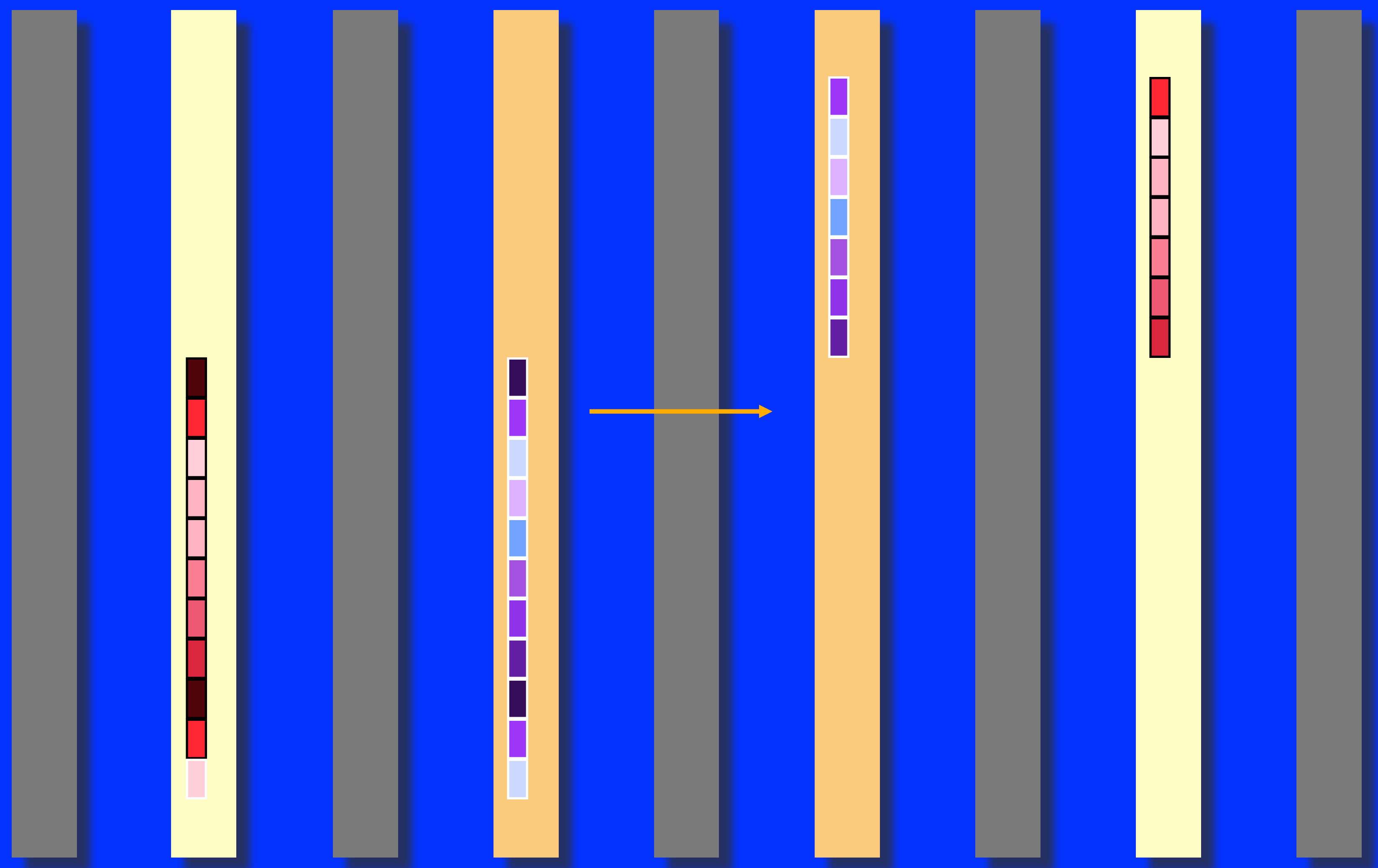


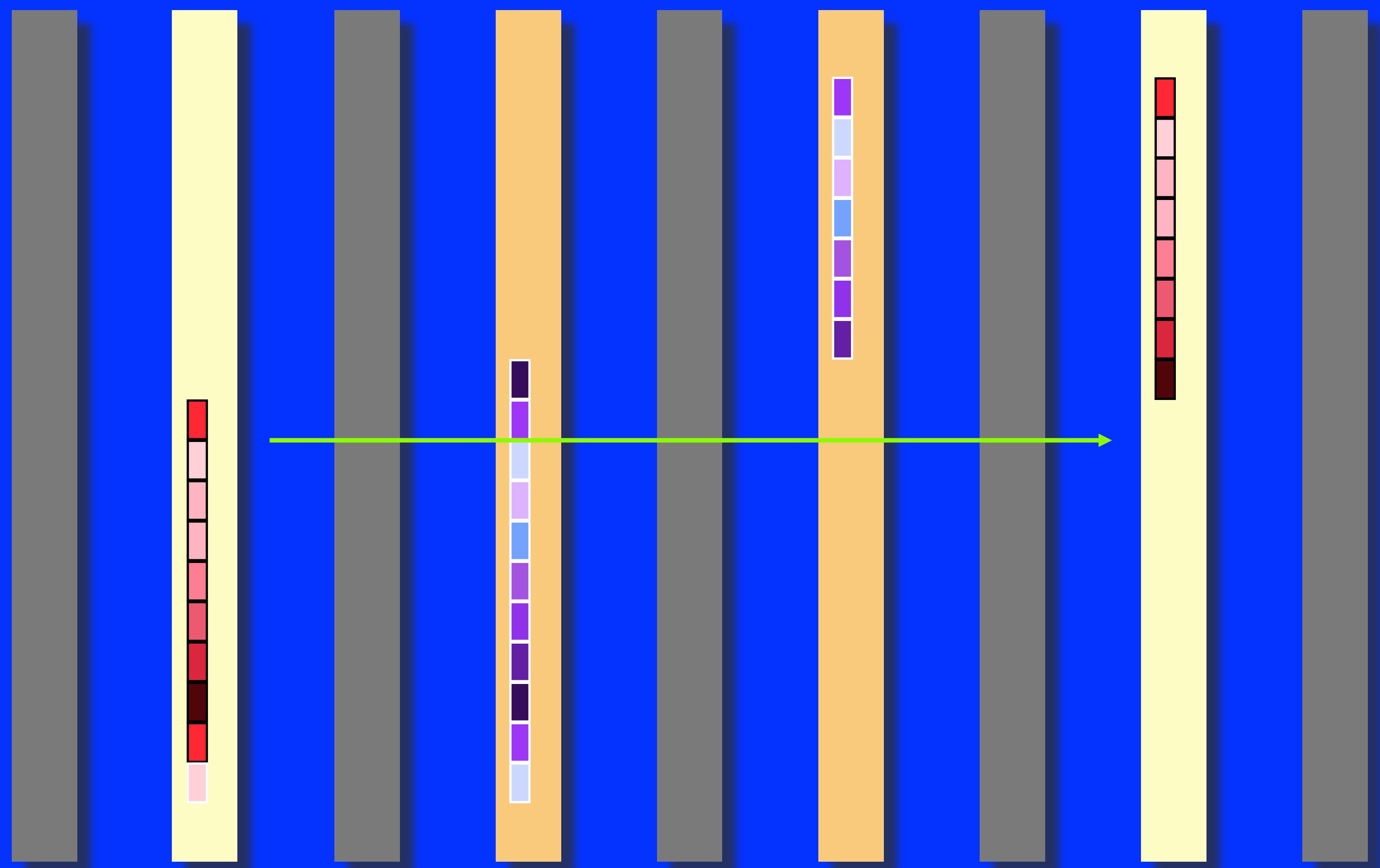


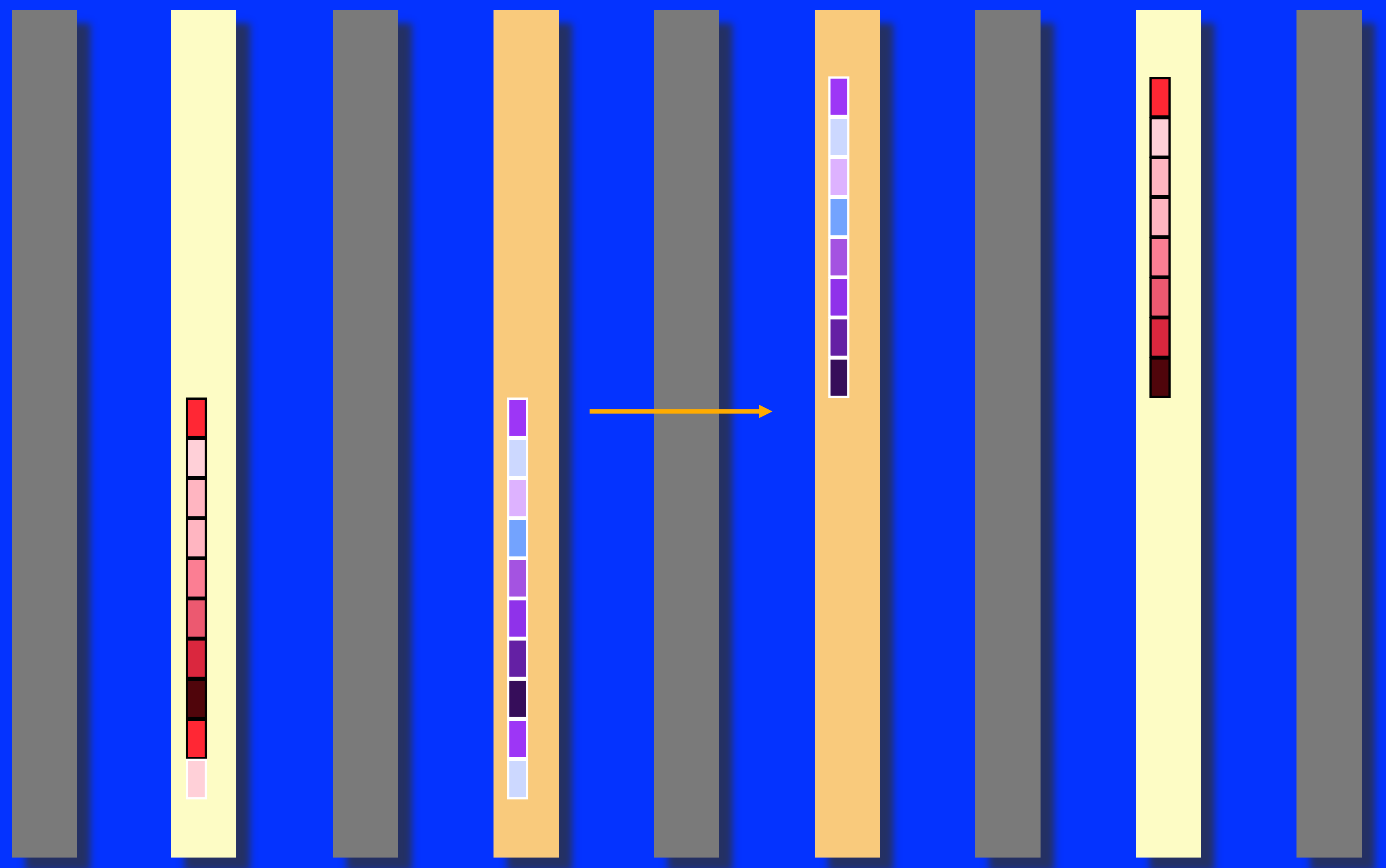


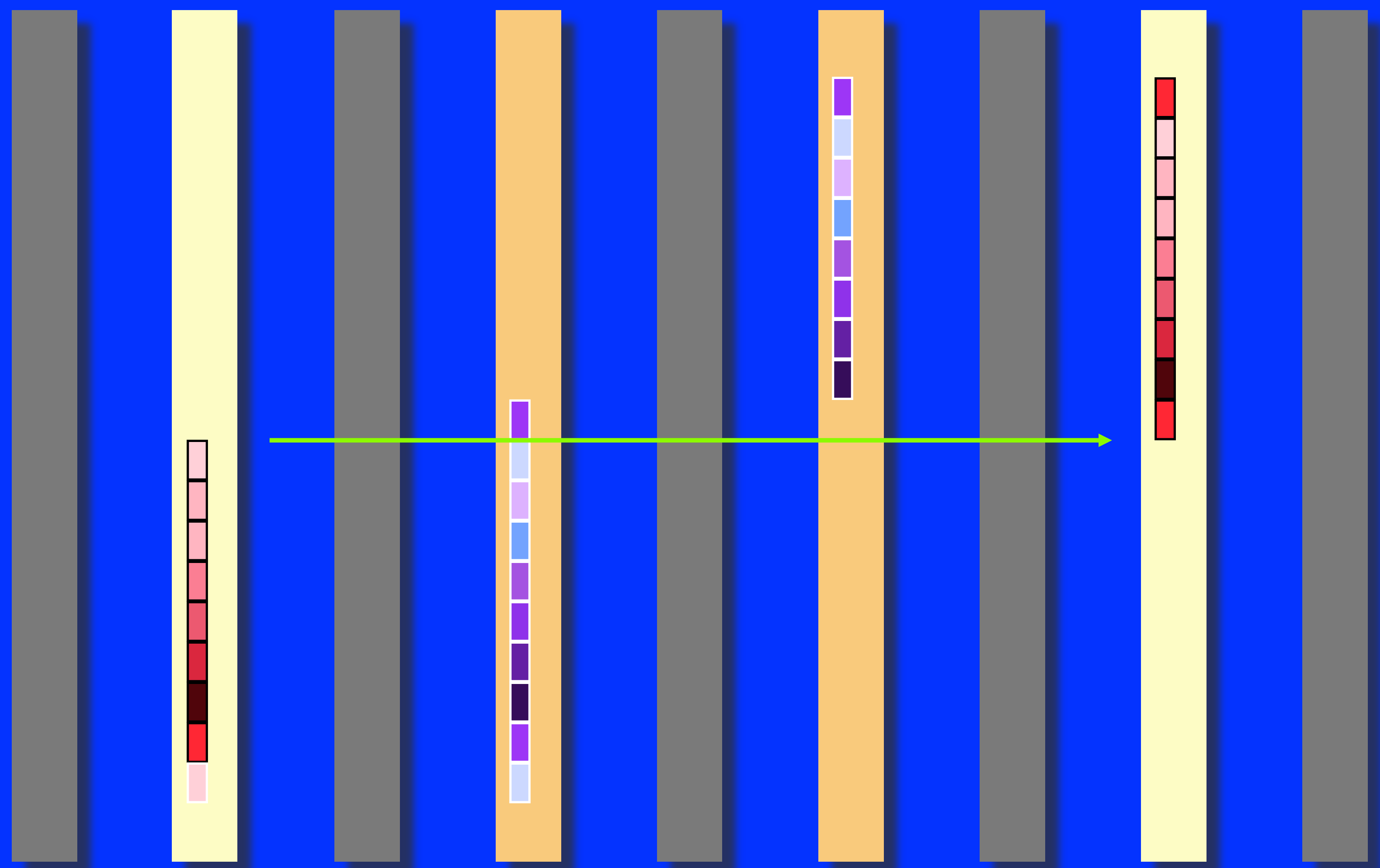


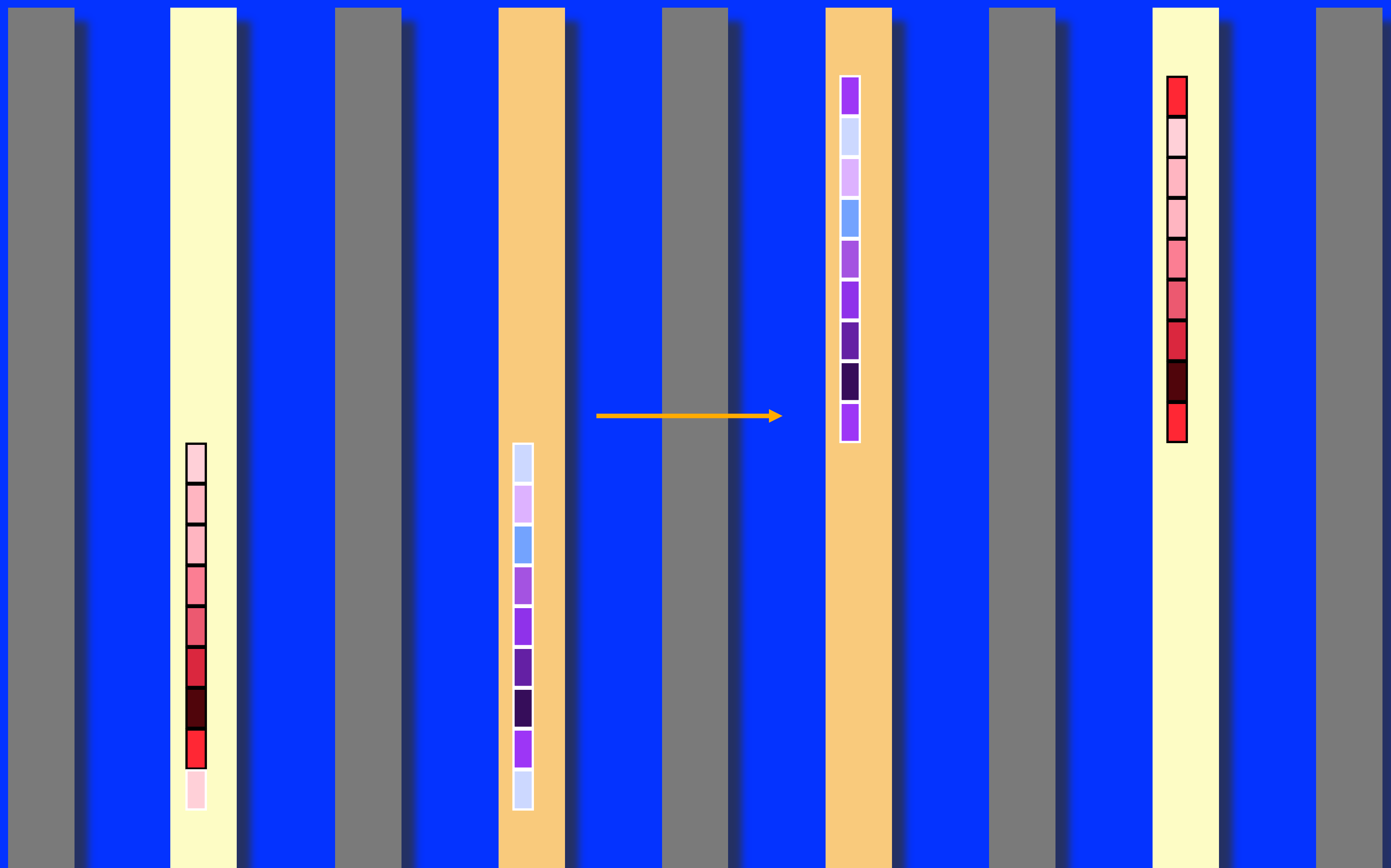


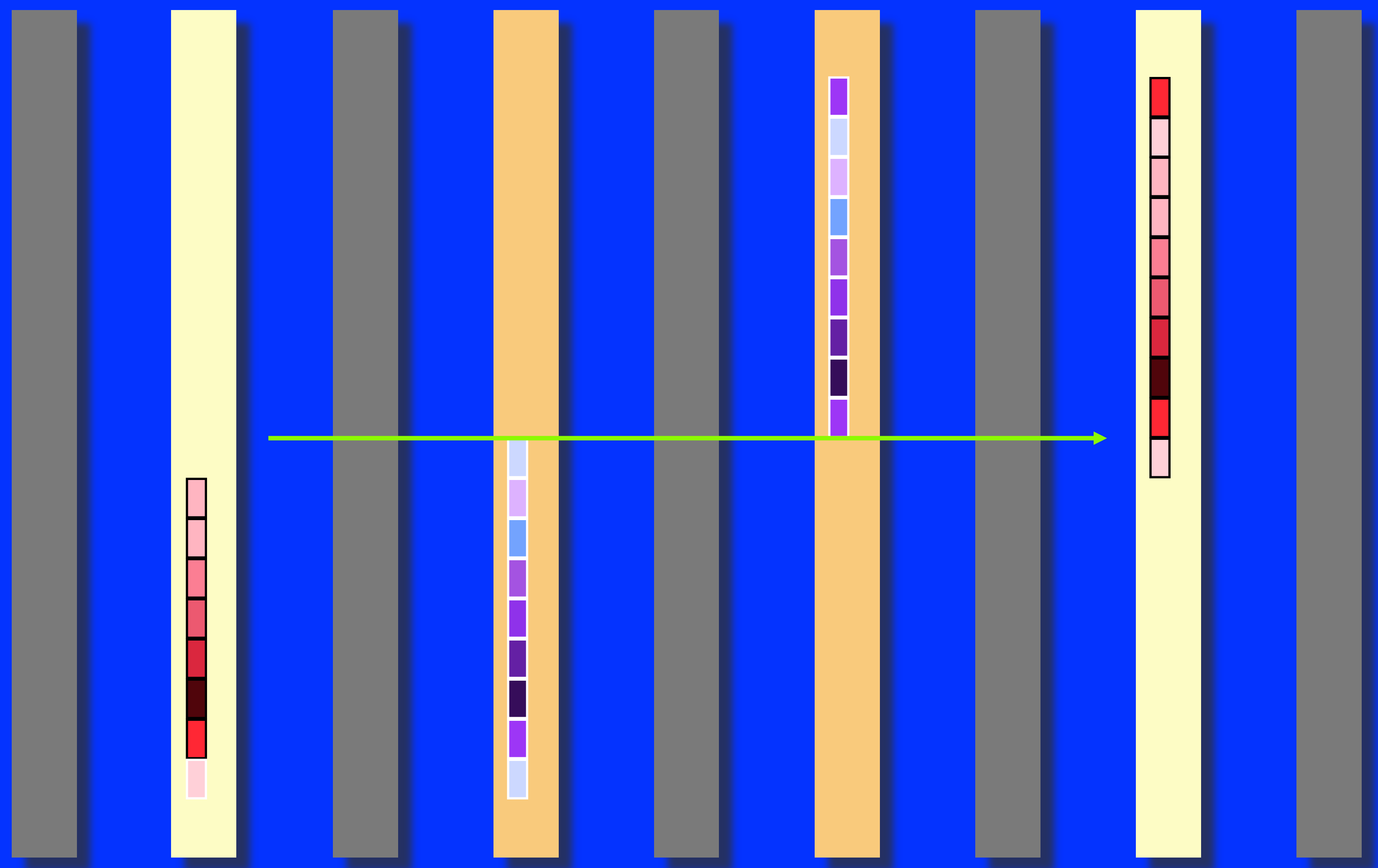


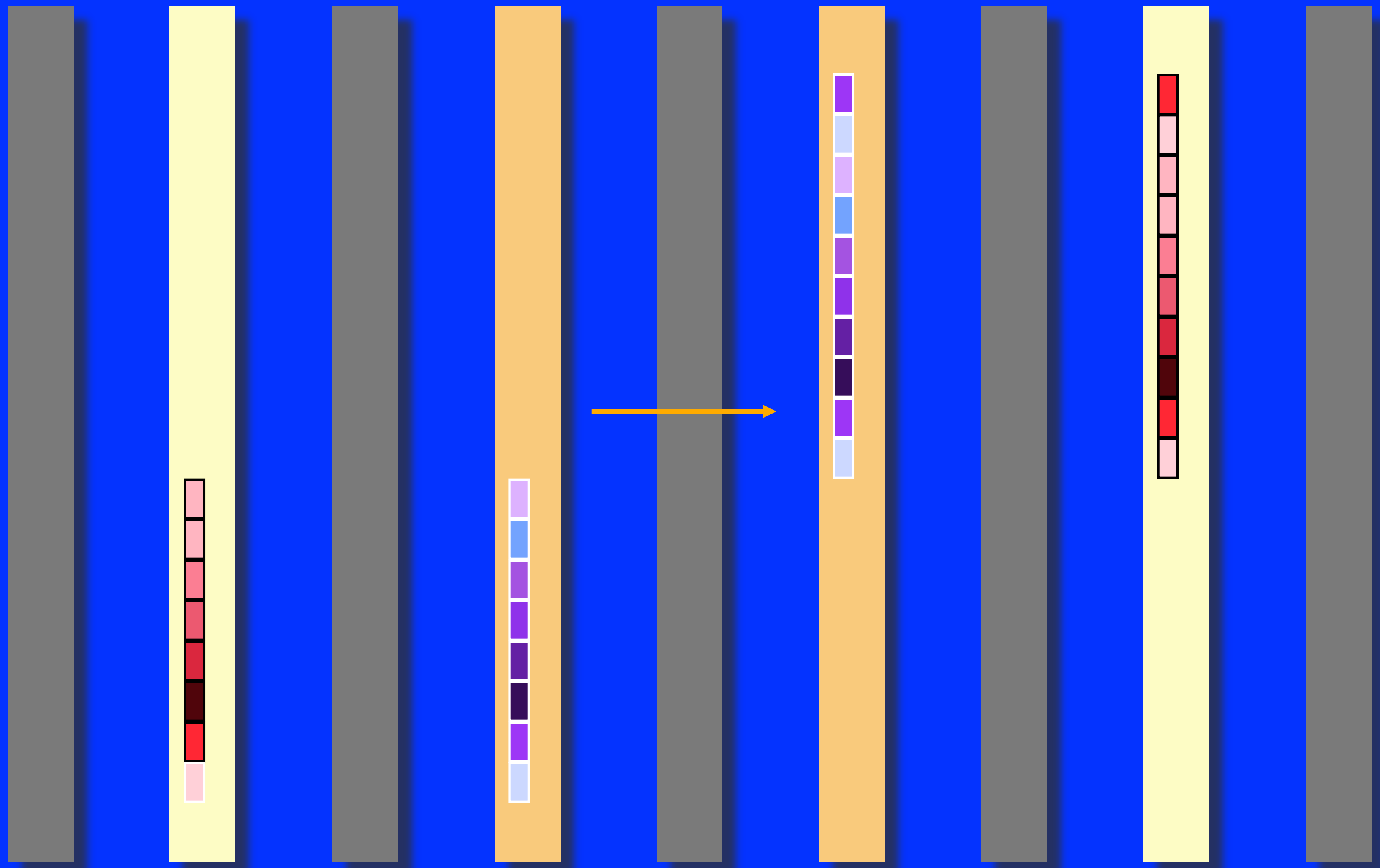


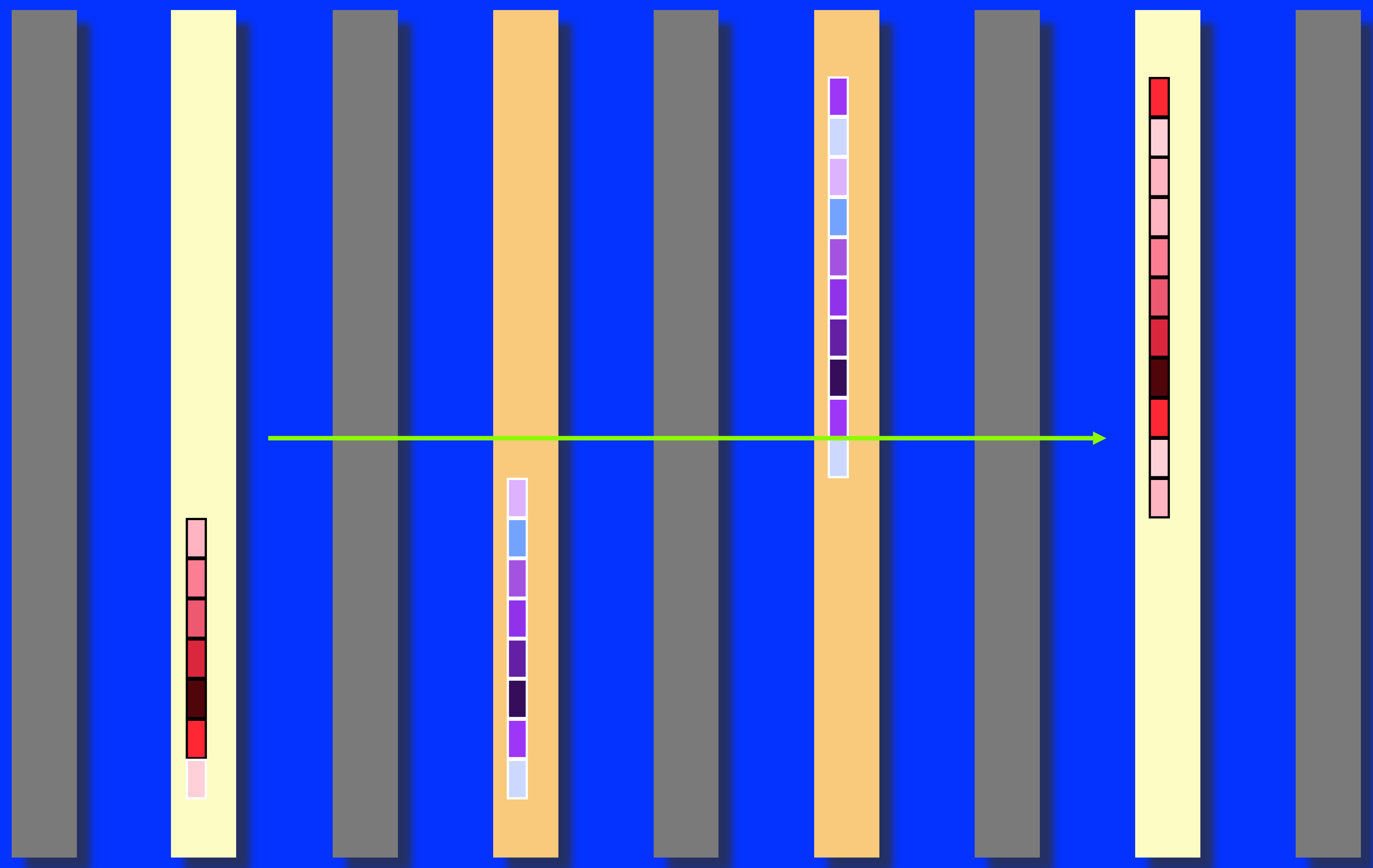


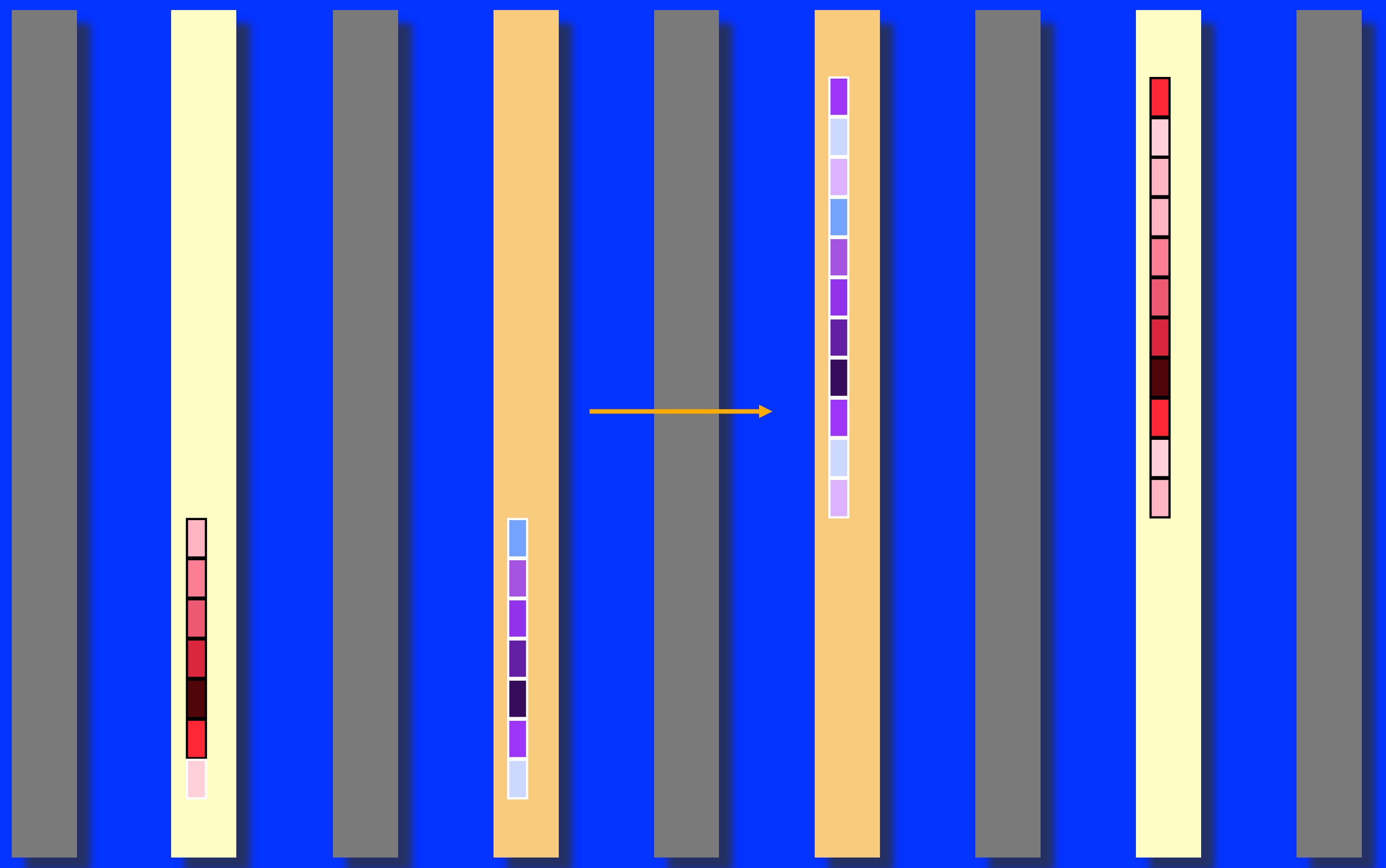


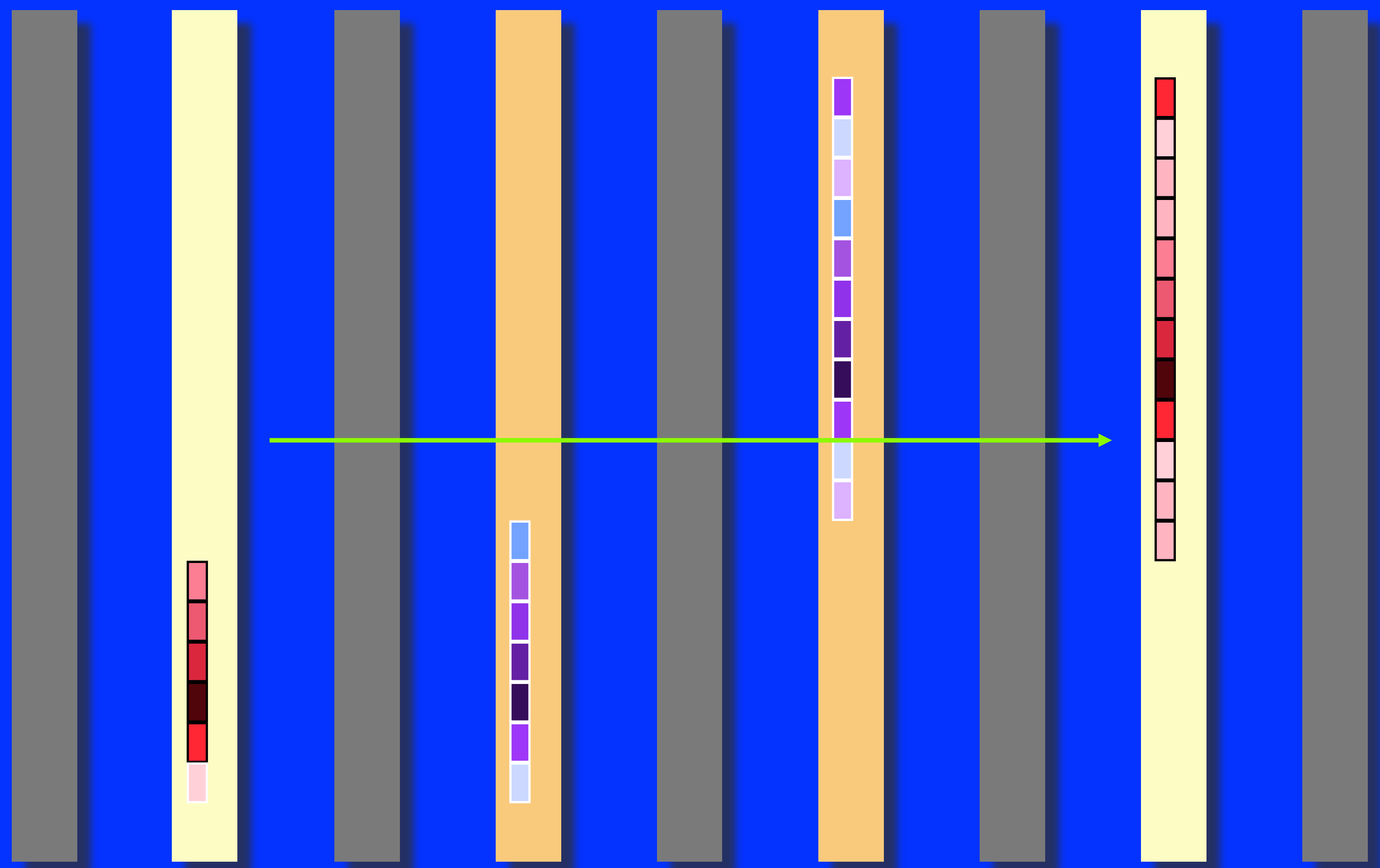


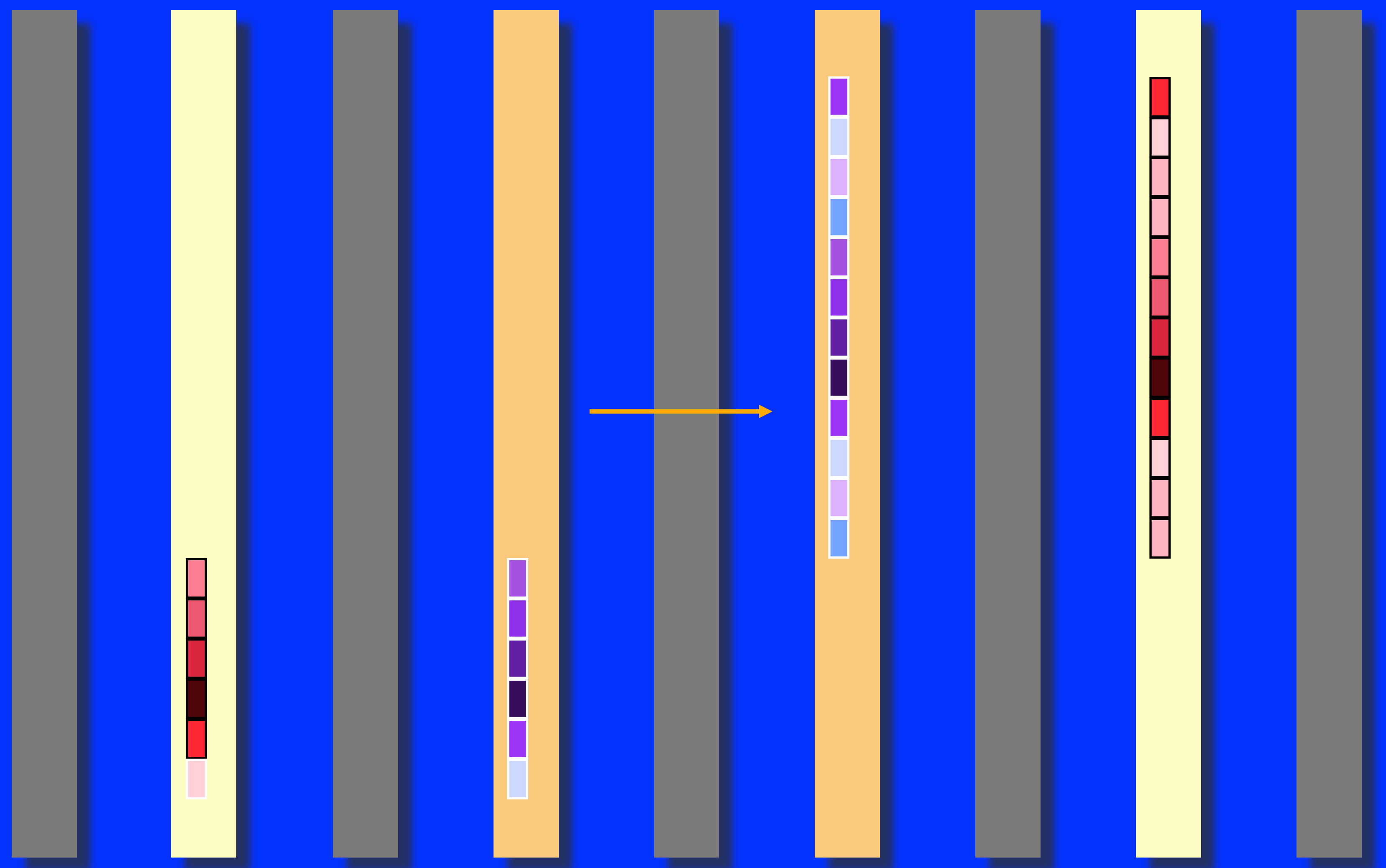


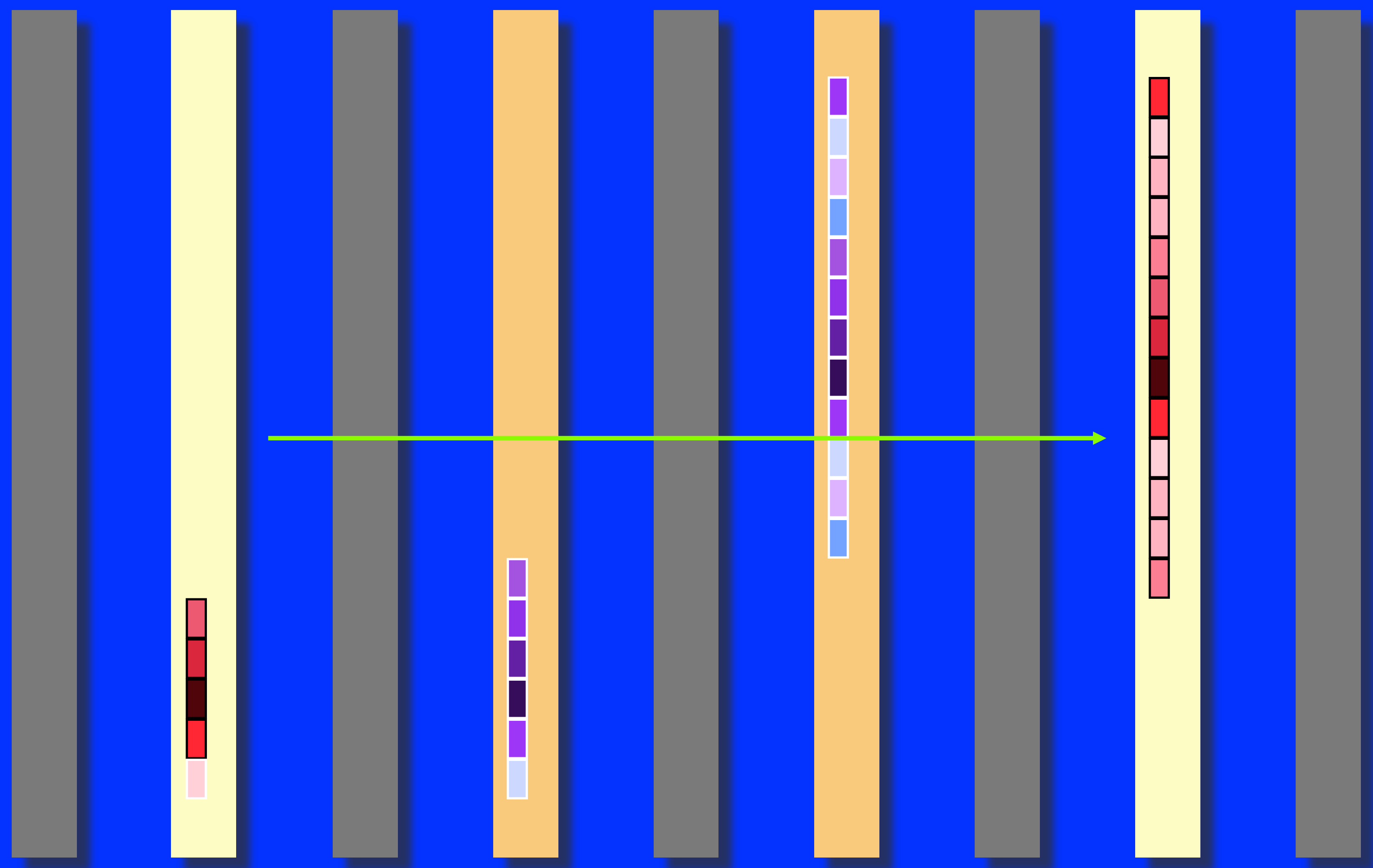


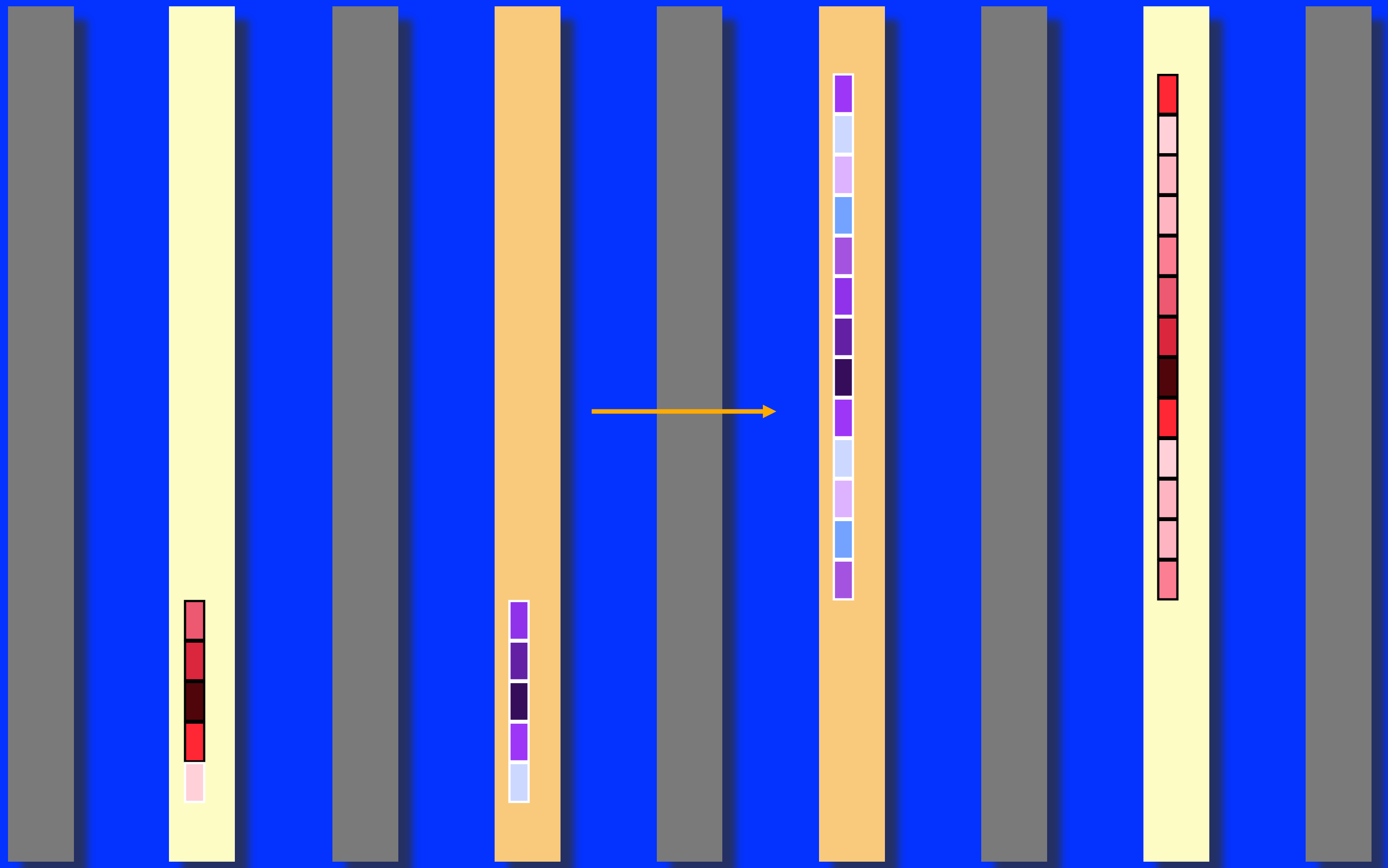


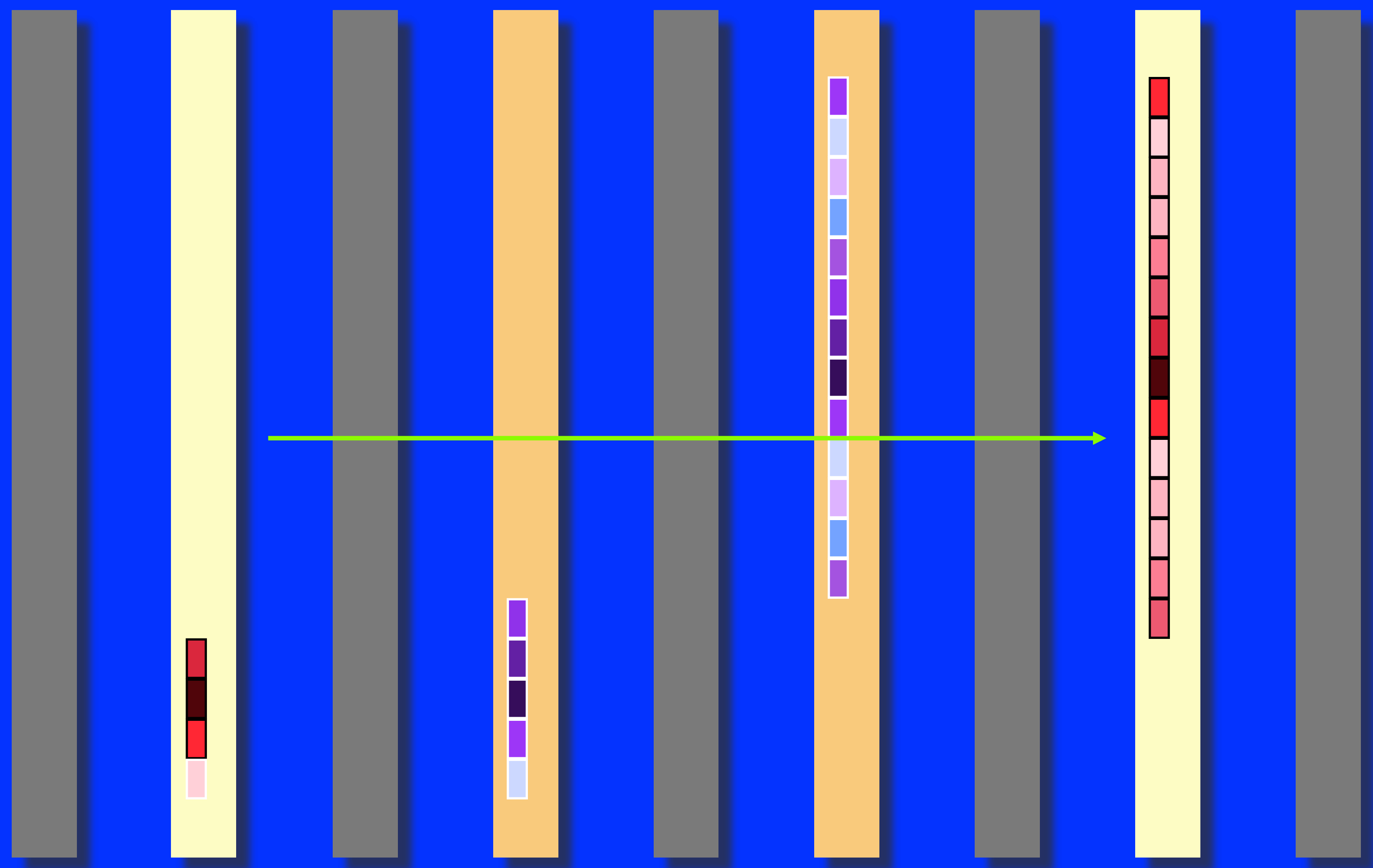


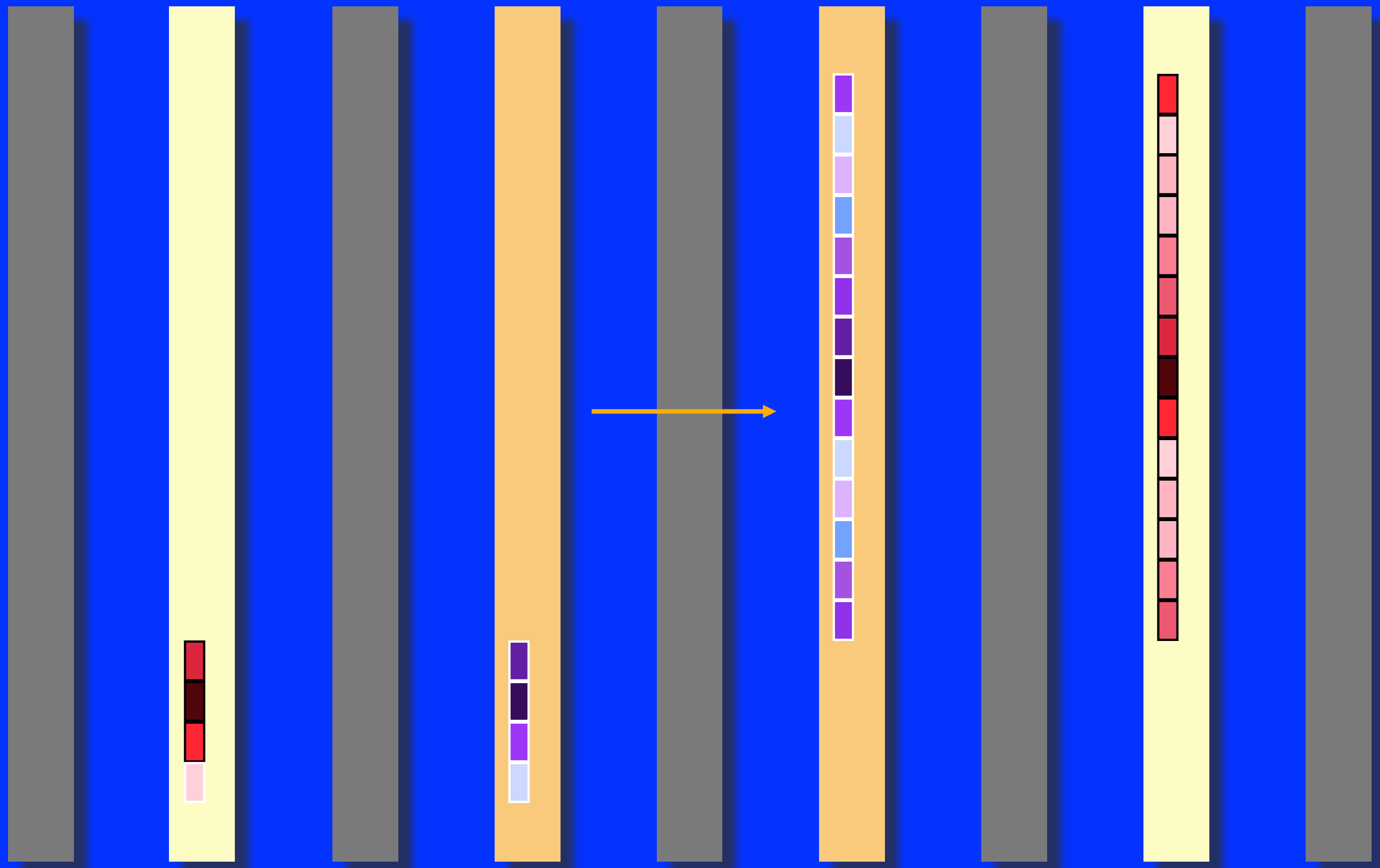


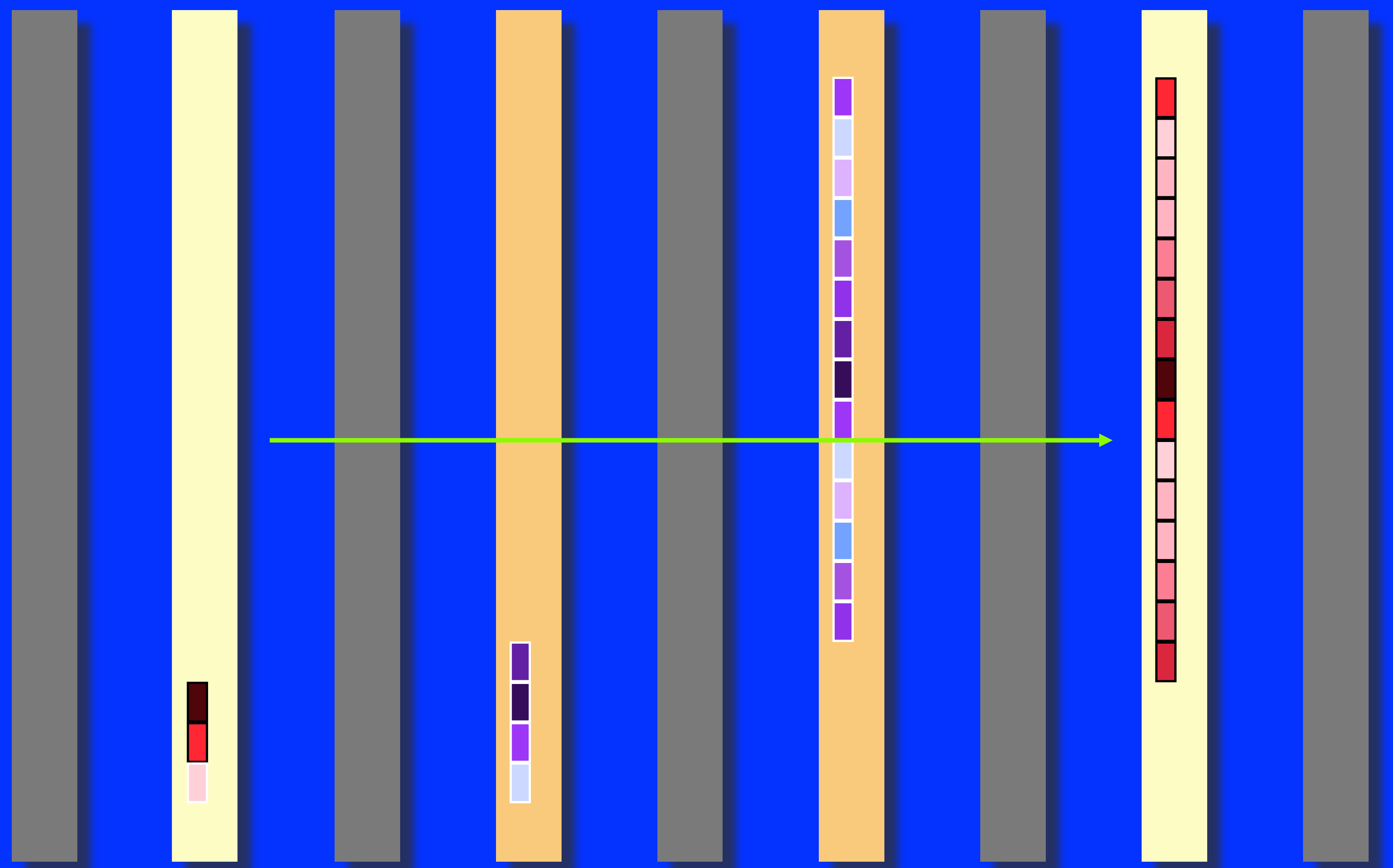


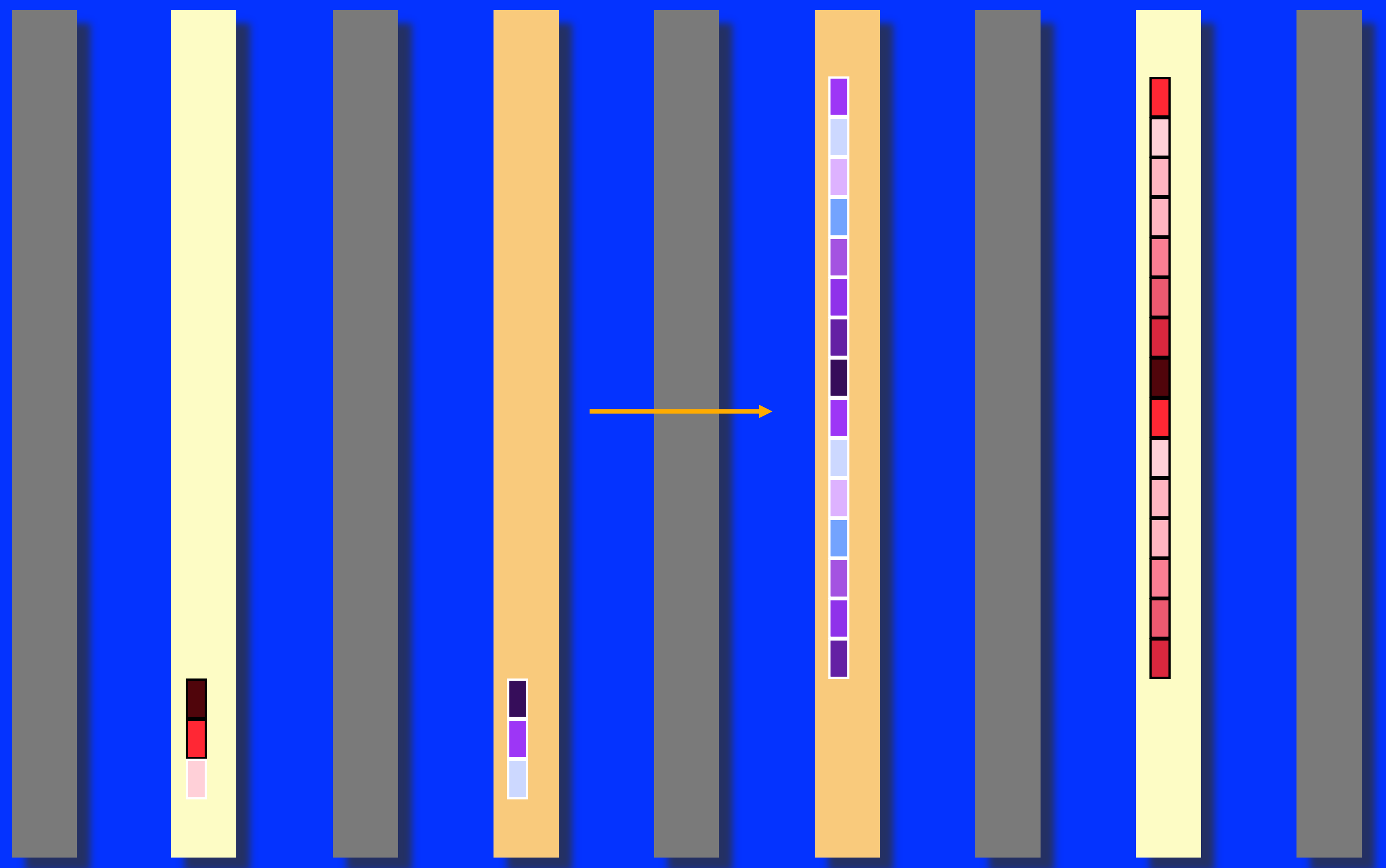


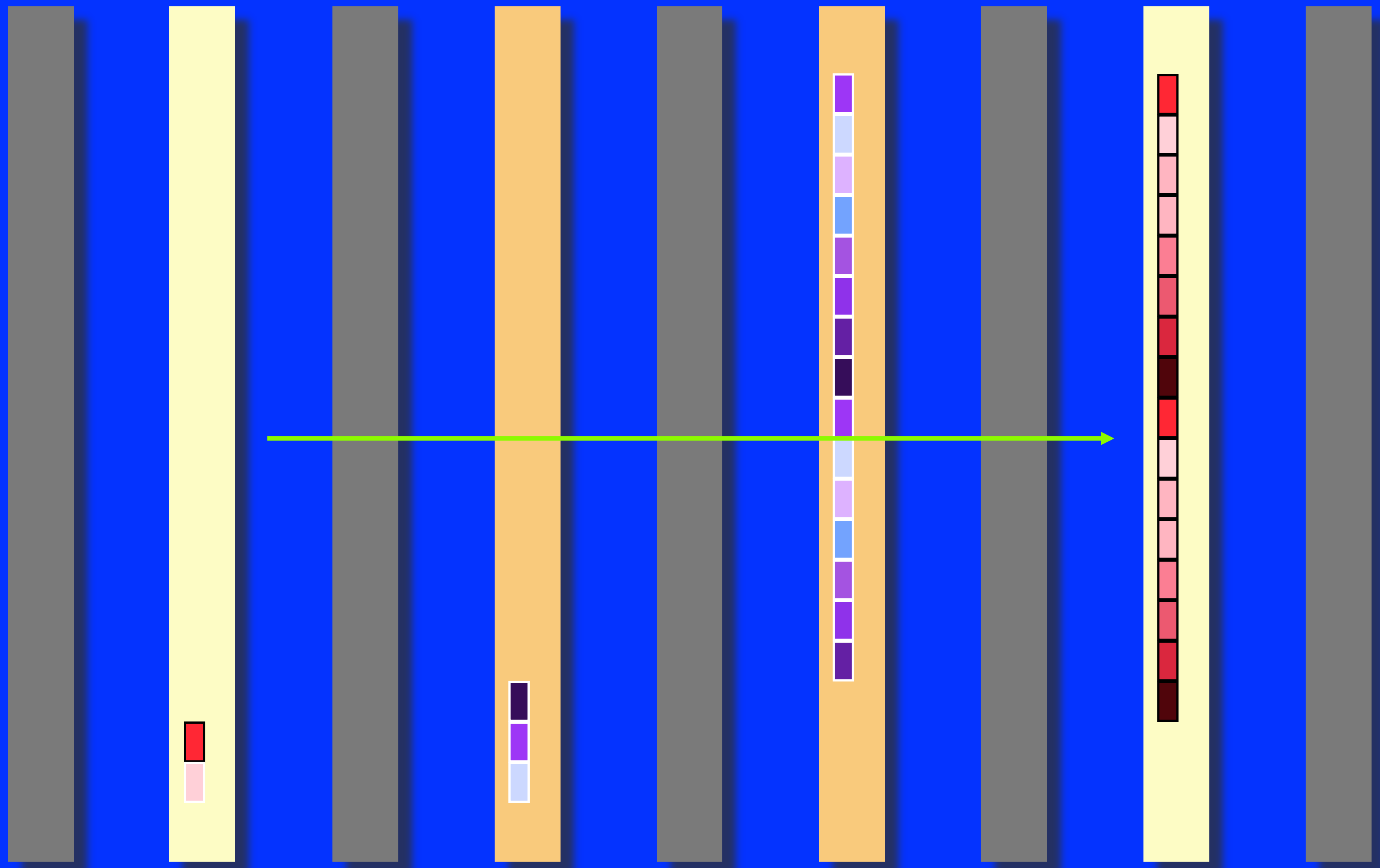


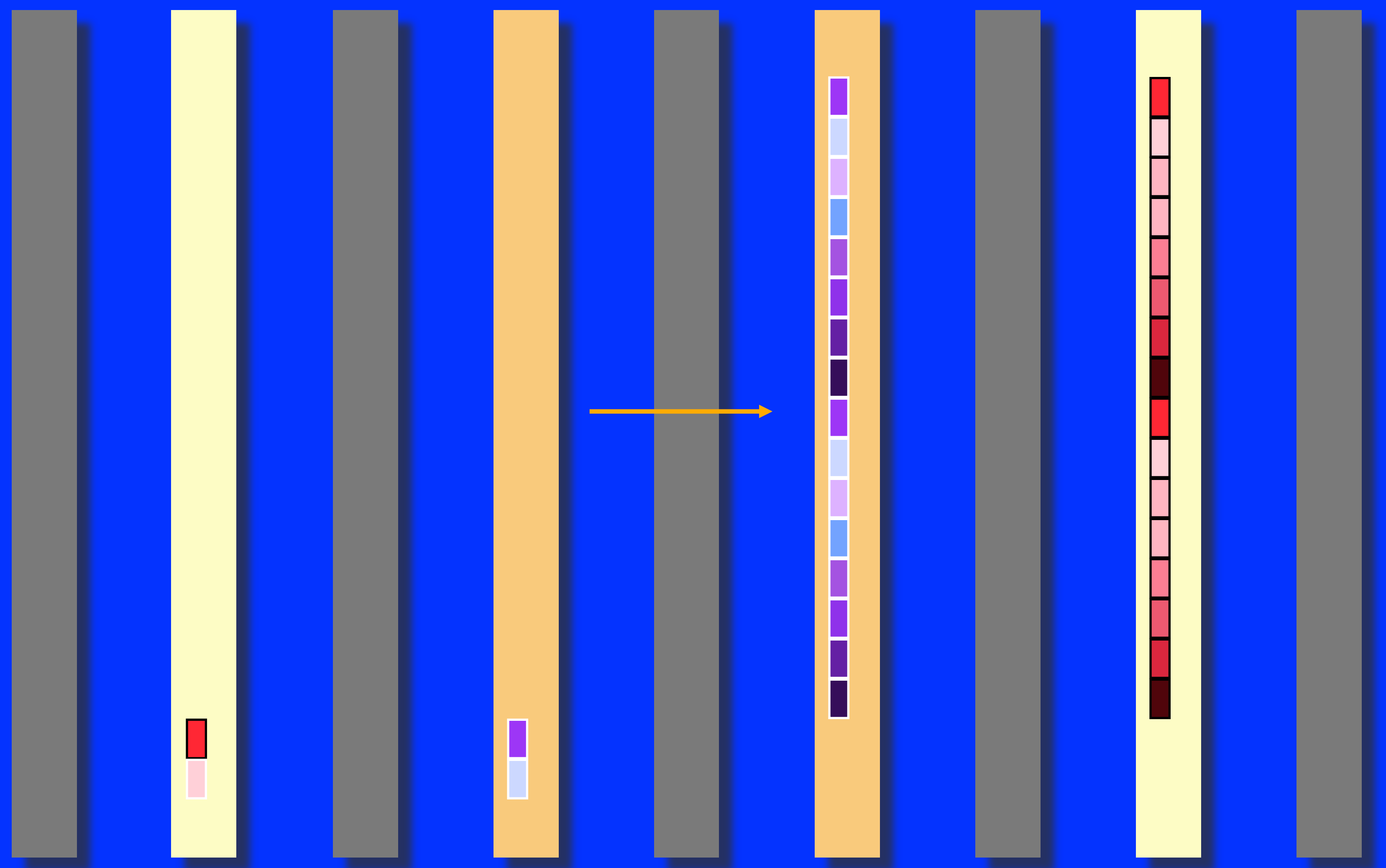


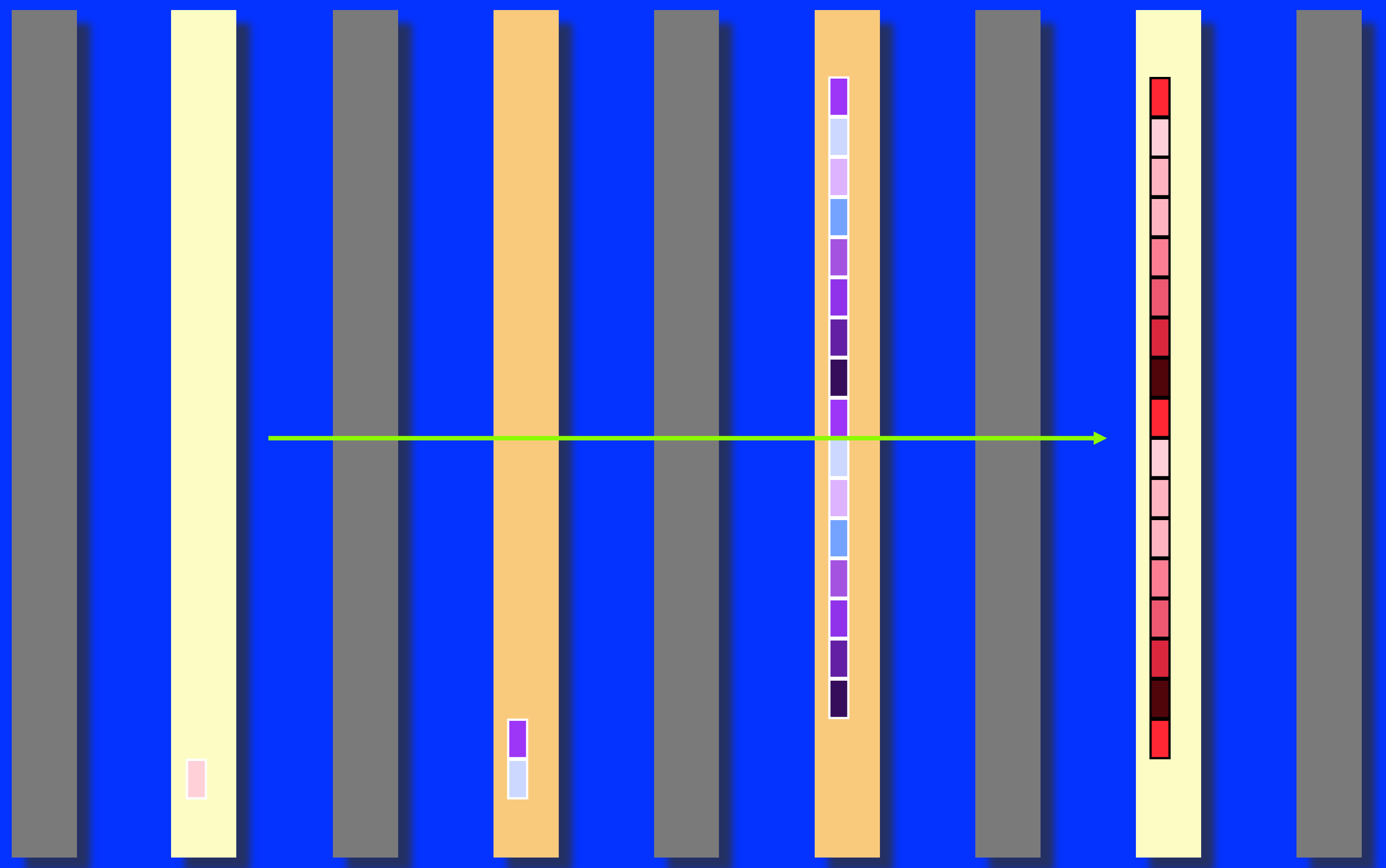


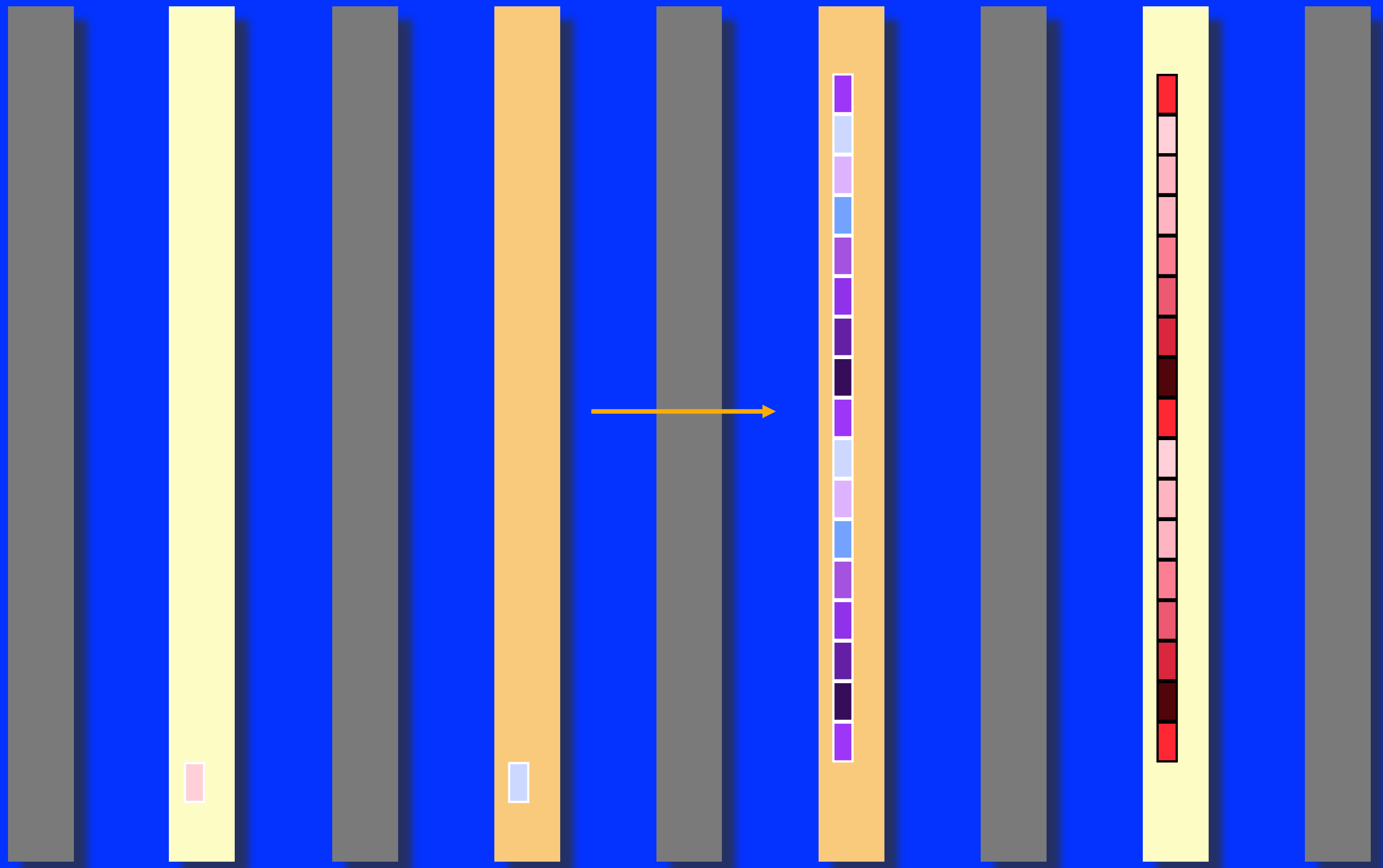


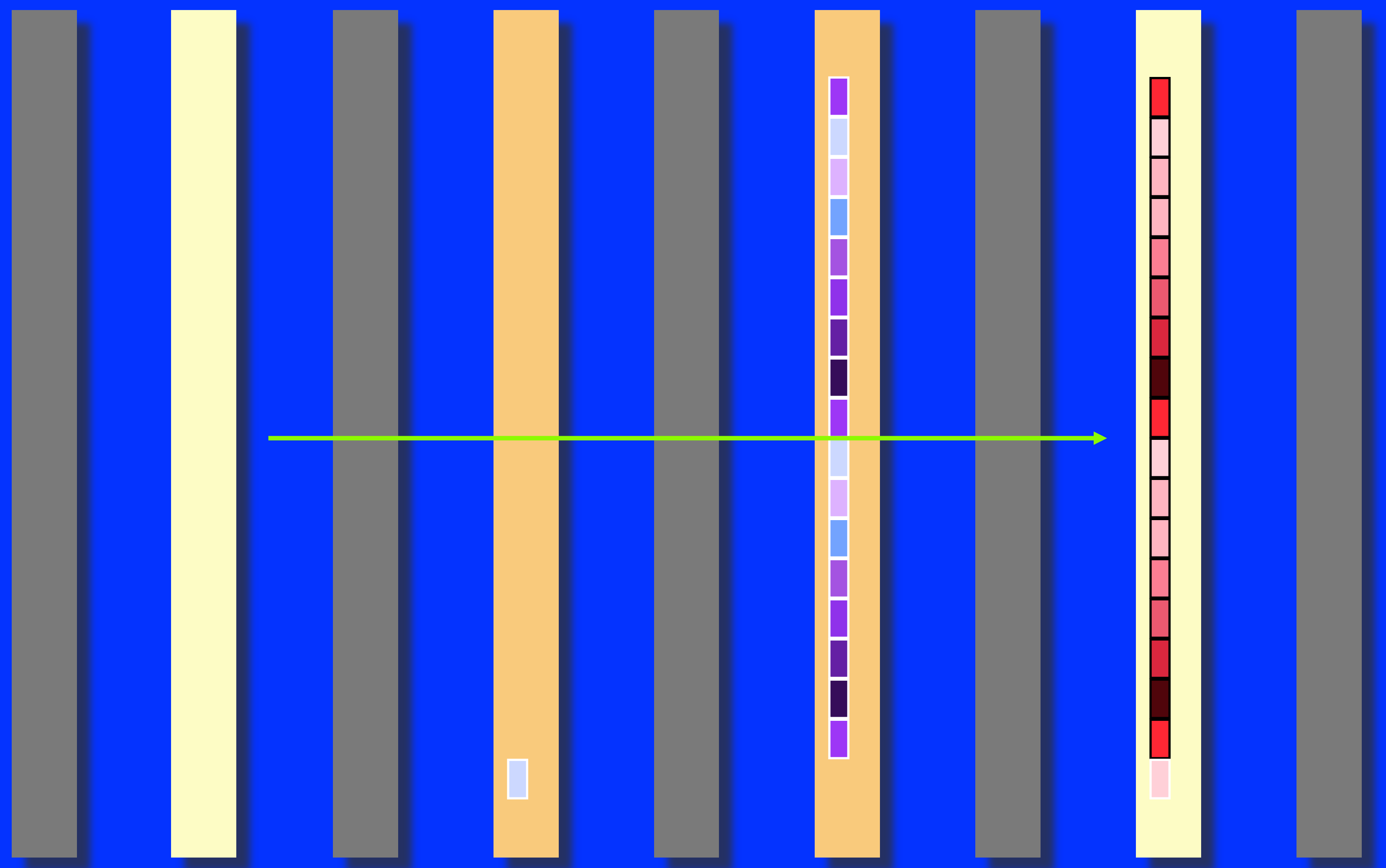


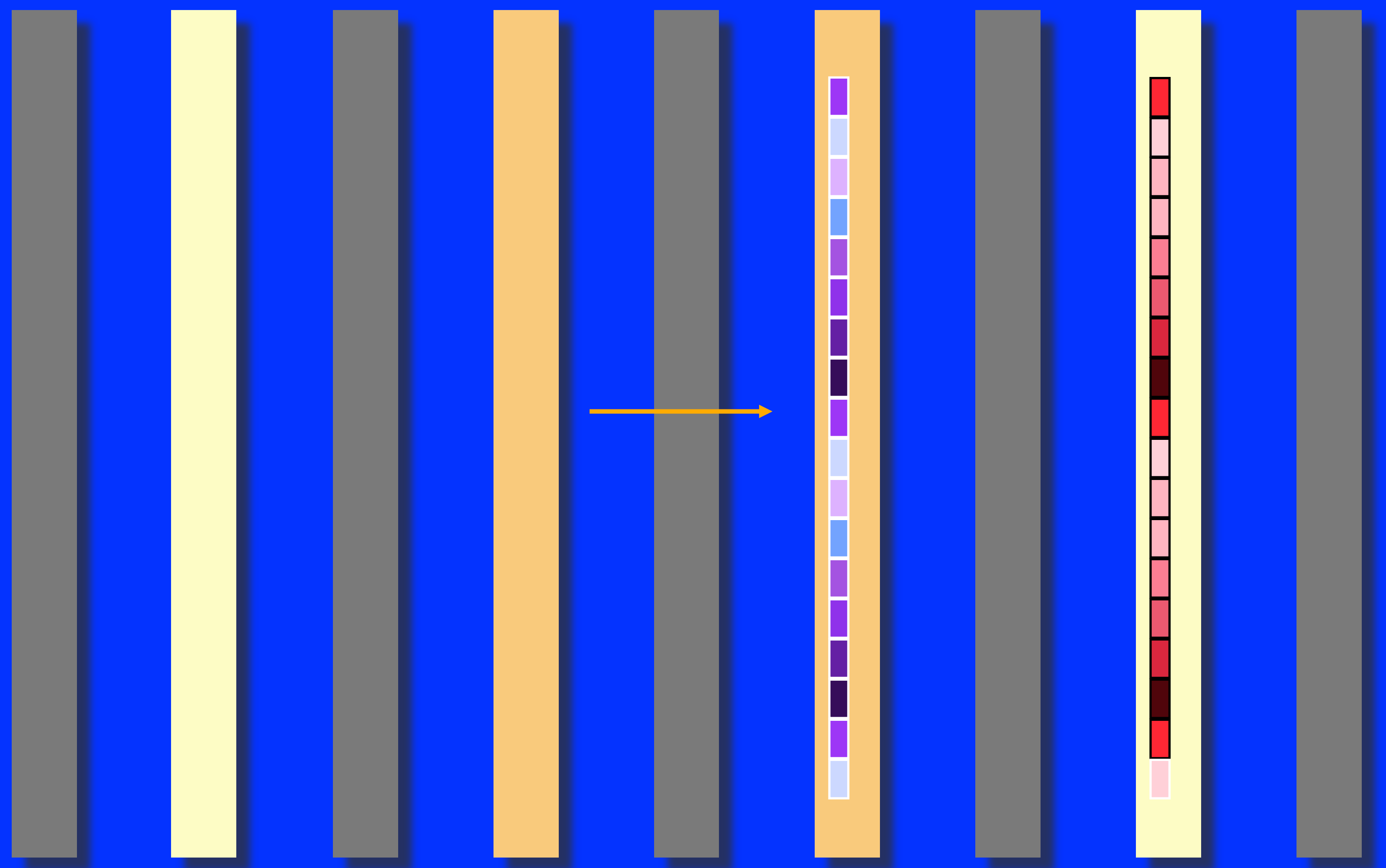


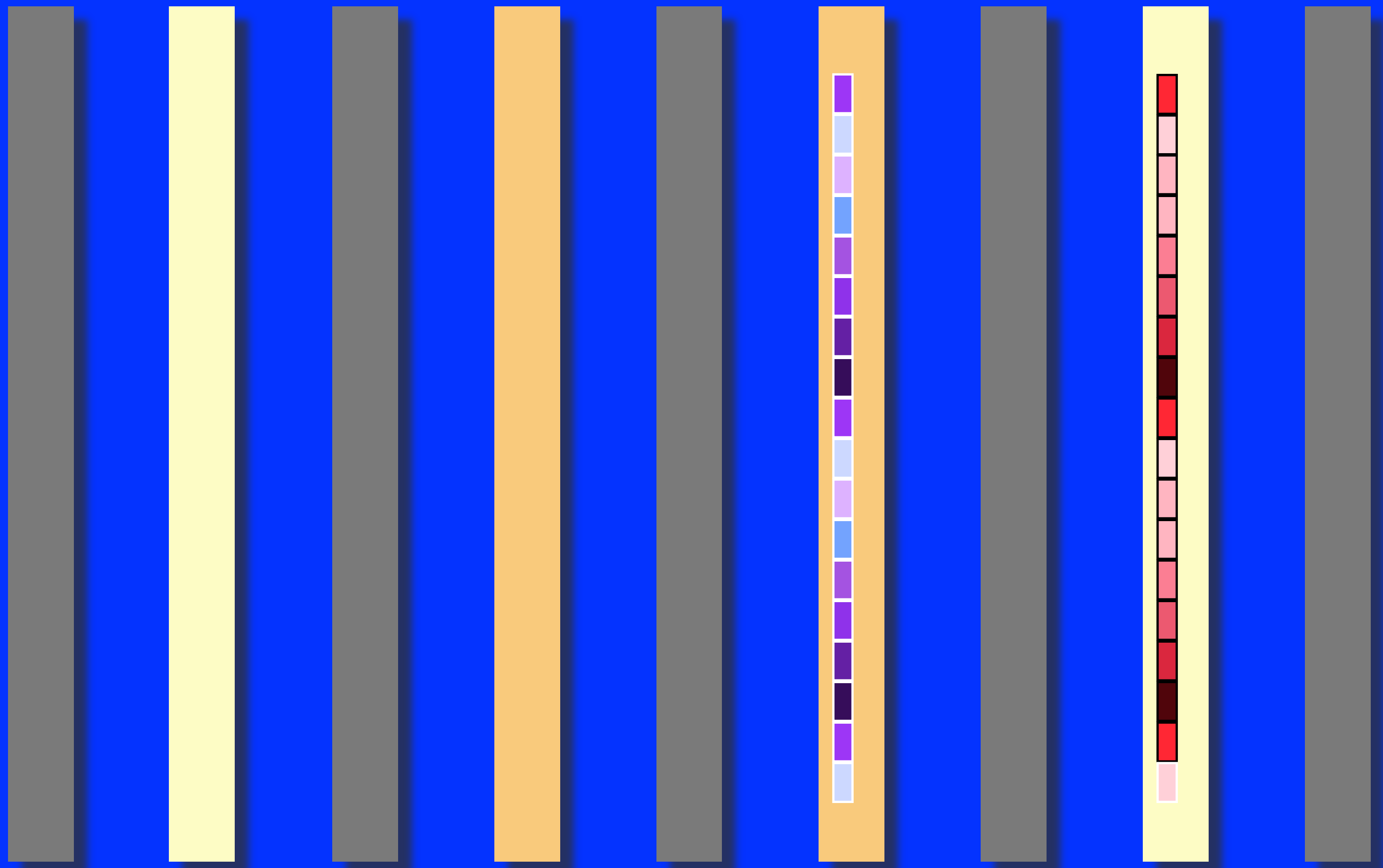












Outline

Part I: Theory

- Model of computation
- Collective communications
- A building block approach to library implementation

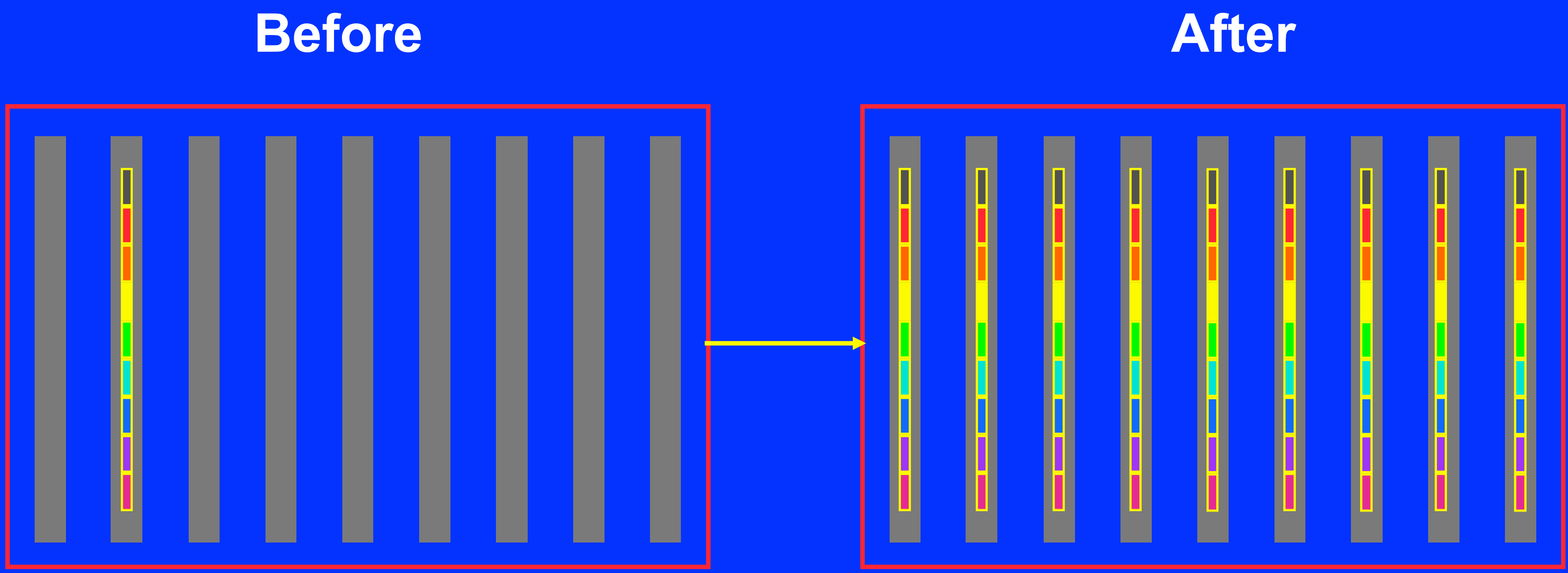
Part II: Practice

- Implementation on the Paragon
- Performance results
- Applications

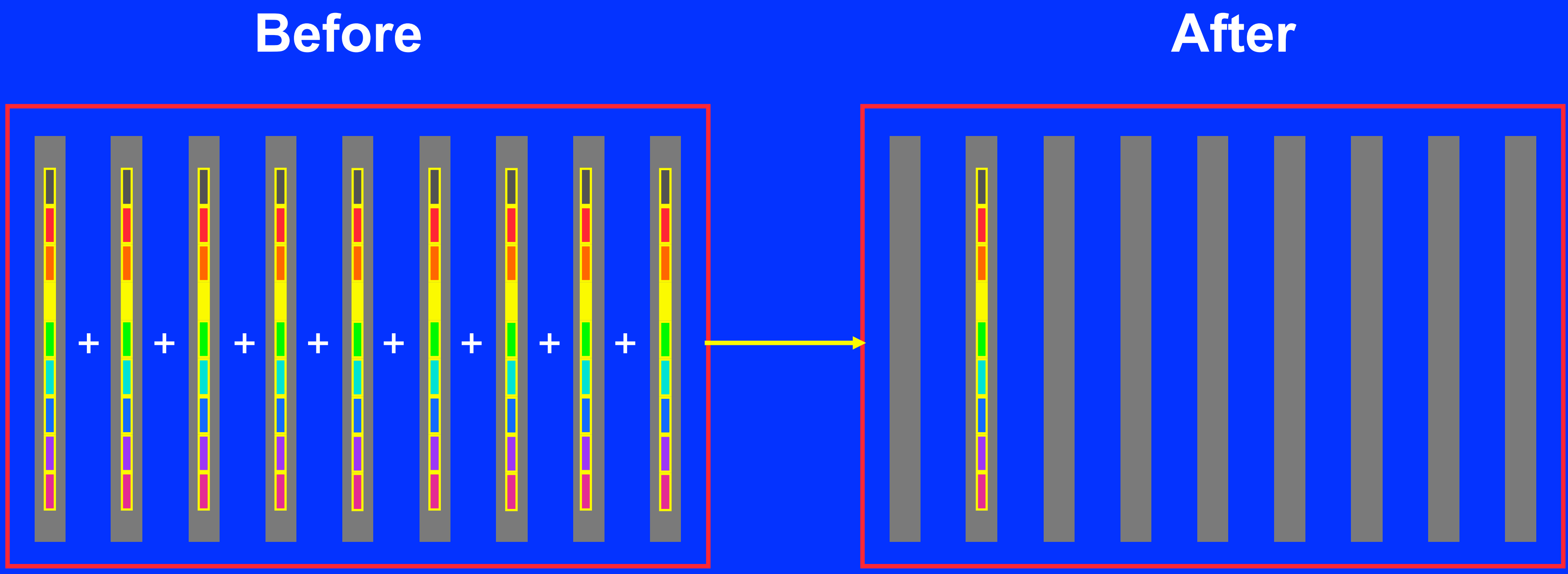
Collective Communications

- Broadcast
- Reduce(-to-one)
- Scatter
- Gather
- Allgather
- Reduce-scatter
- Allreduce

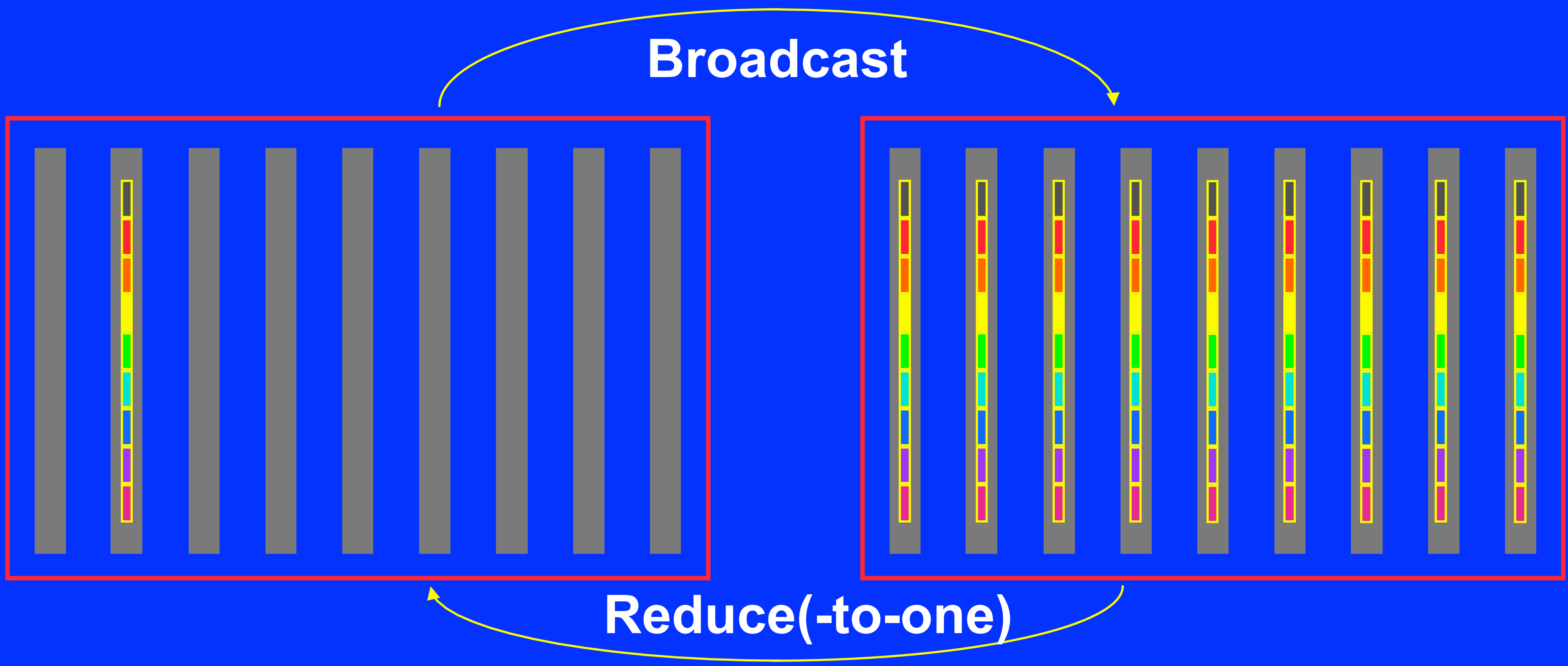
Broadcast



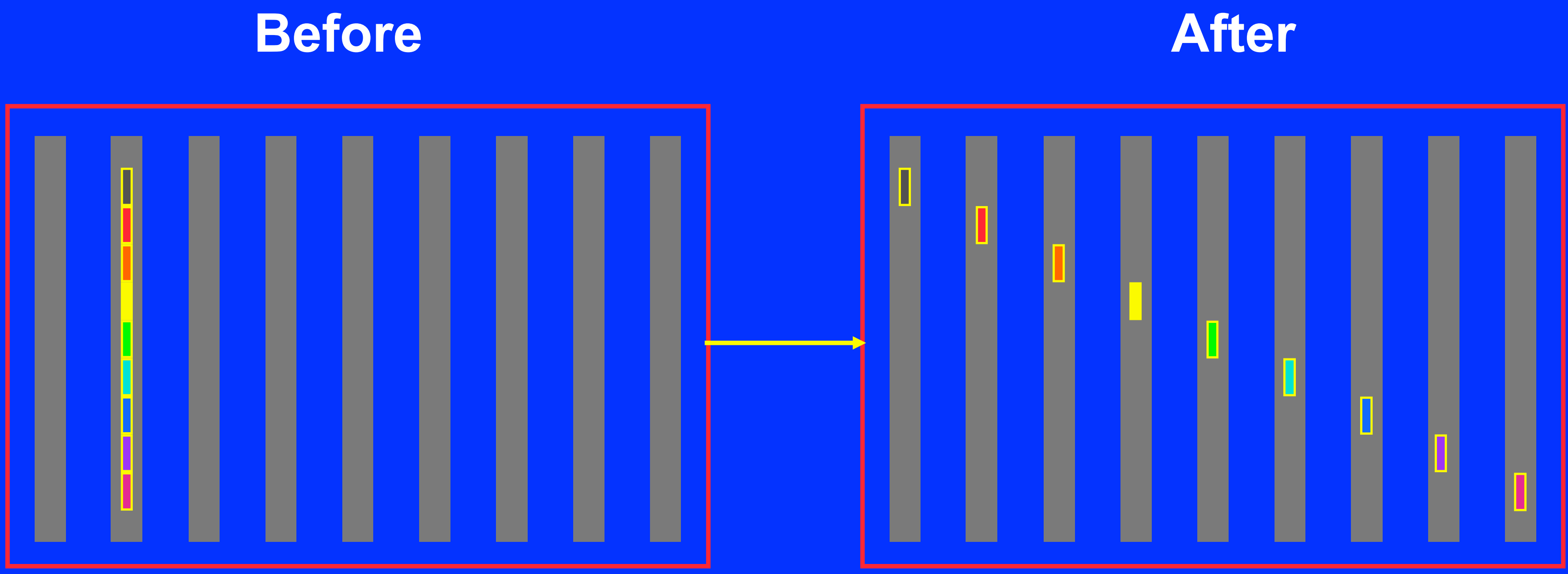
Reduce(-to-one)



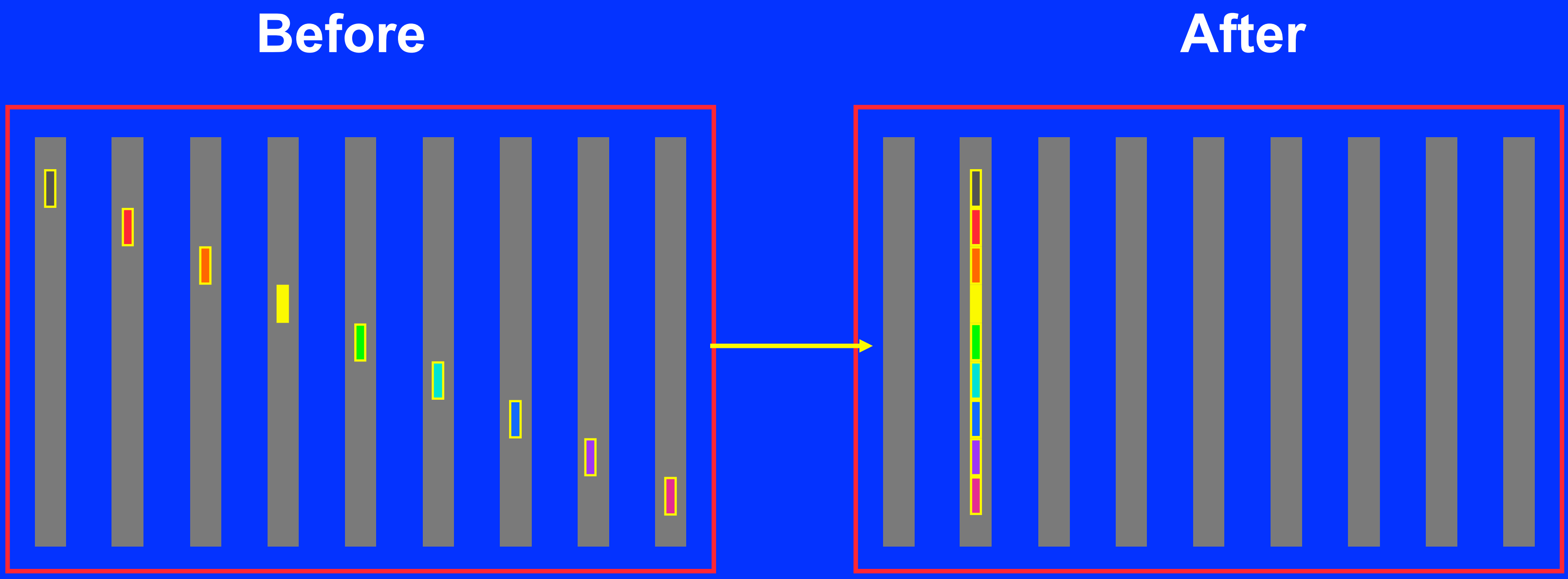
Broadcast/Reduce(-to-one)



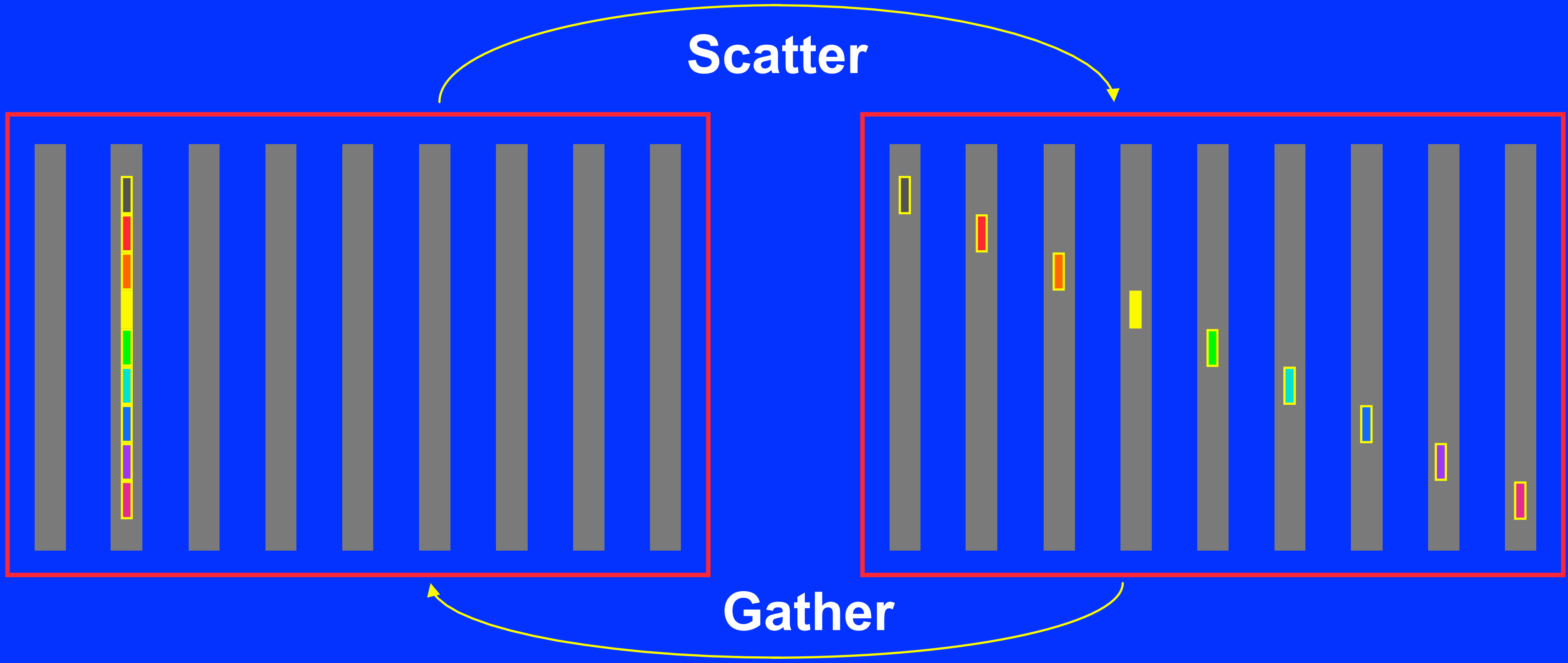
Scatter



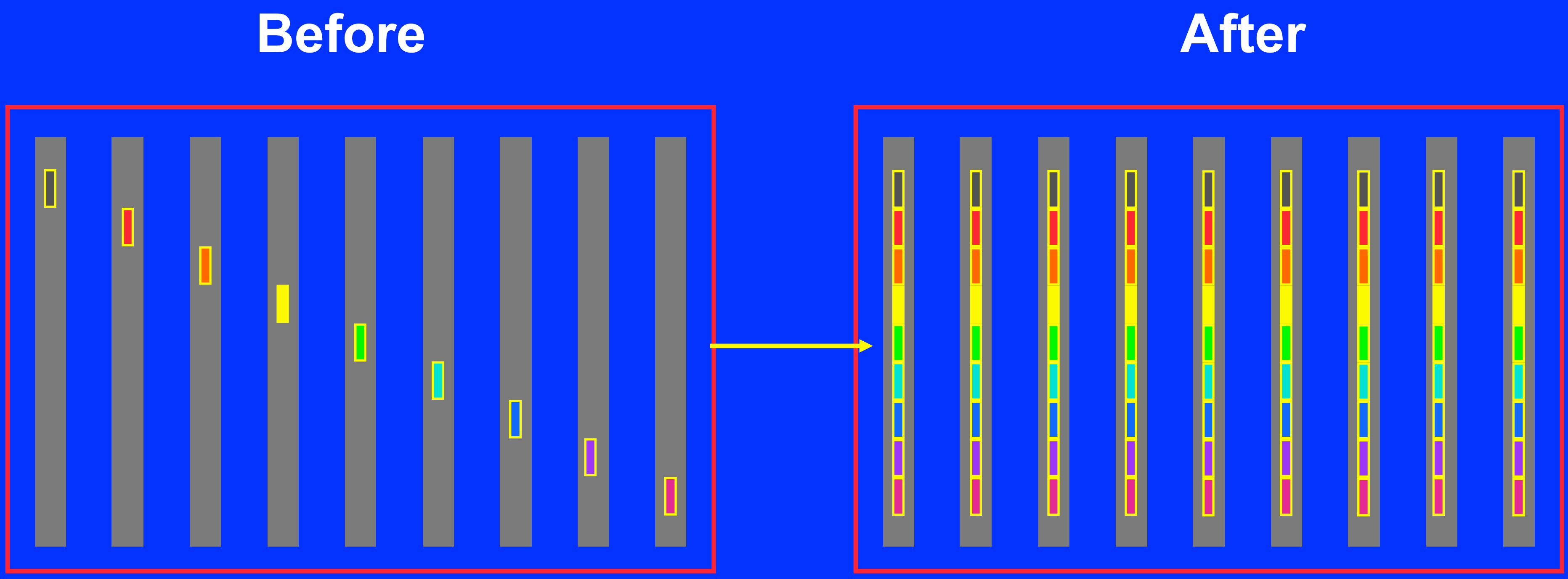
Gather



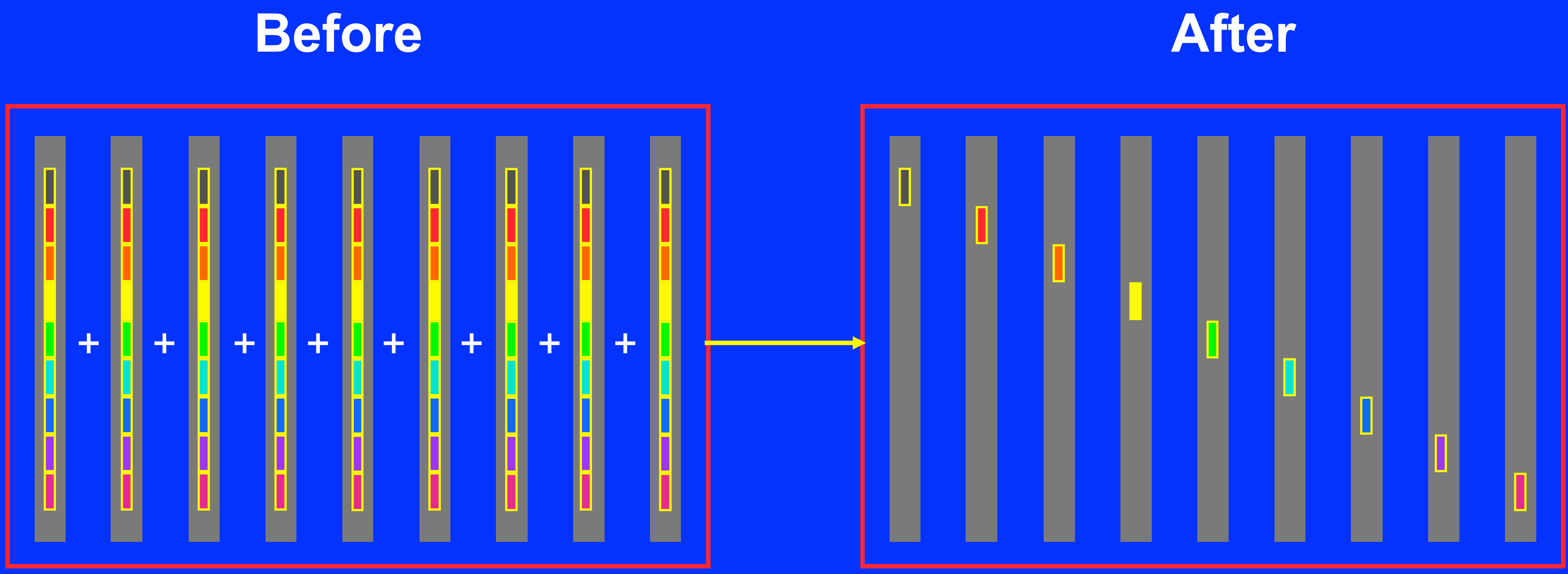
Scatter/Gather



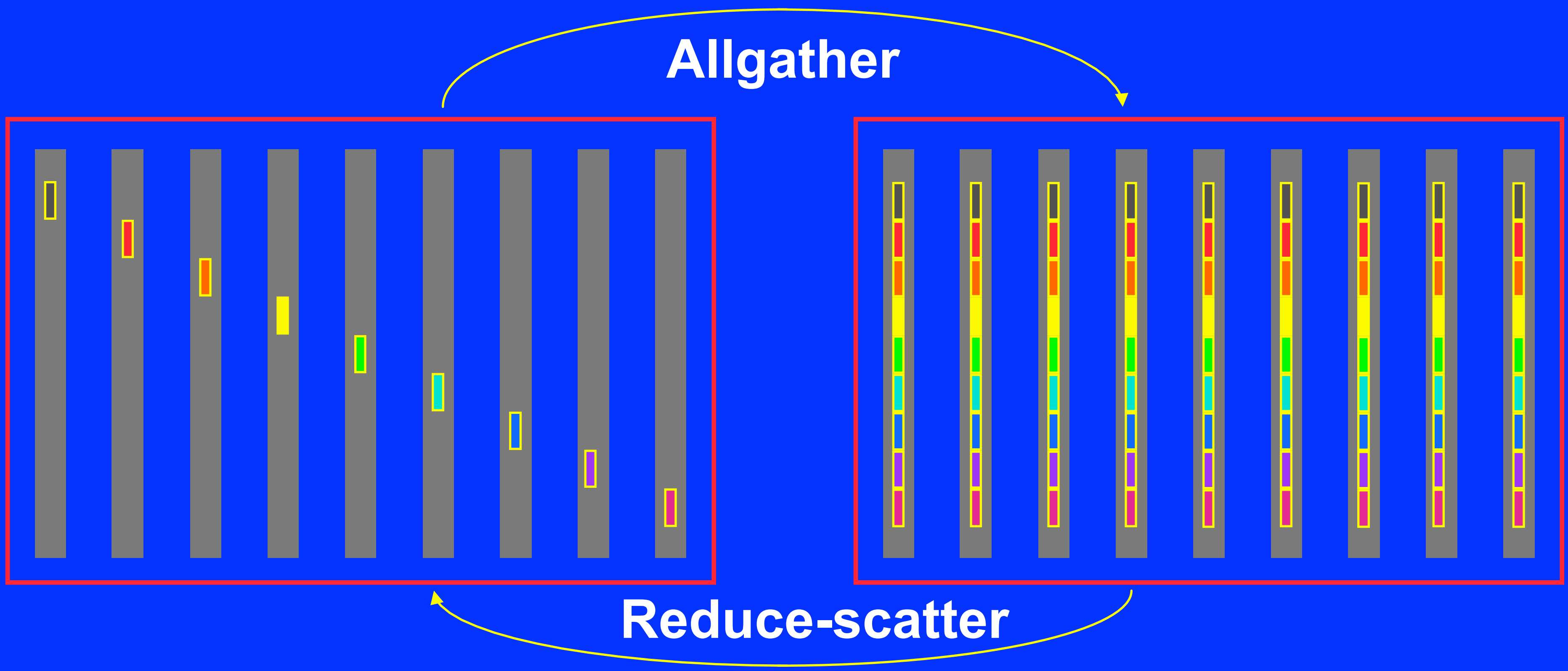
Allgather



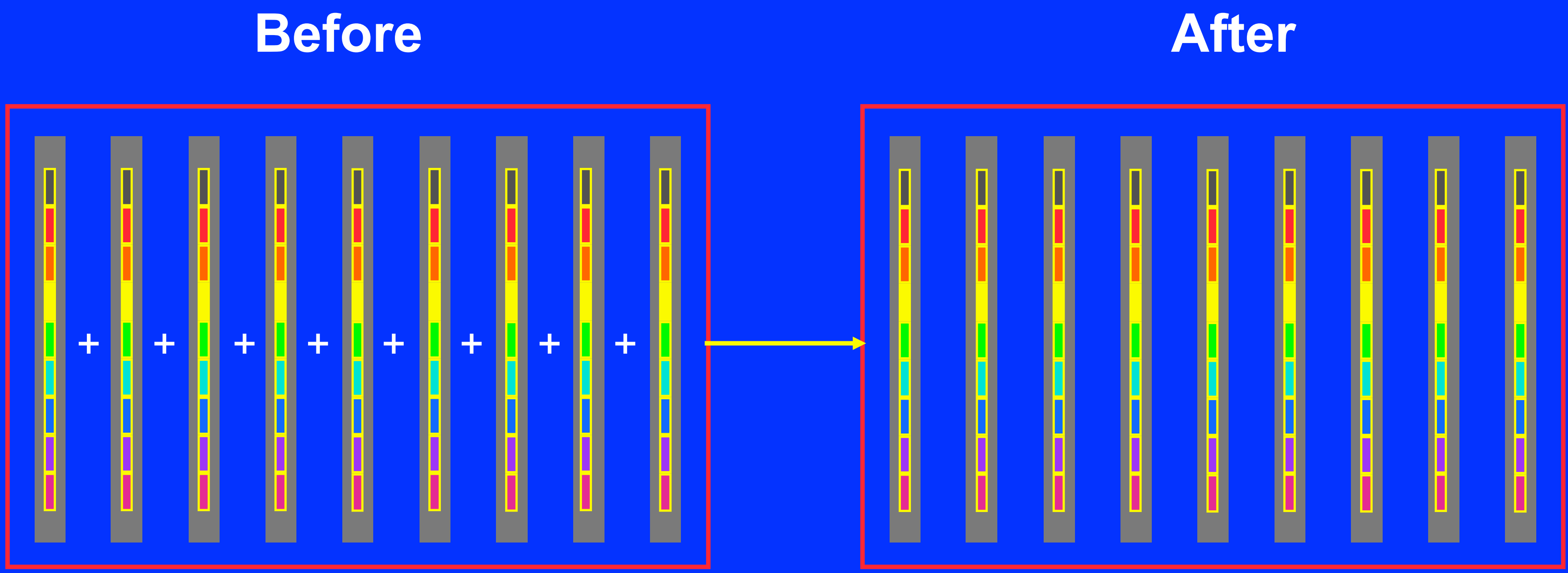
Reduce-scatter



Allgather/Reduce-scatter



Allreduce



Lower bounds (startup)

Broadcast	$\lceil log(p) \rceil \alpha$
Reduce(-to-one)	$\lceil log(p) \rceil \alpha$
Scatter/Gather	$\lceil log(p) \rceil \alpha$
Allgather	$\lceil log(p) \rceil \alpha$
Reduce-scatter	$\lceil log(p) \rceil \alpha$
Allreduce	$\lceil log(p) \rceil \alpha$

Lower bounds (bandwidth)

• Broadcast	$n\beta$
• Reduce(-to-one)	$n\beta + \frac{p-1}{p} n\gamma$
• Scatter/Gather	$\frac{p-1}{p} n\beta$
• Allgather	$\frac{p-1}{p} n\beta$
• Reduce-scatter	$\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$
• Allreduce	$2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$

Outline

Part I: Theory

- Model of computation
- Collective communications
- **A building block approach to library implementation**

Part II: Practice

- Implementation on the Paragon
- Performance results
- Applications

A building block approach to library implementation

- Short vector case
- Long vector case
- Hybrid algorithms

Short vector case

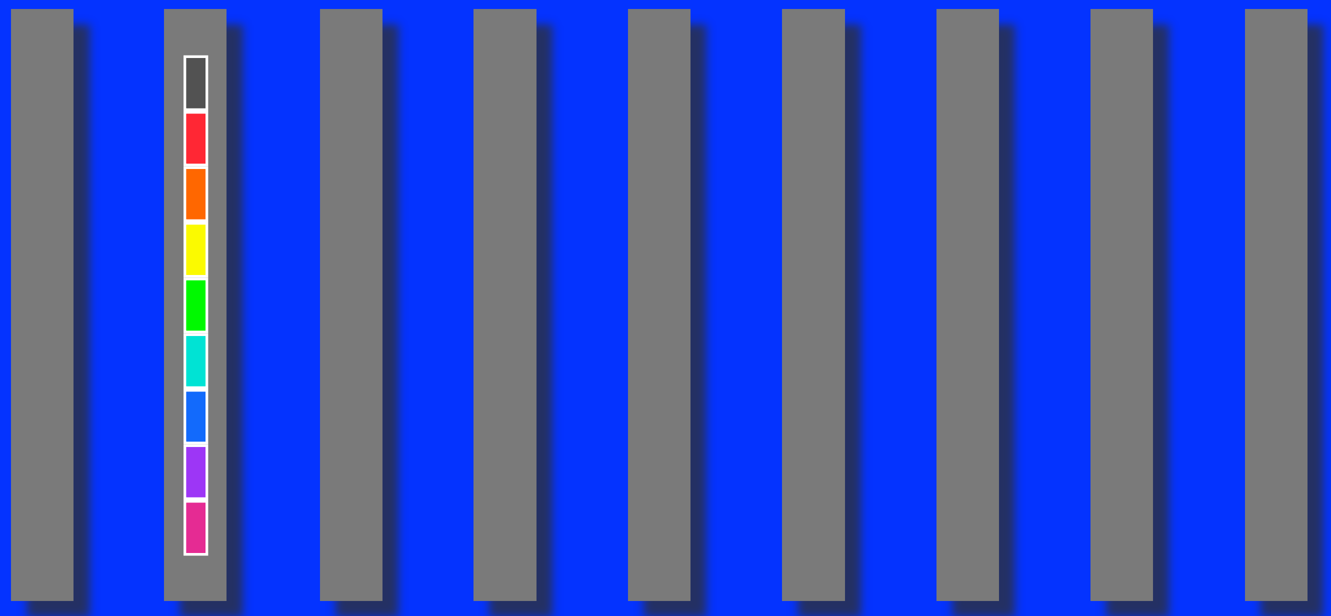
- Primary concern:
  - algorithms must have low latency cost
- Secondary concerns:
  - algorithms must work for arbitrary number of nodes
    - » in particular, not just for power-of-two numbers of nodes
  - algorithms should avoid network conflicts
    - » not absolutely necessary, but nice if possible

Minimum spanning tree based algorithms

- We will show how the following building blocks:
  - broadcast/combine-to-one
  - scatter/gather

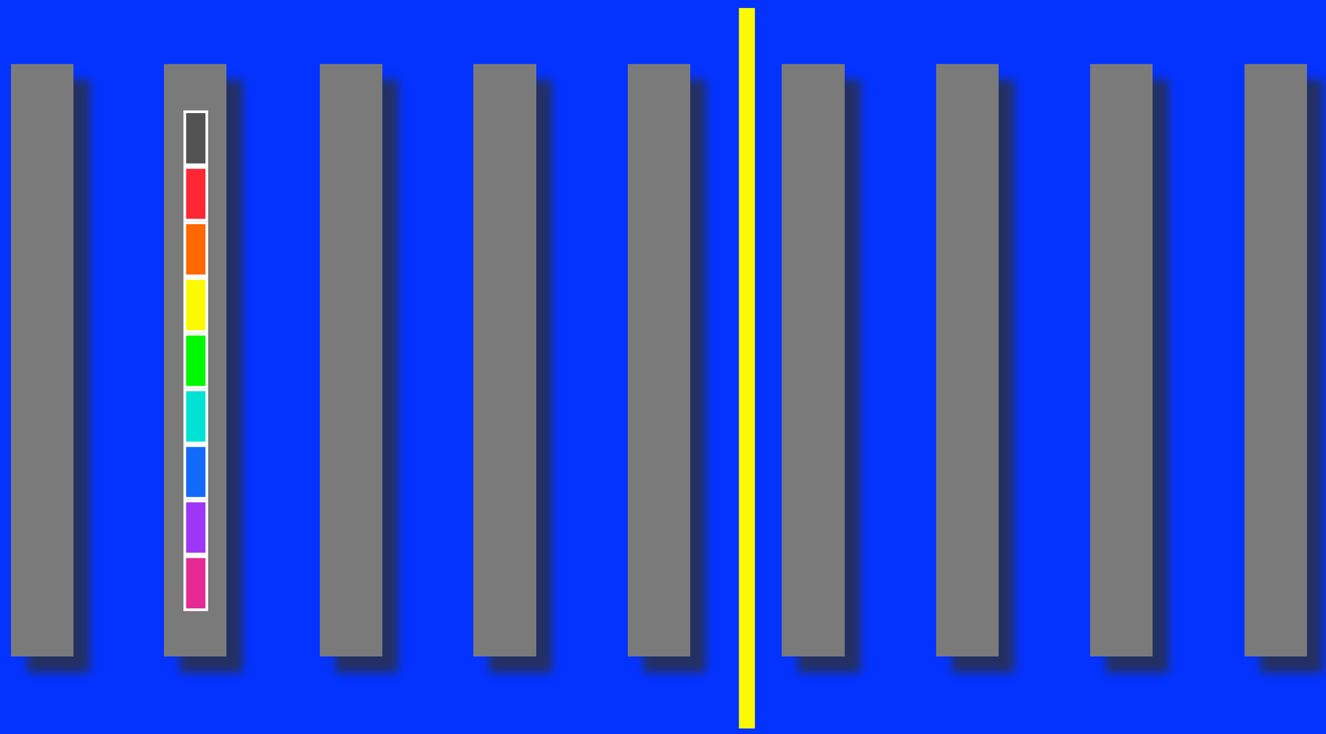
  can be implemented using minimum spanning trees embedded in the logical linear array while attaining
  - minimal latency
  - implementation for arbitrary numbers of nodes
  - no network conflicts

General principles



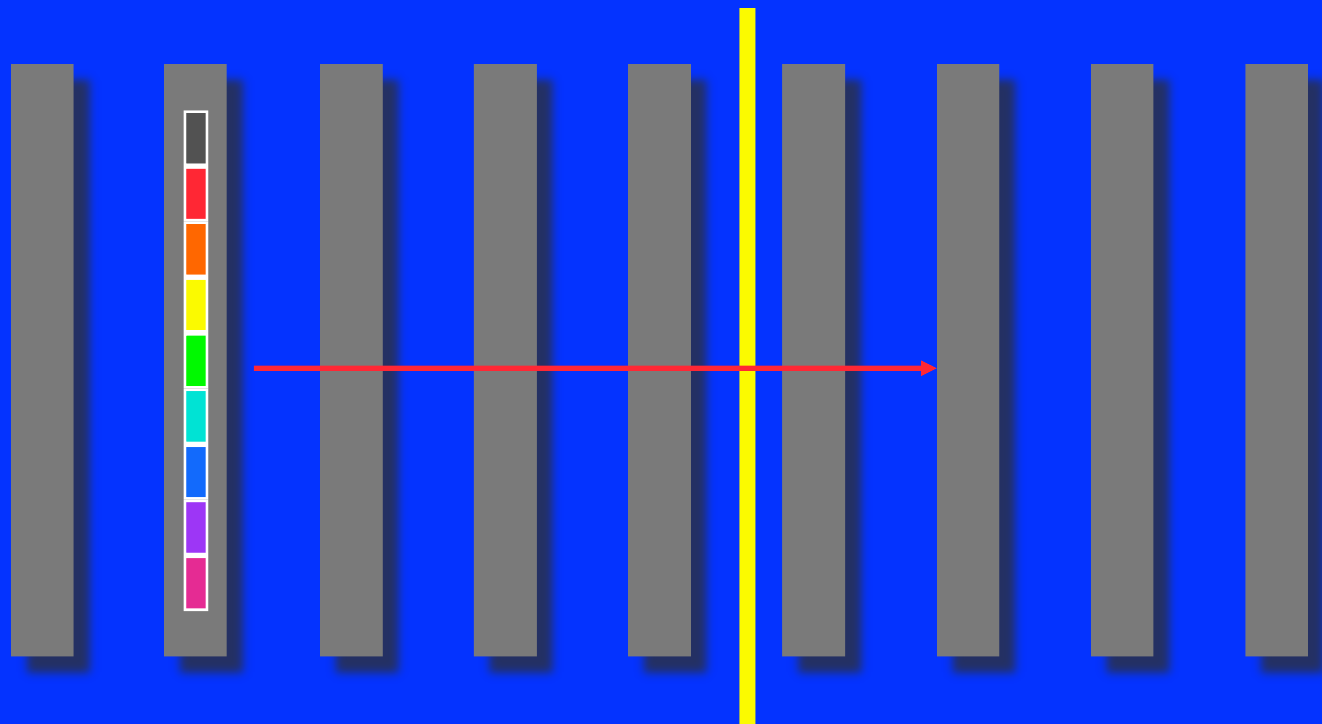
- **message starts on one processor**

General principles



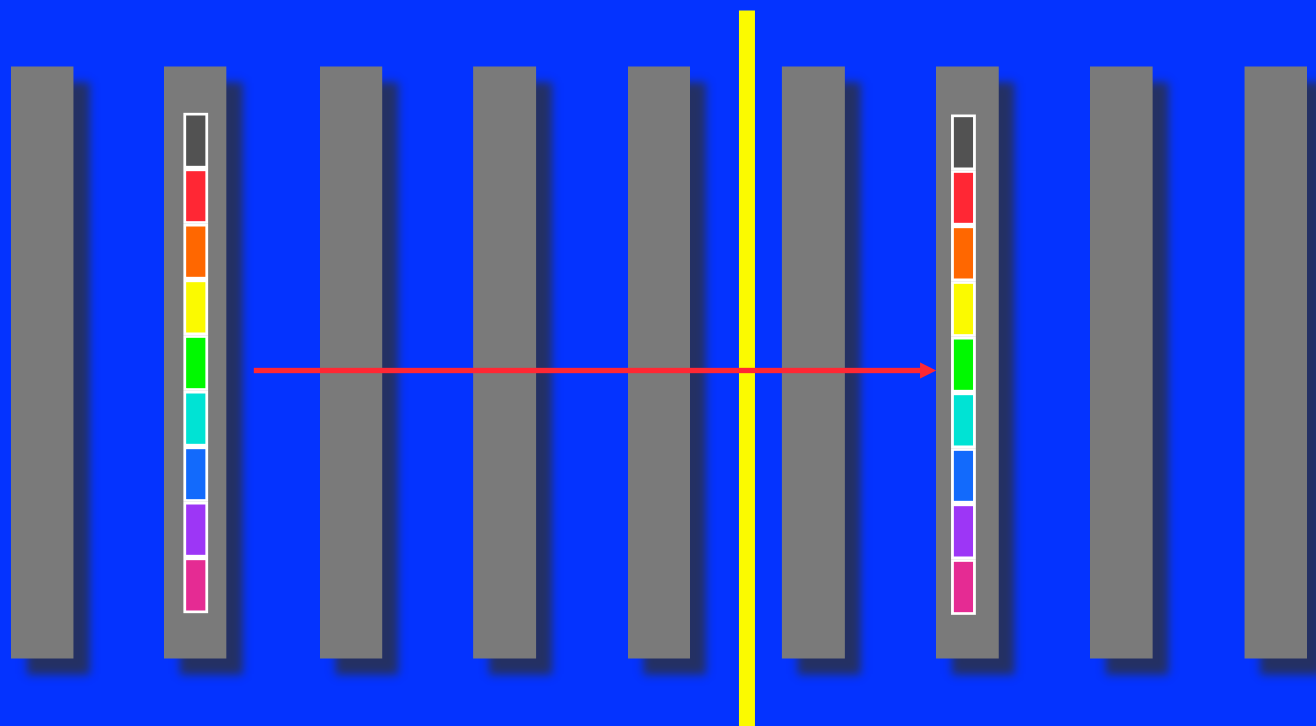
- divide logical linear array in half

General principles



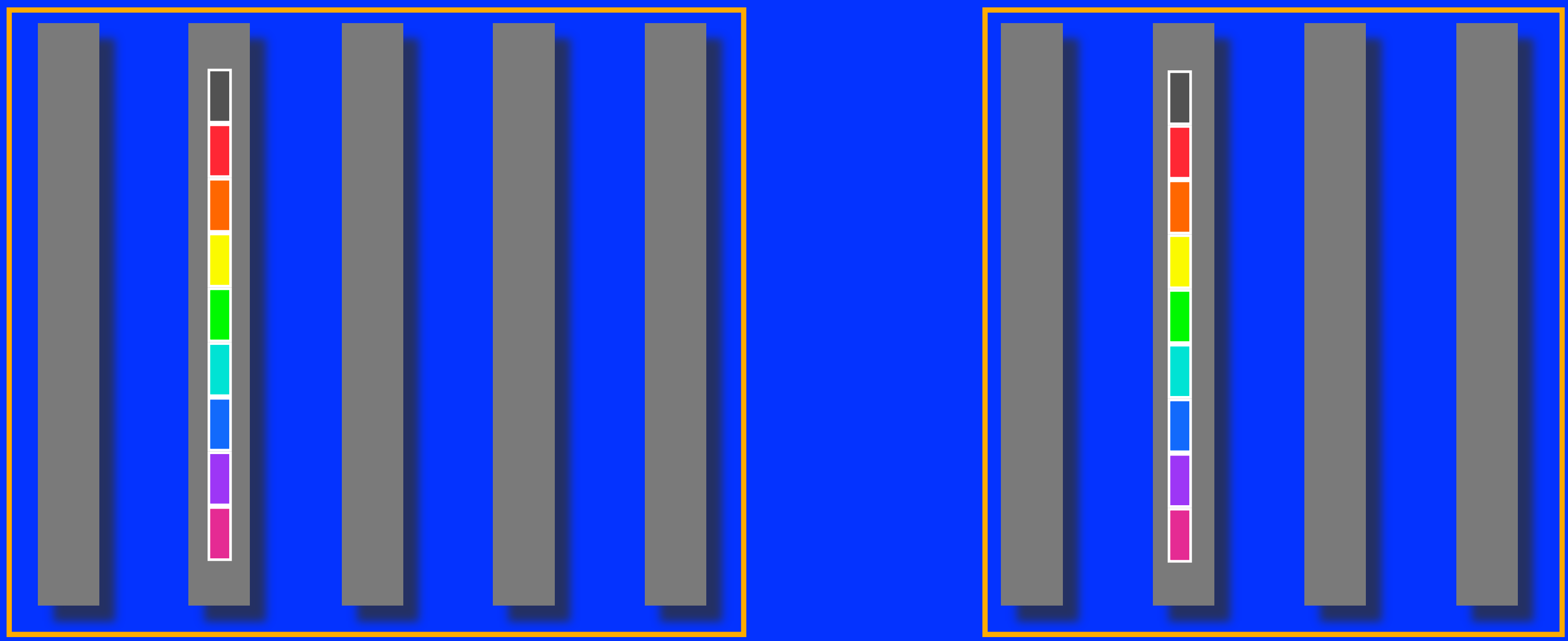
- send message to the half of the network that does not contain the current node (root) that holds the message

General principles



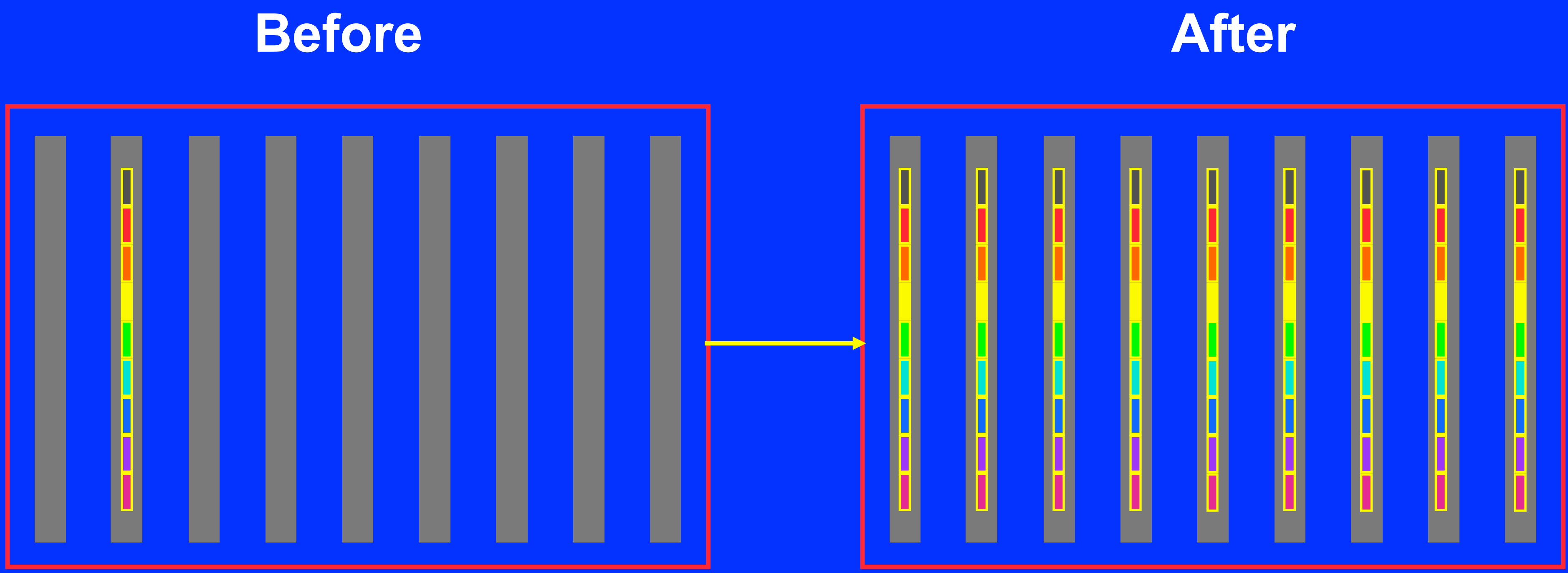
- send message to the half of the network that does not contain the current node (root) that holds the message

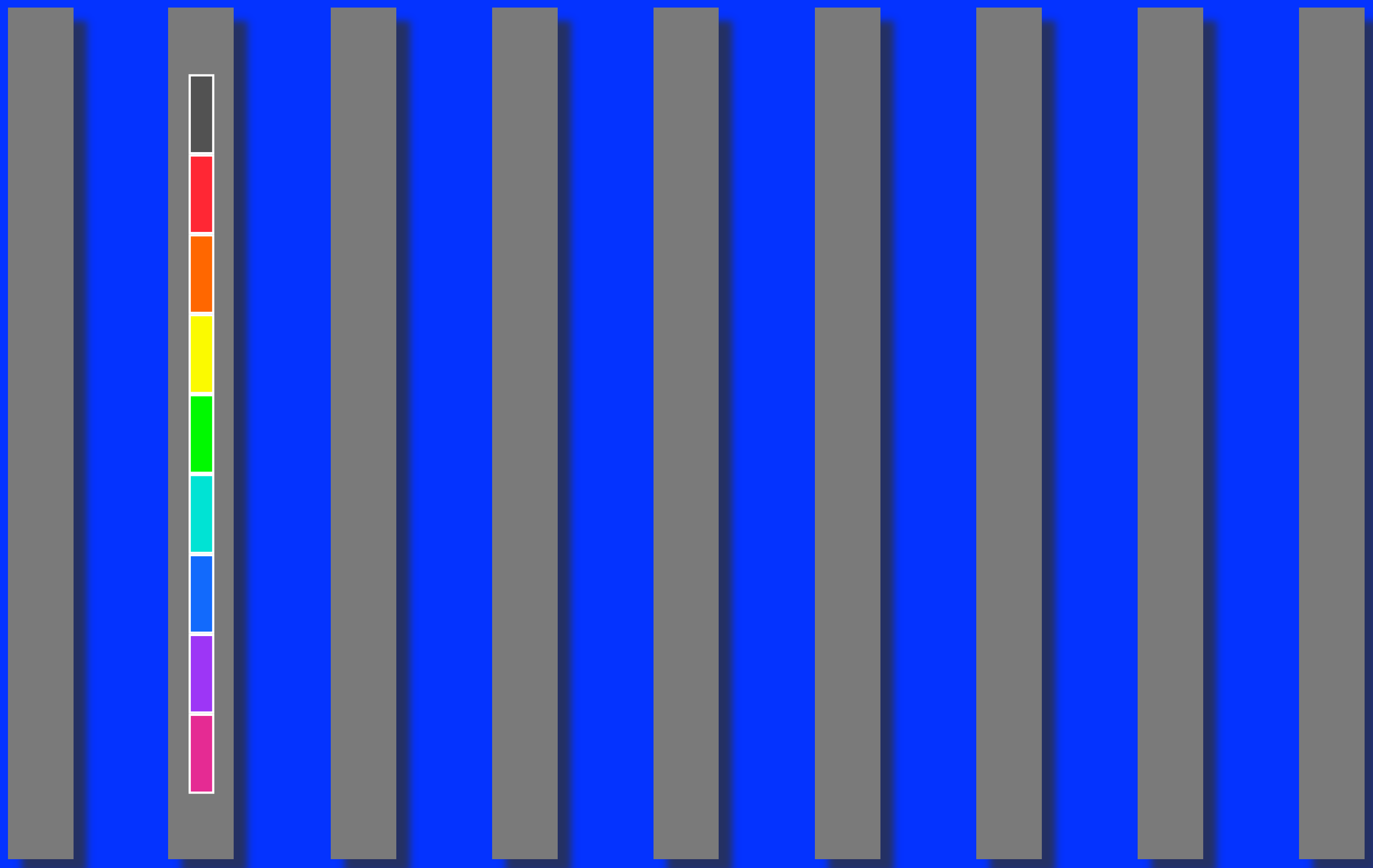
General principles

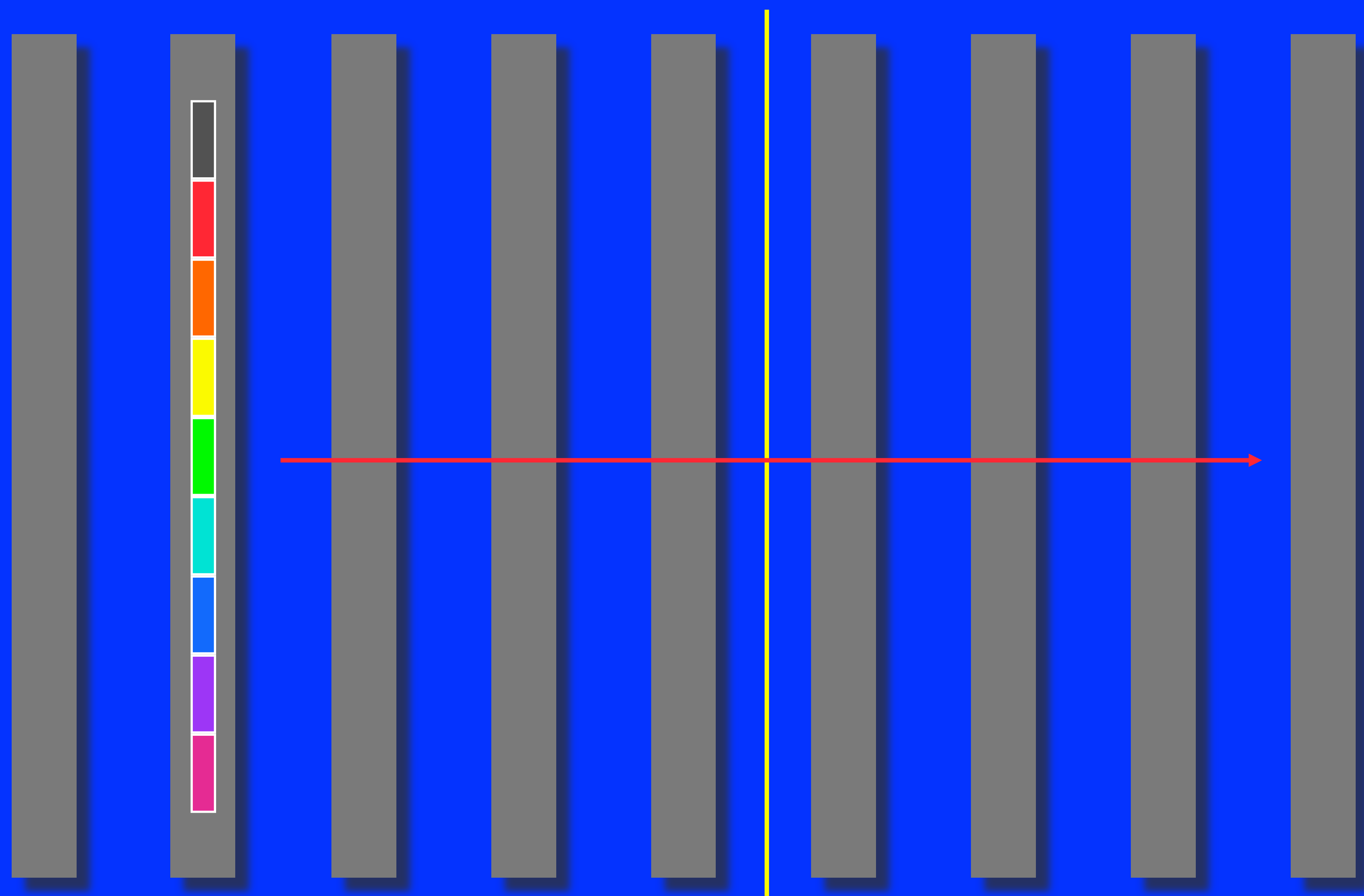


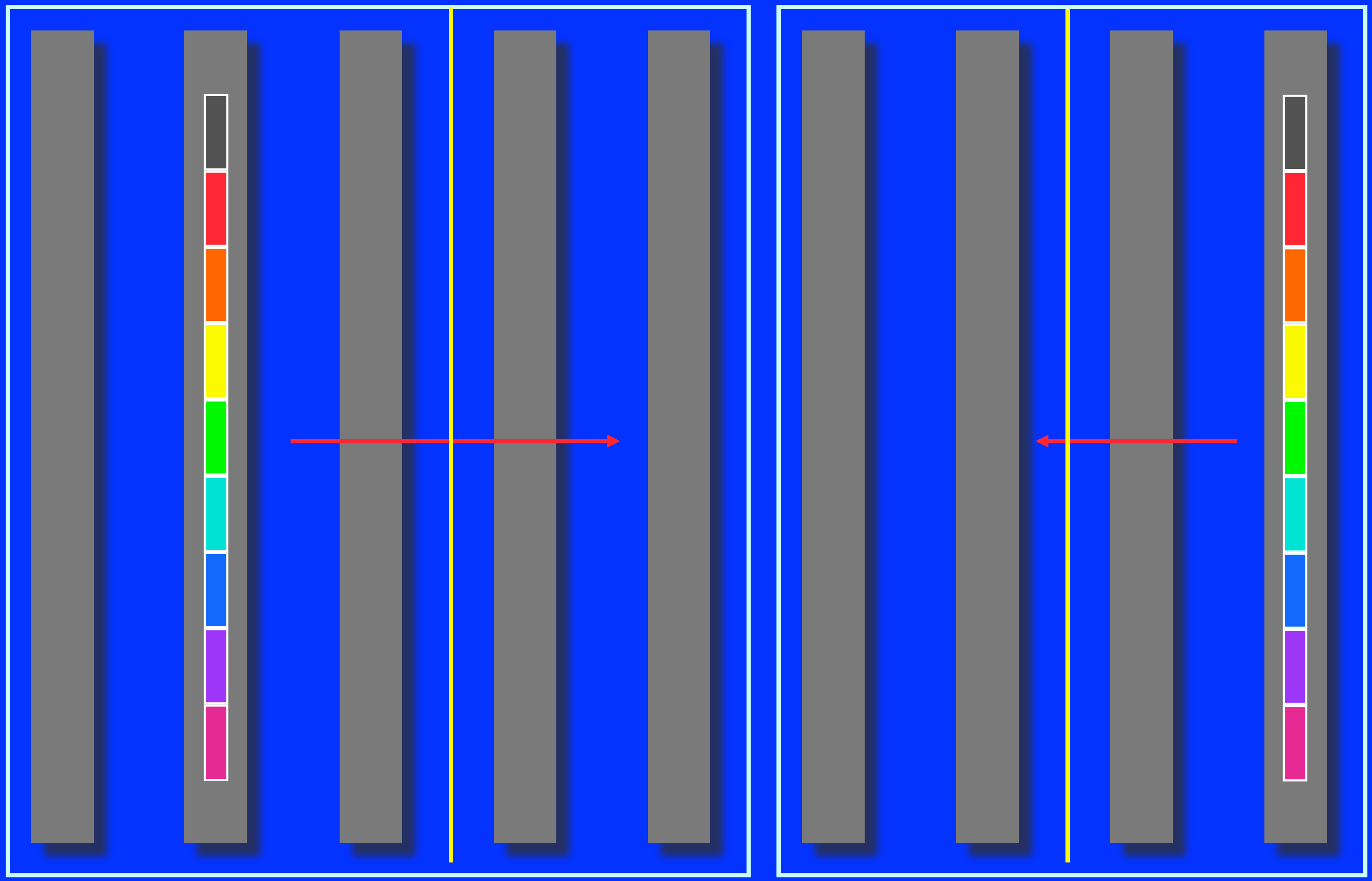
- continue recursively in each of the two halves

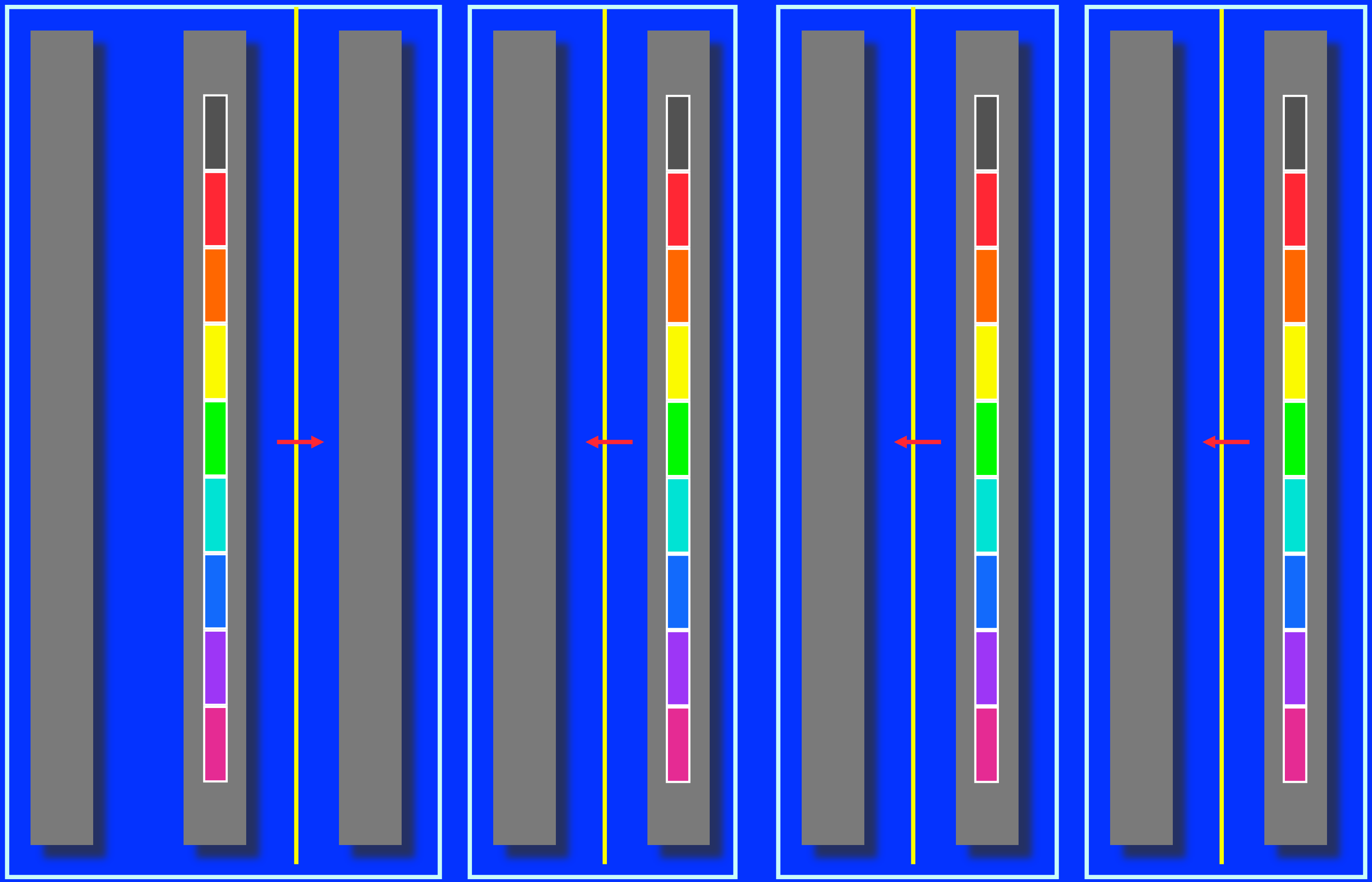
Broadcast

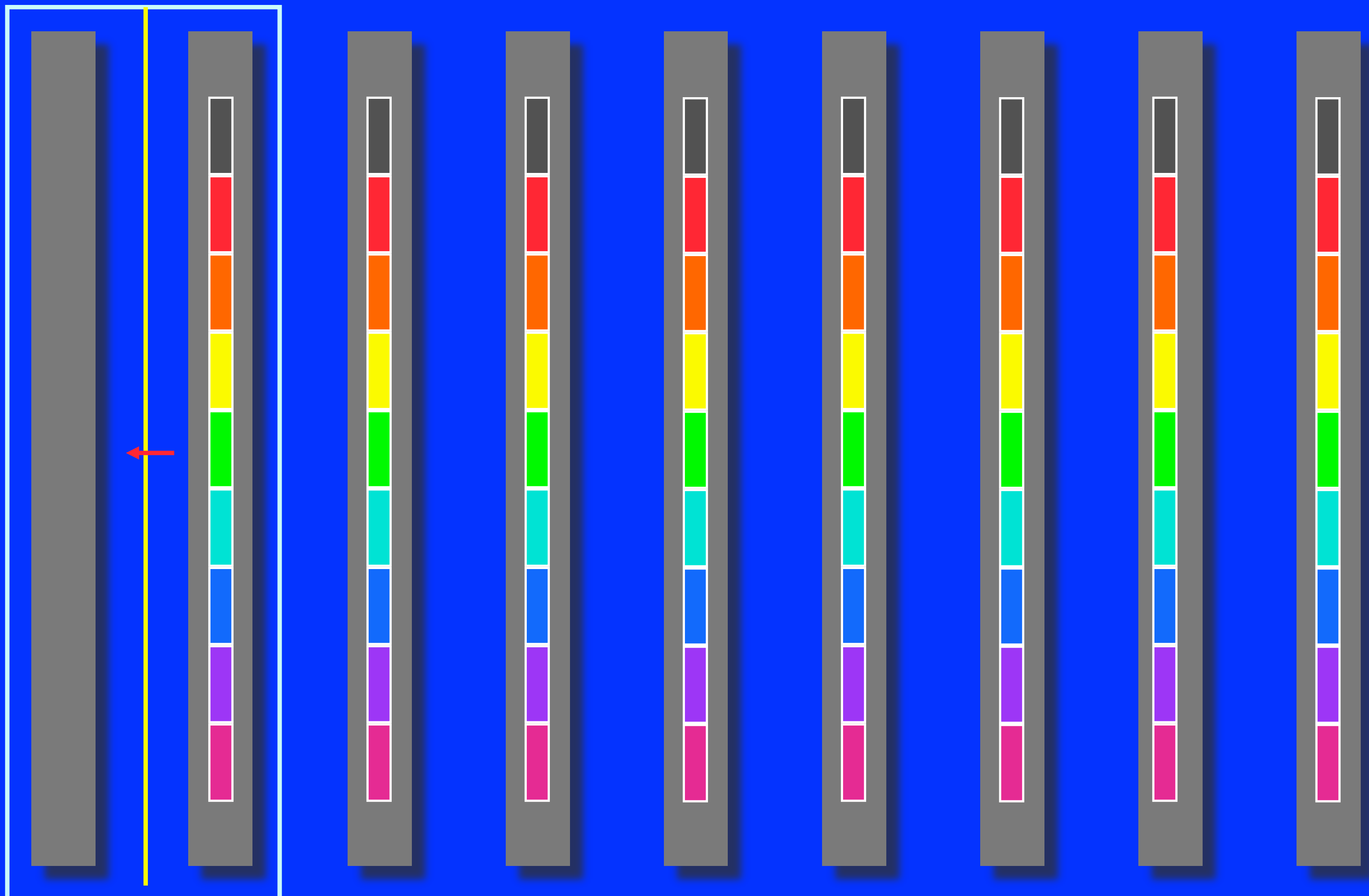




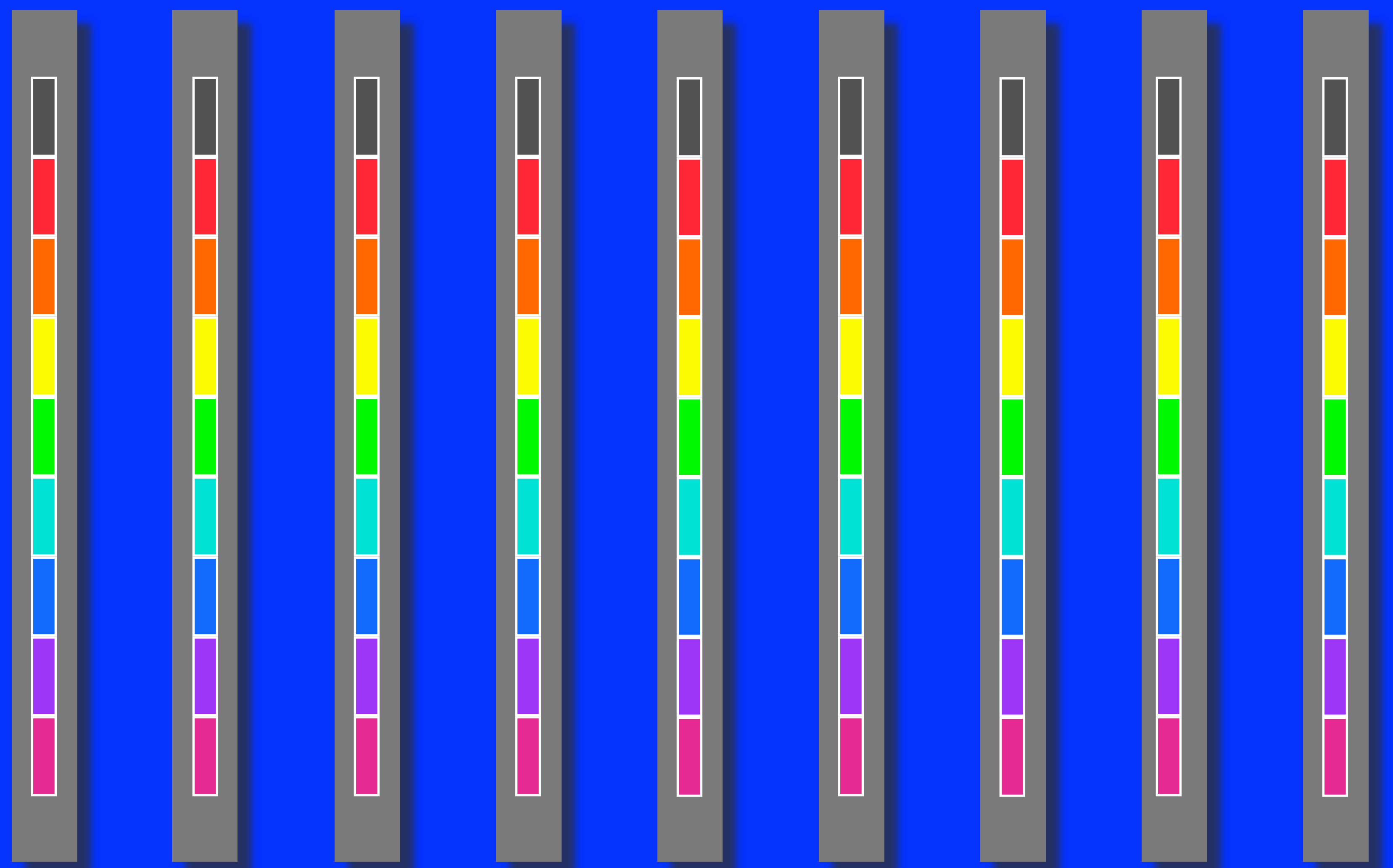








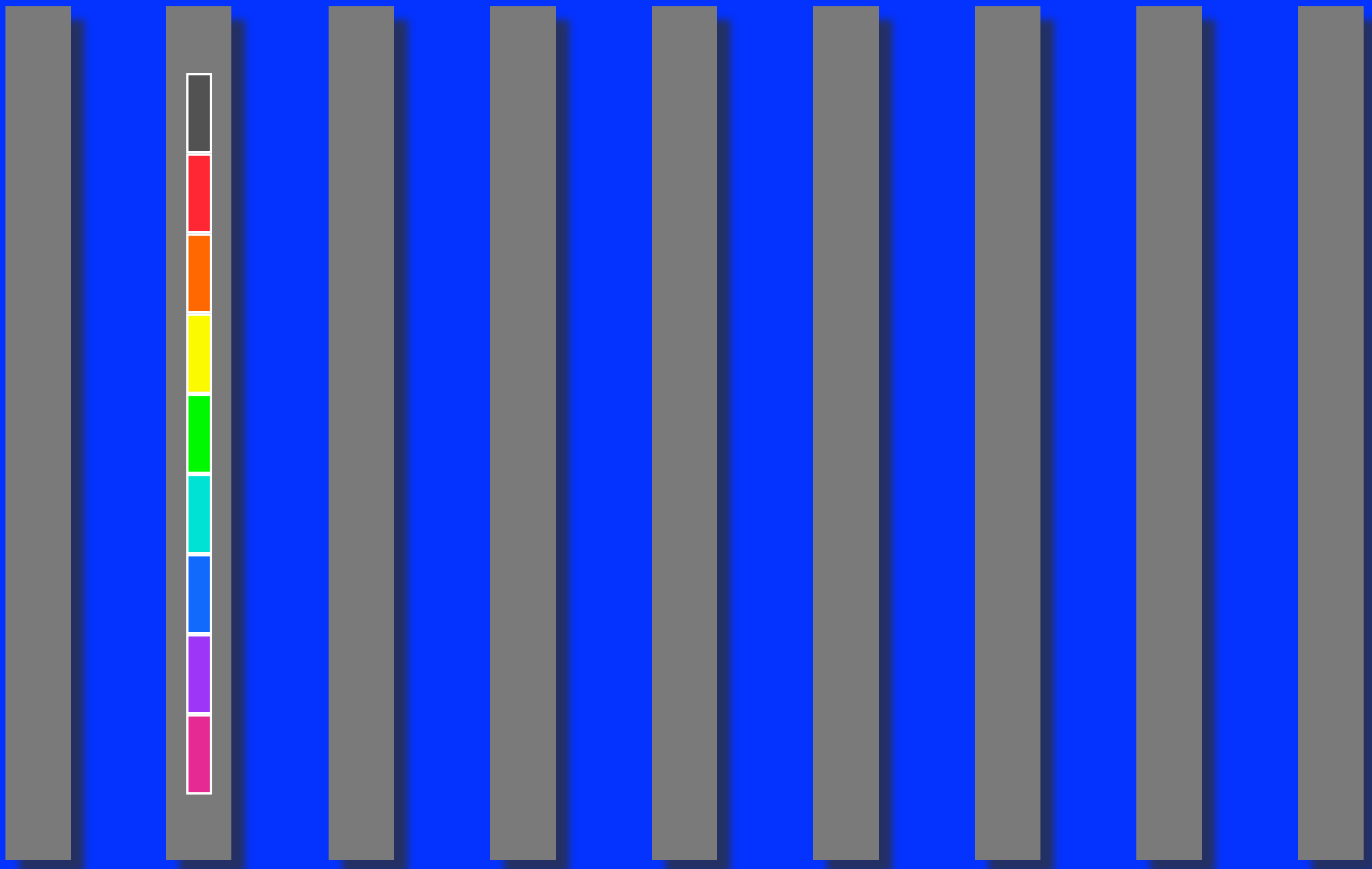
121

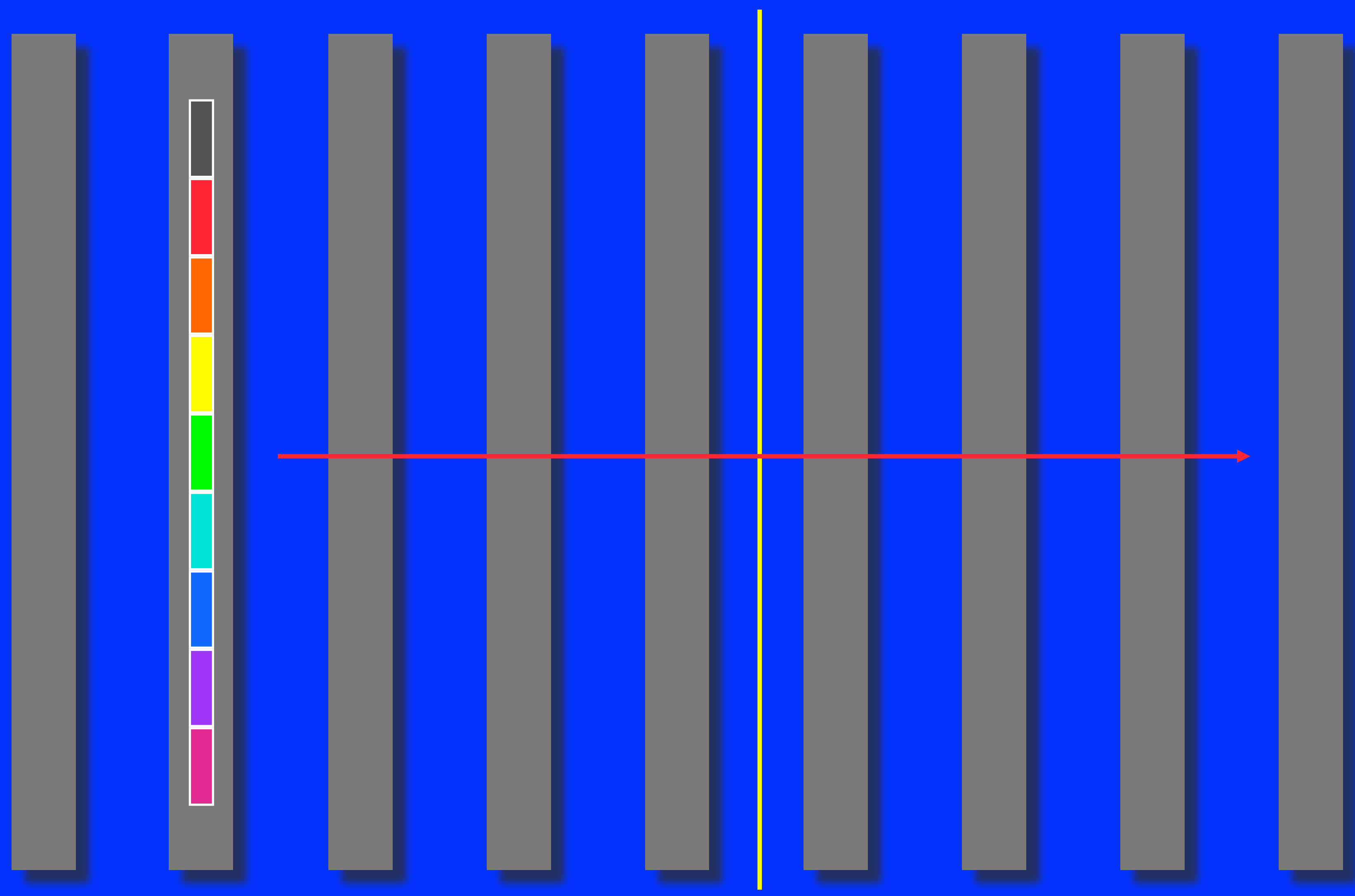


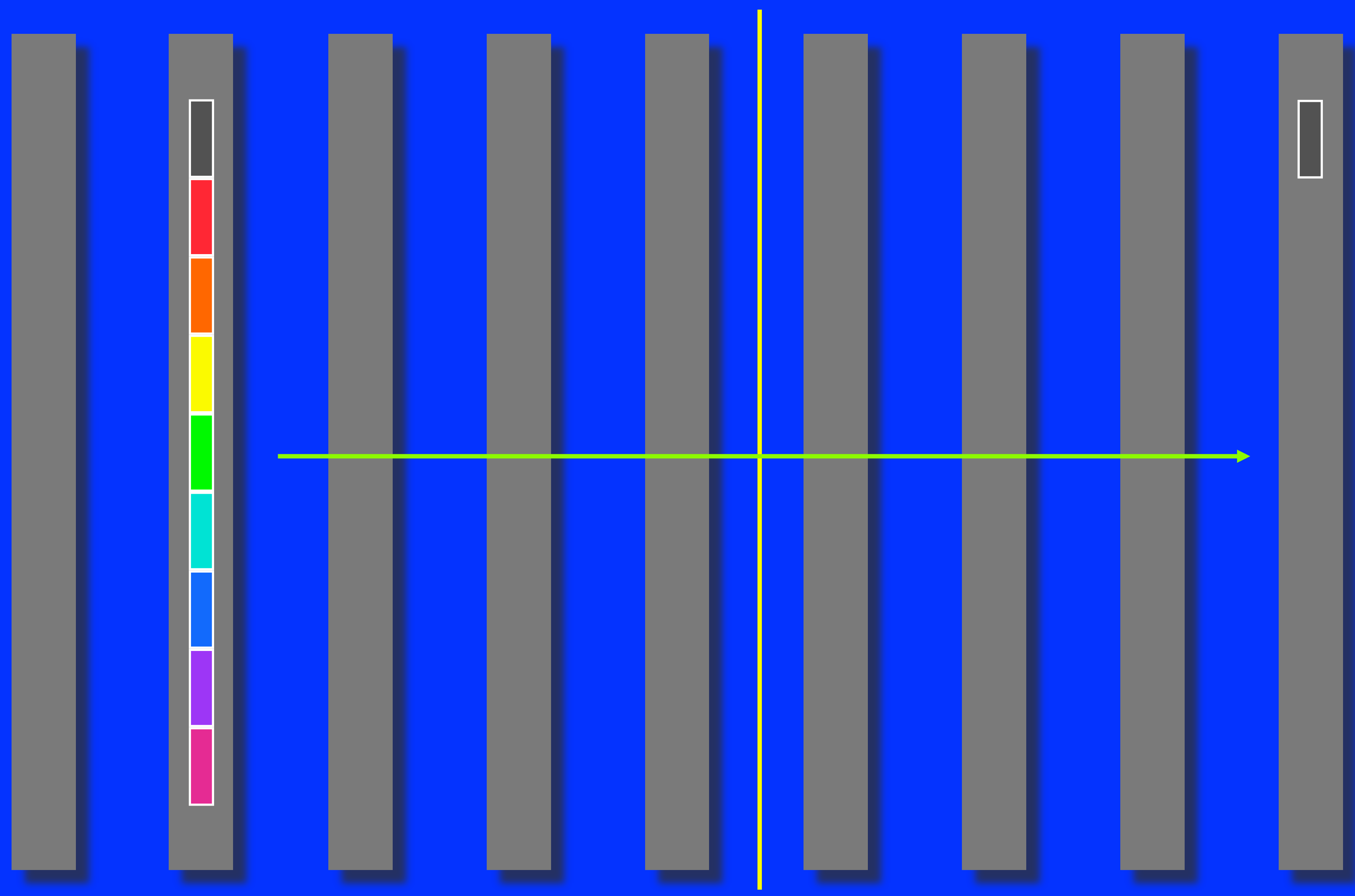
Let us view this more closely

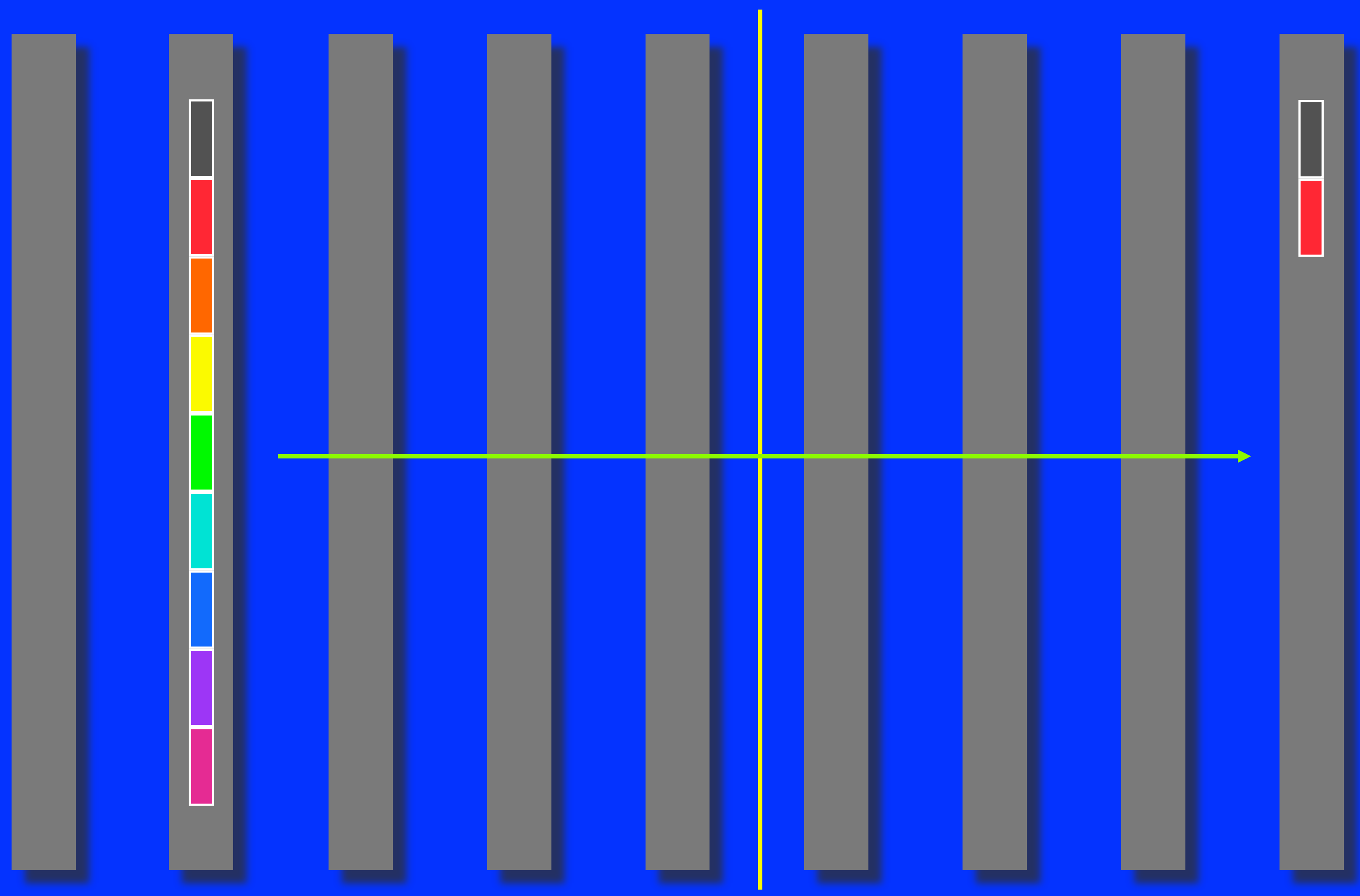
- Red arrows indicate startup of communication (leading to latency, α)

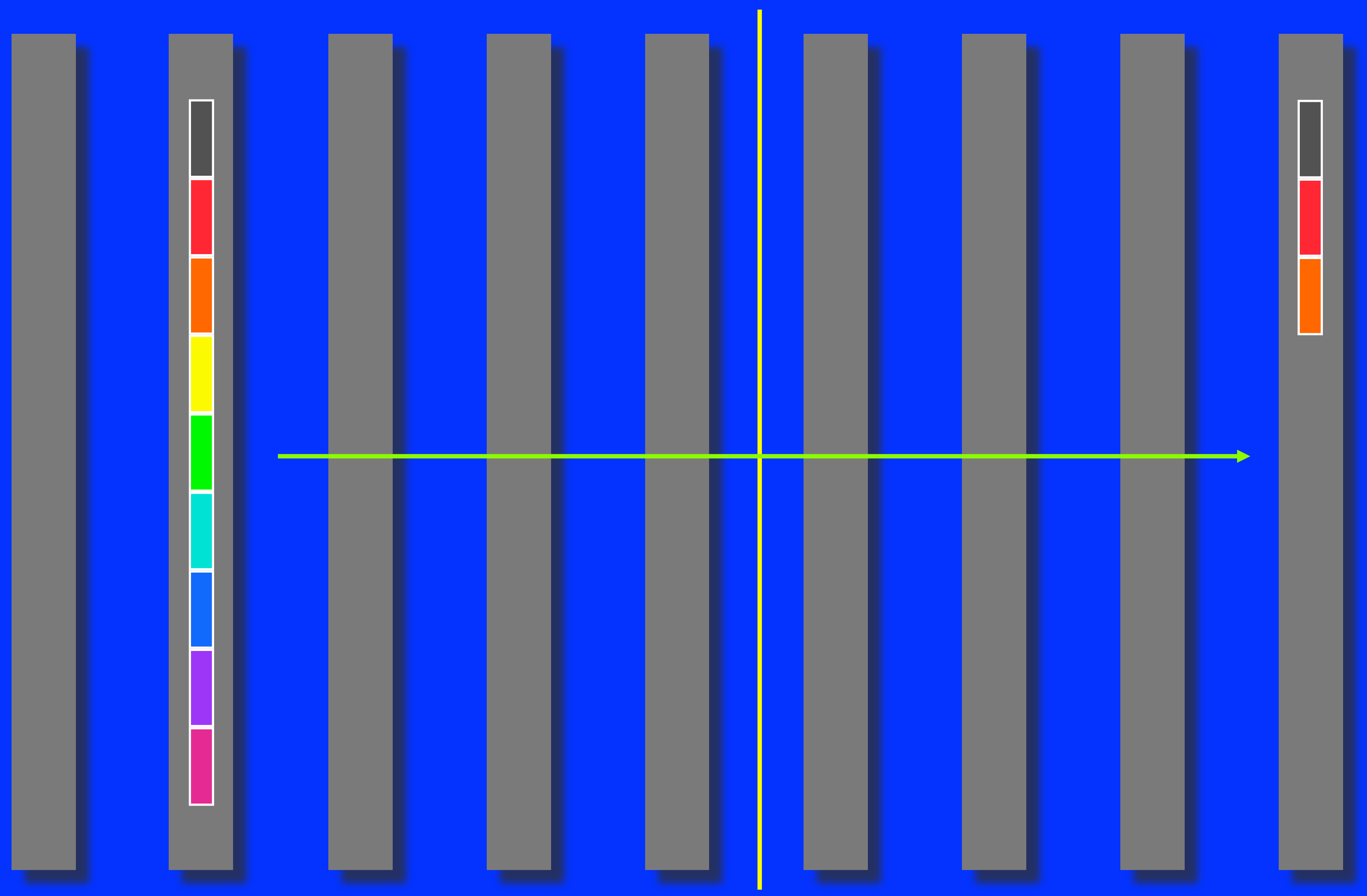
- Green arrows indicate packets in transit (leading to a bandwidth related cost proportional to β and the length of the packet)

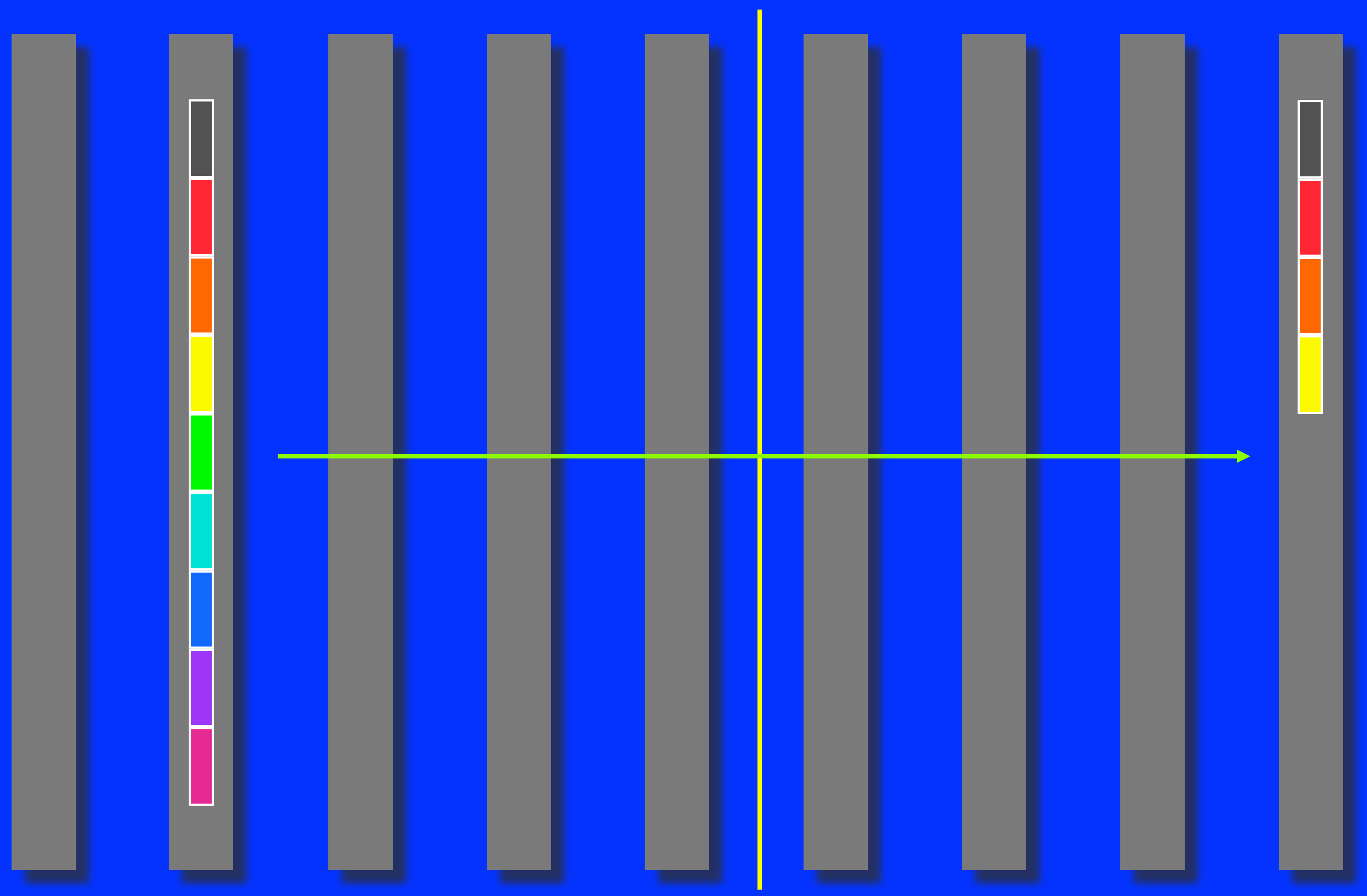


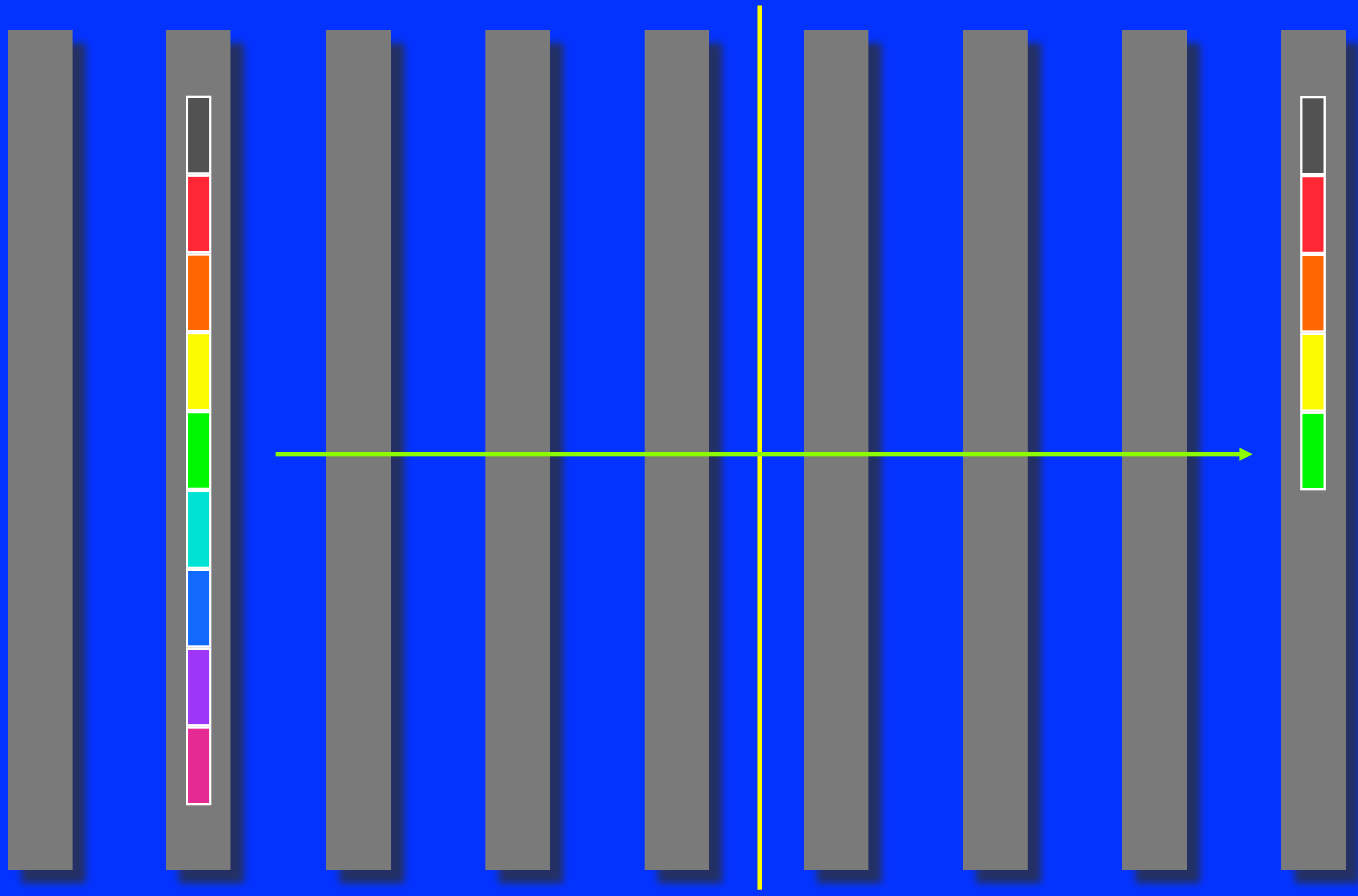


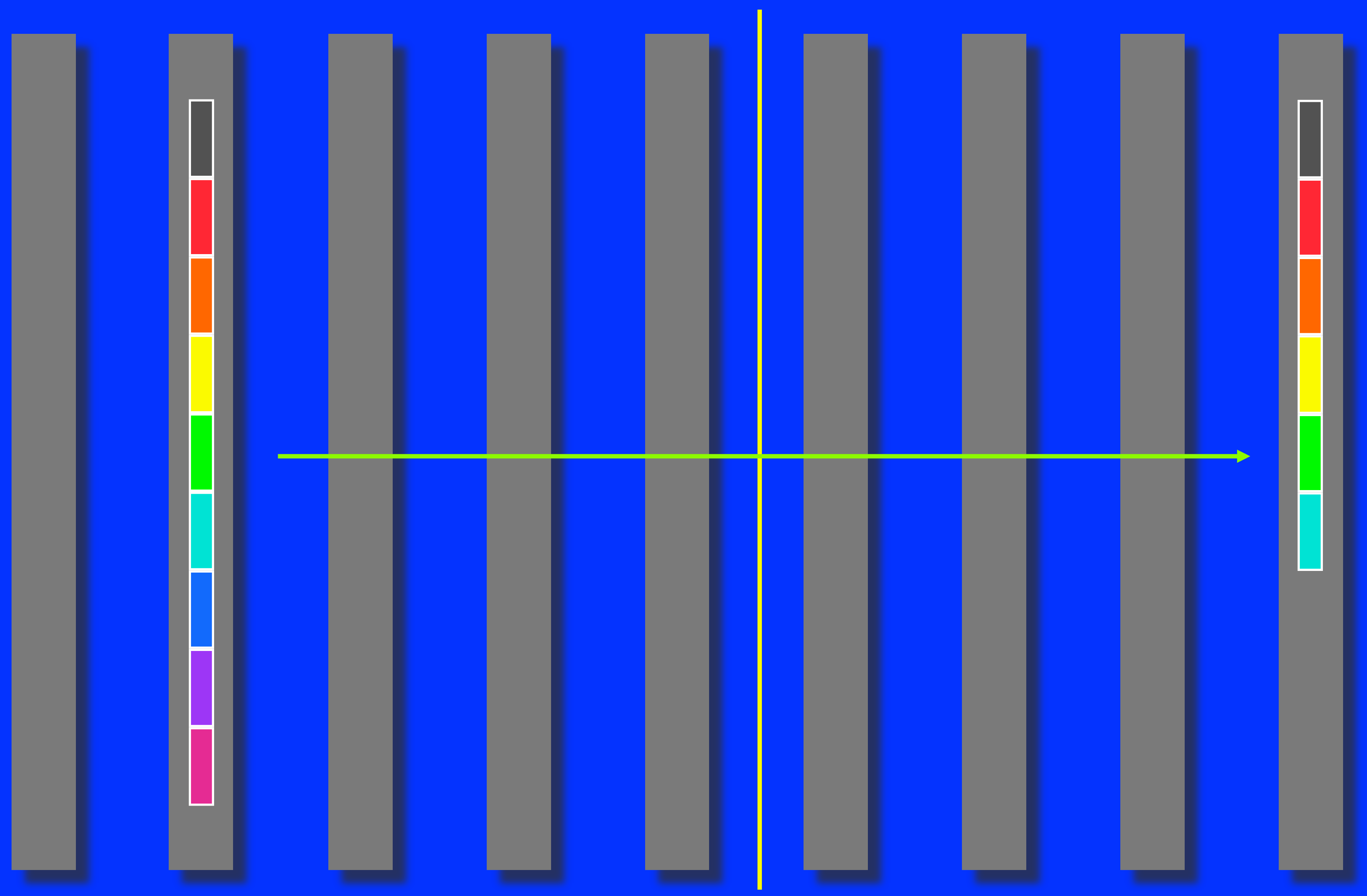


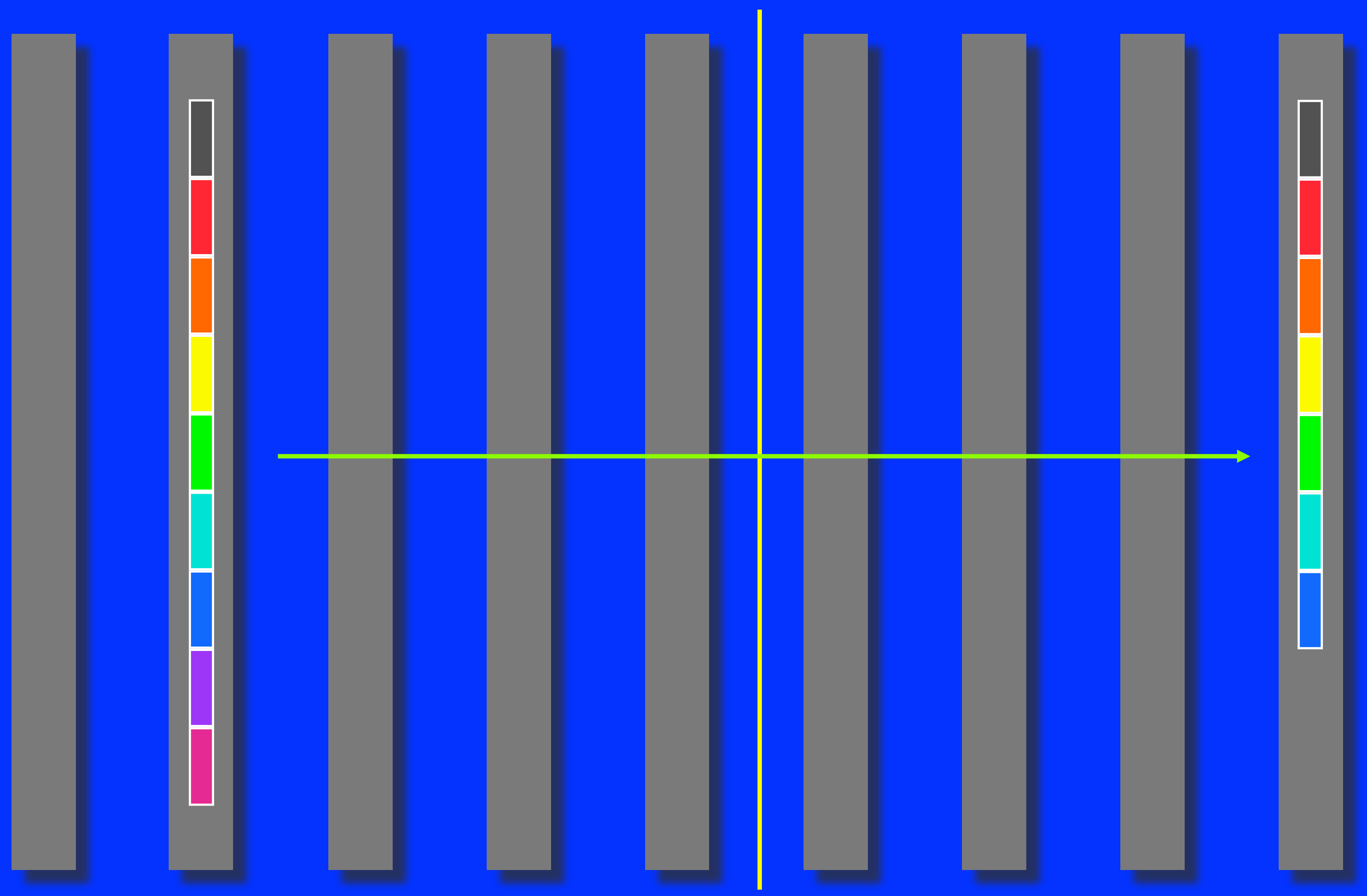




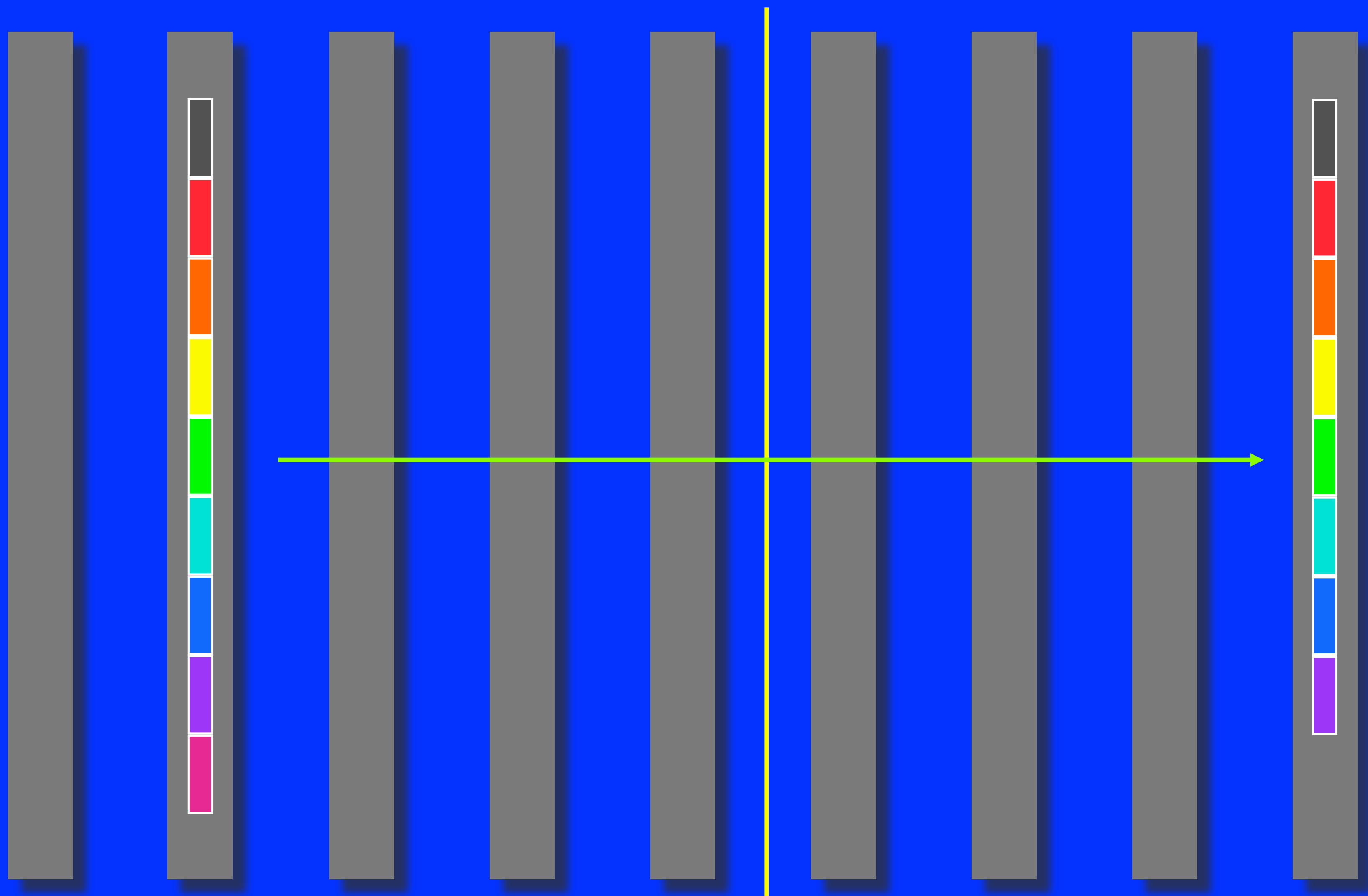


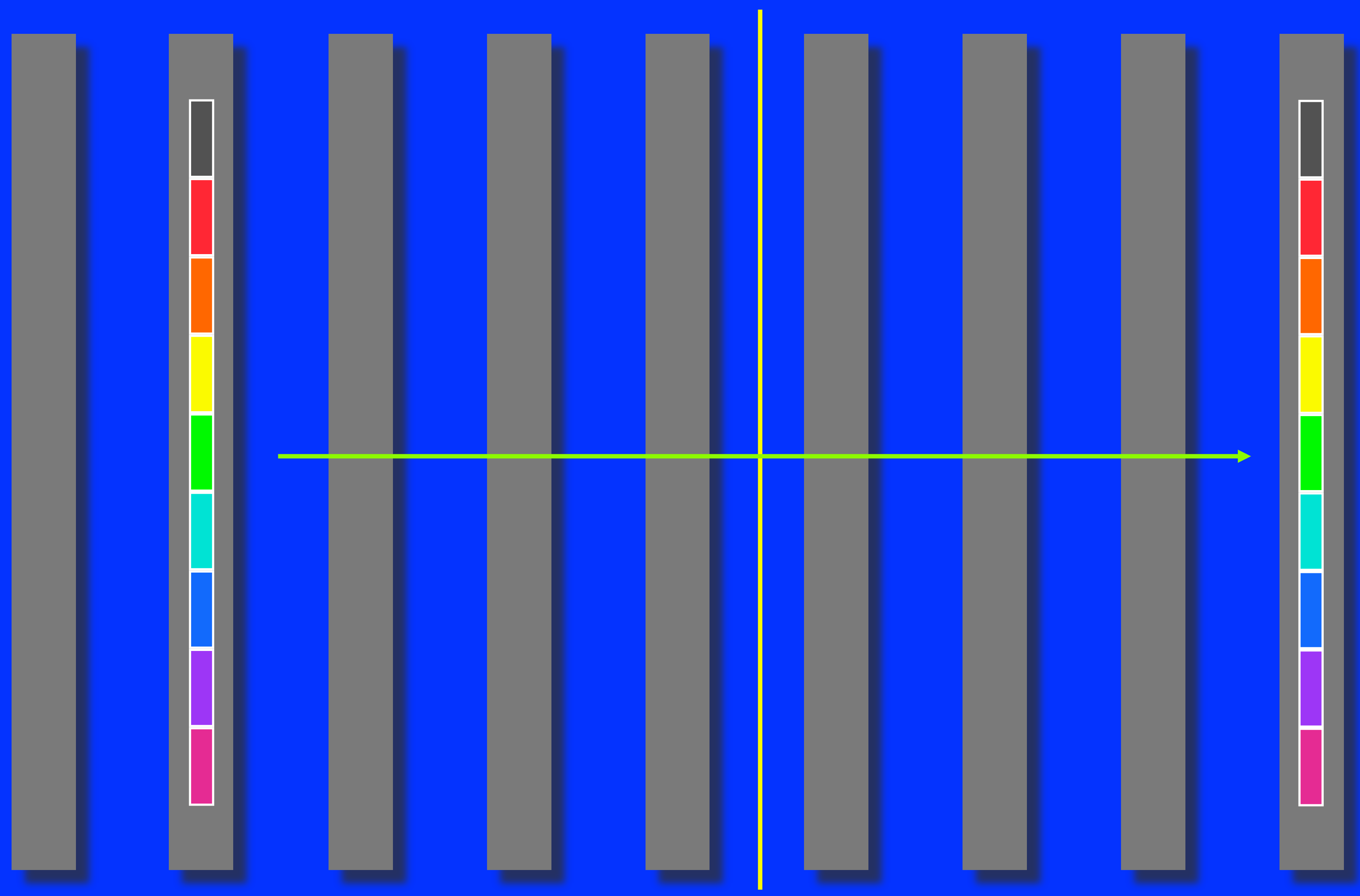


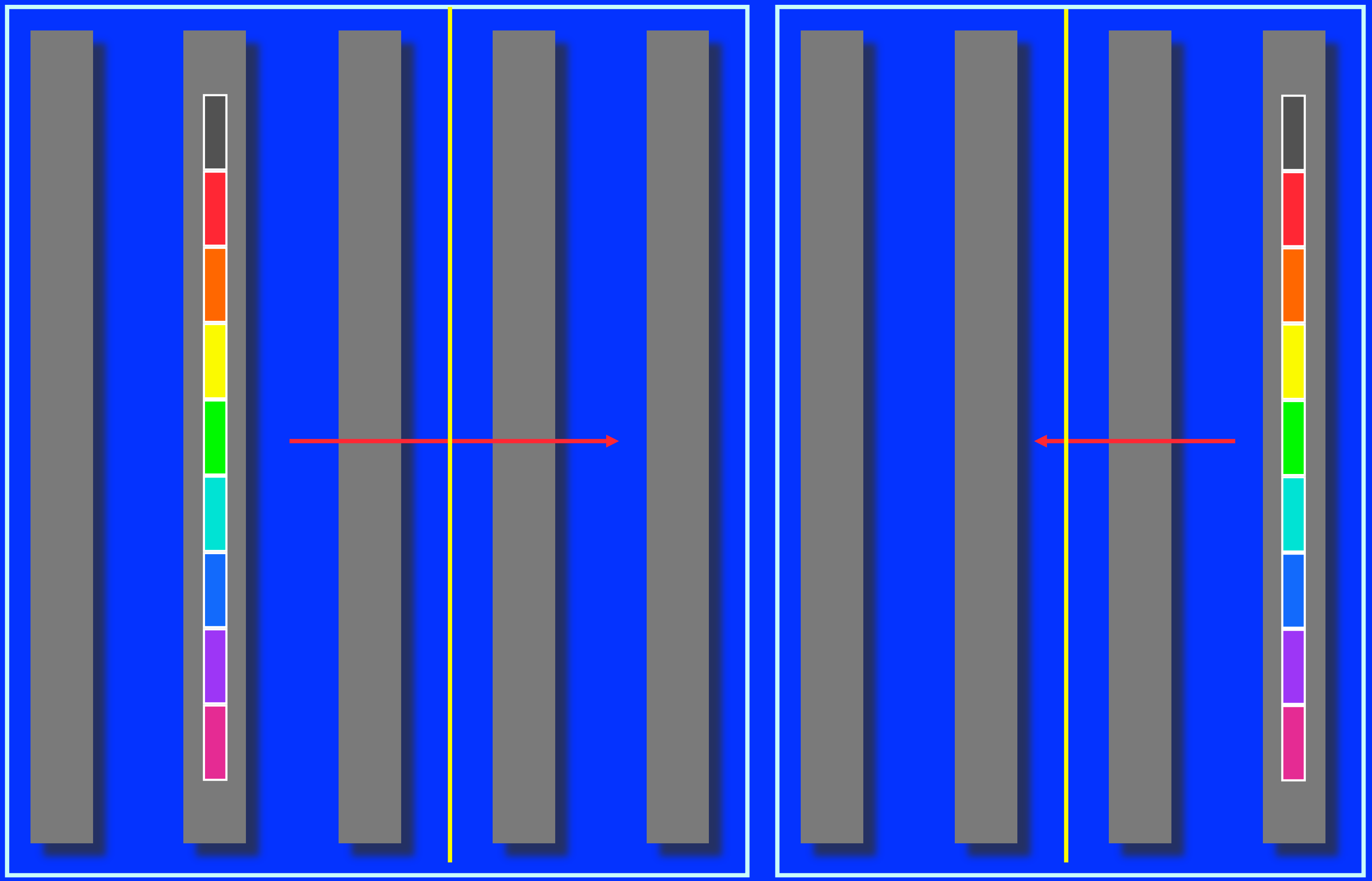


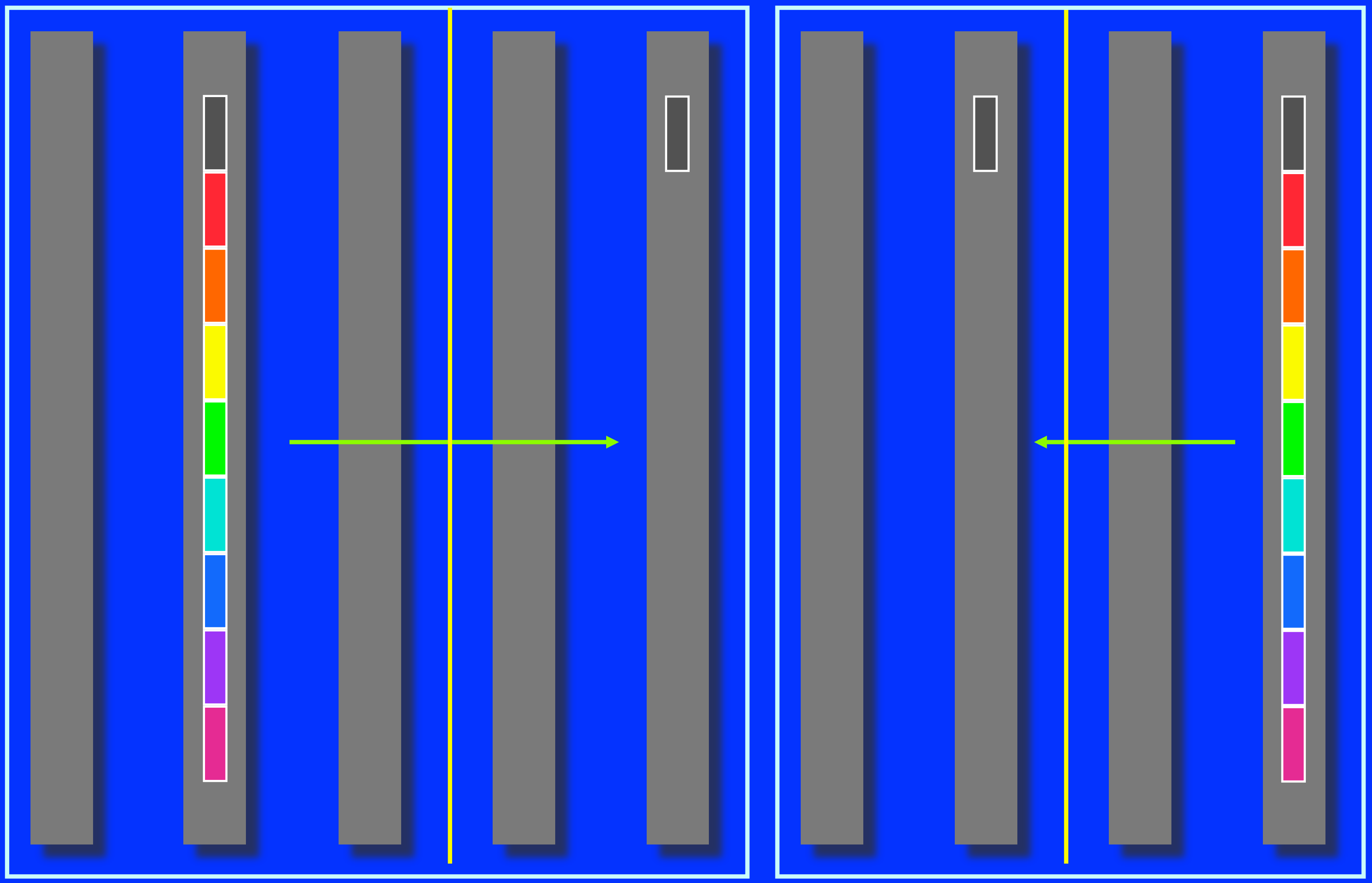


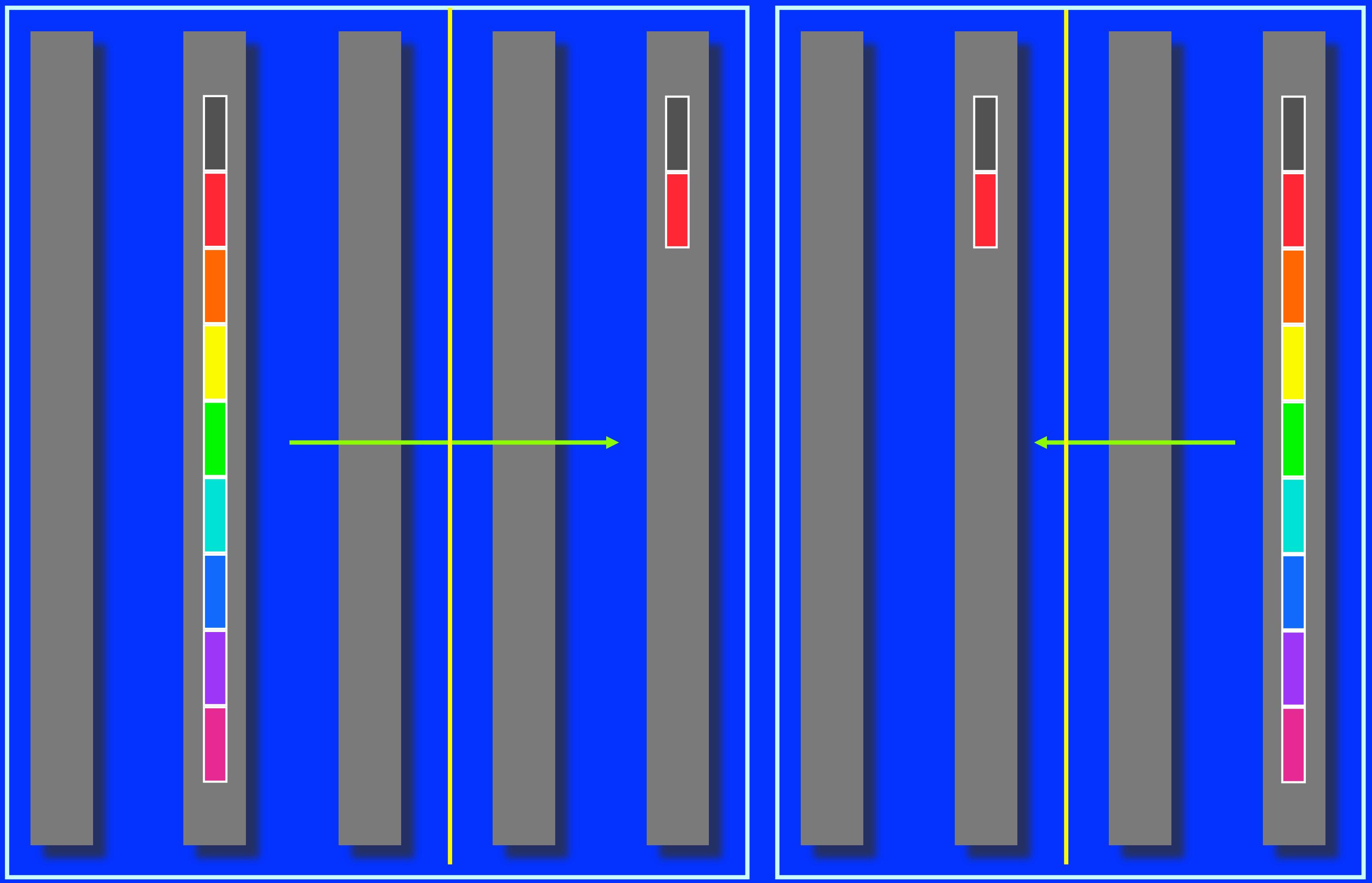
132

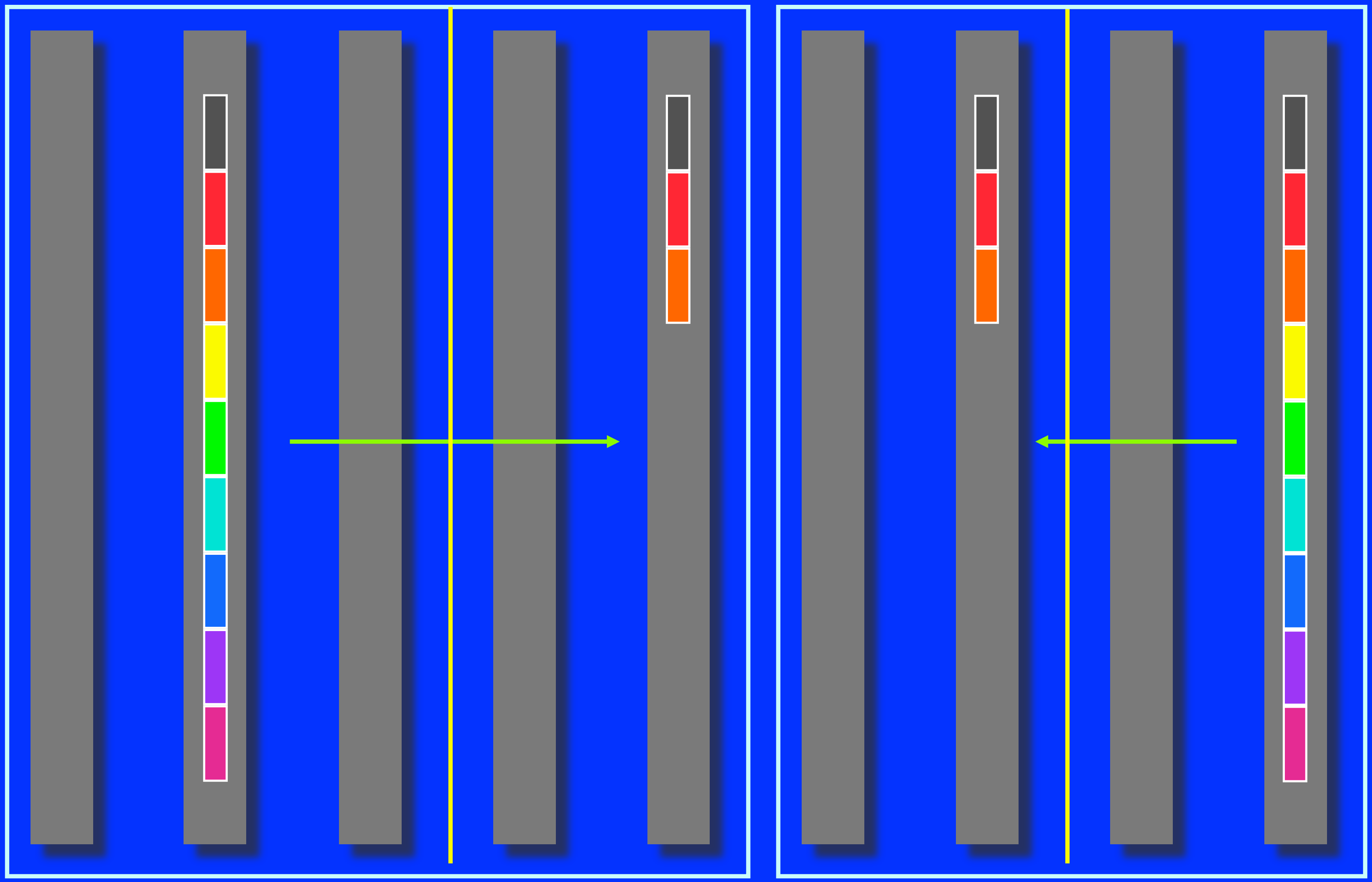


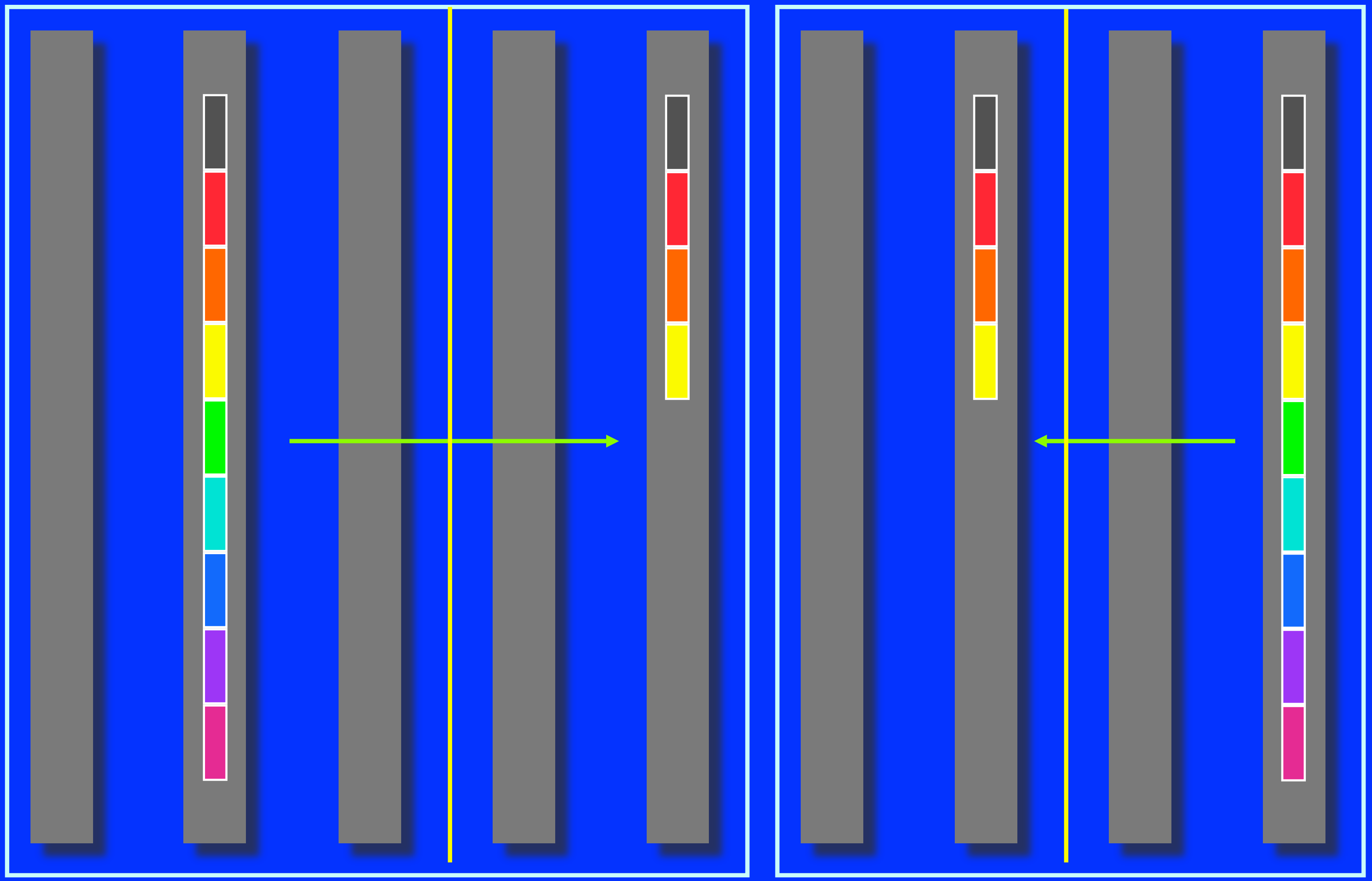


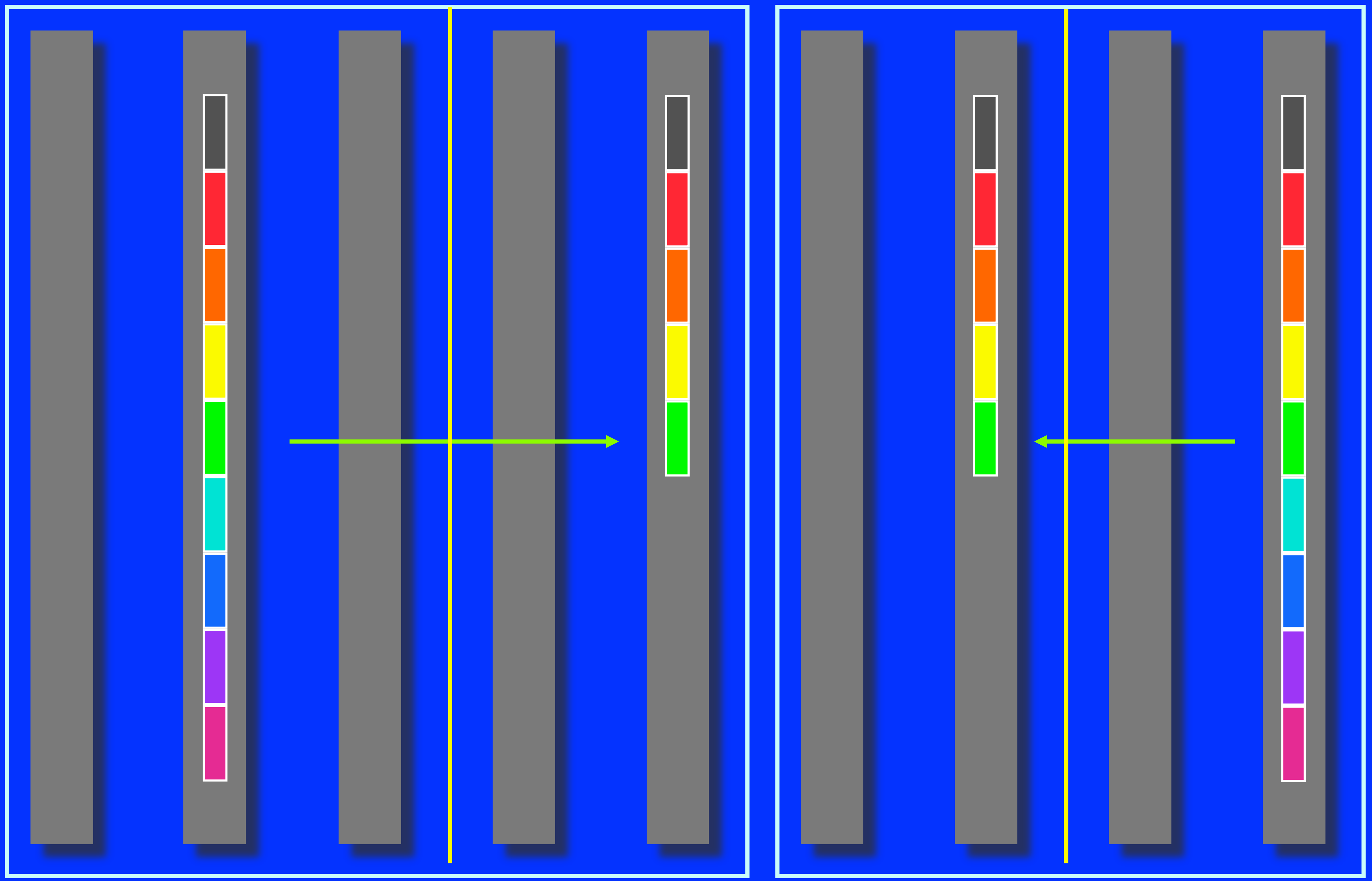


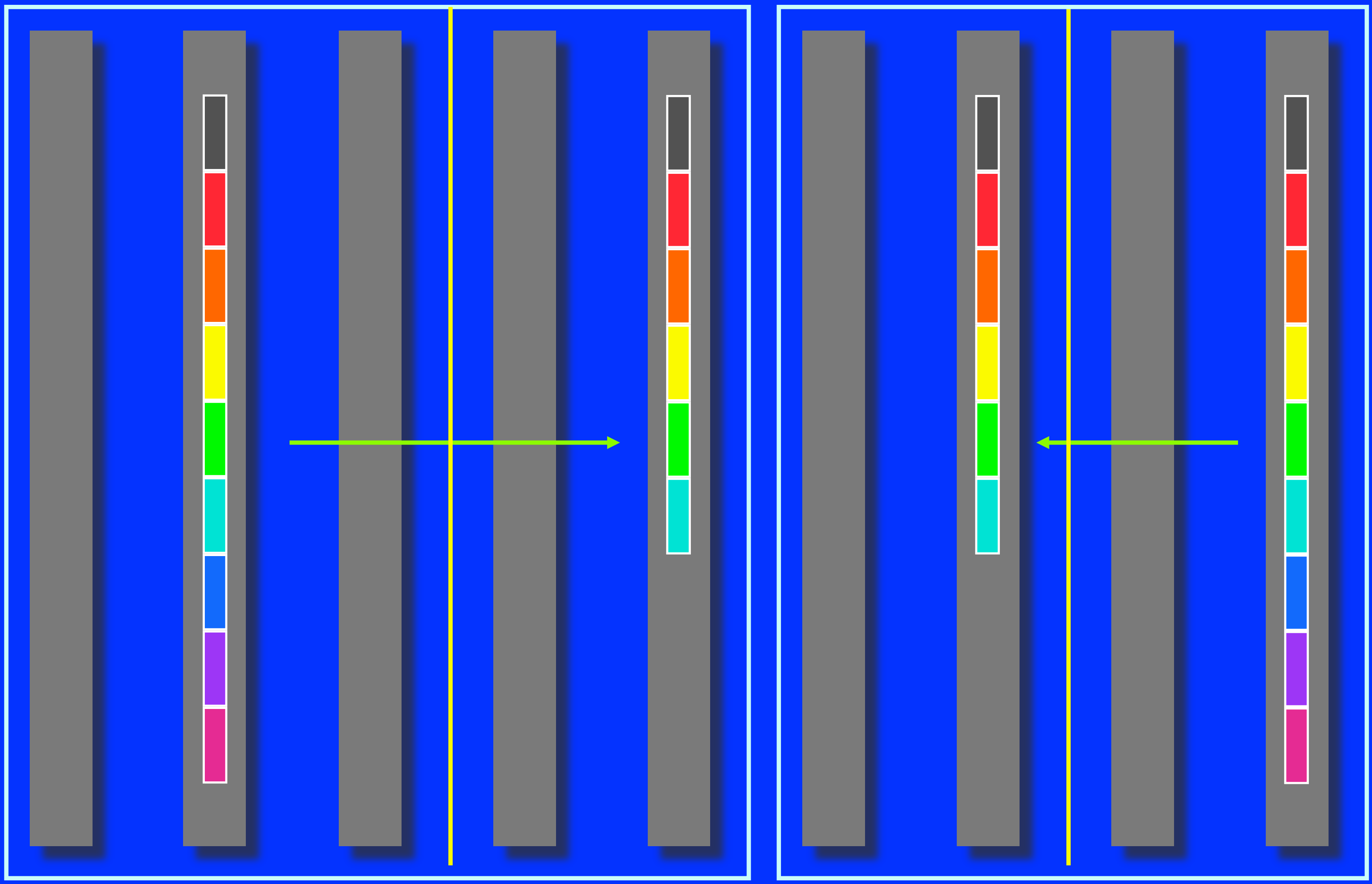


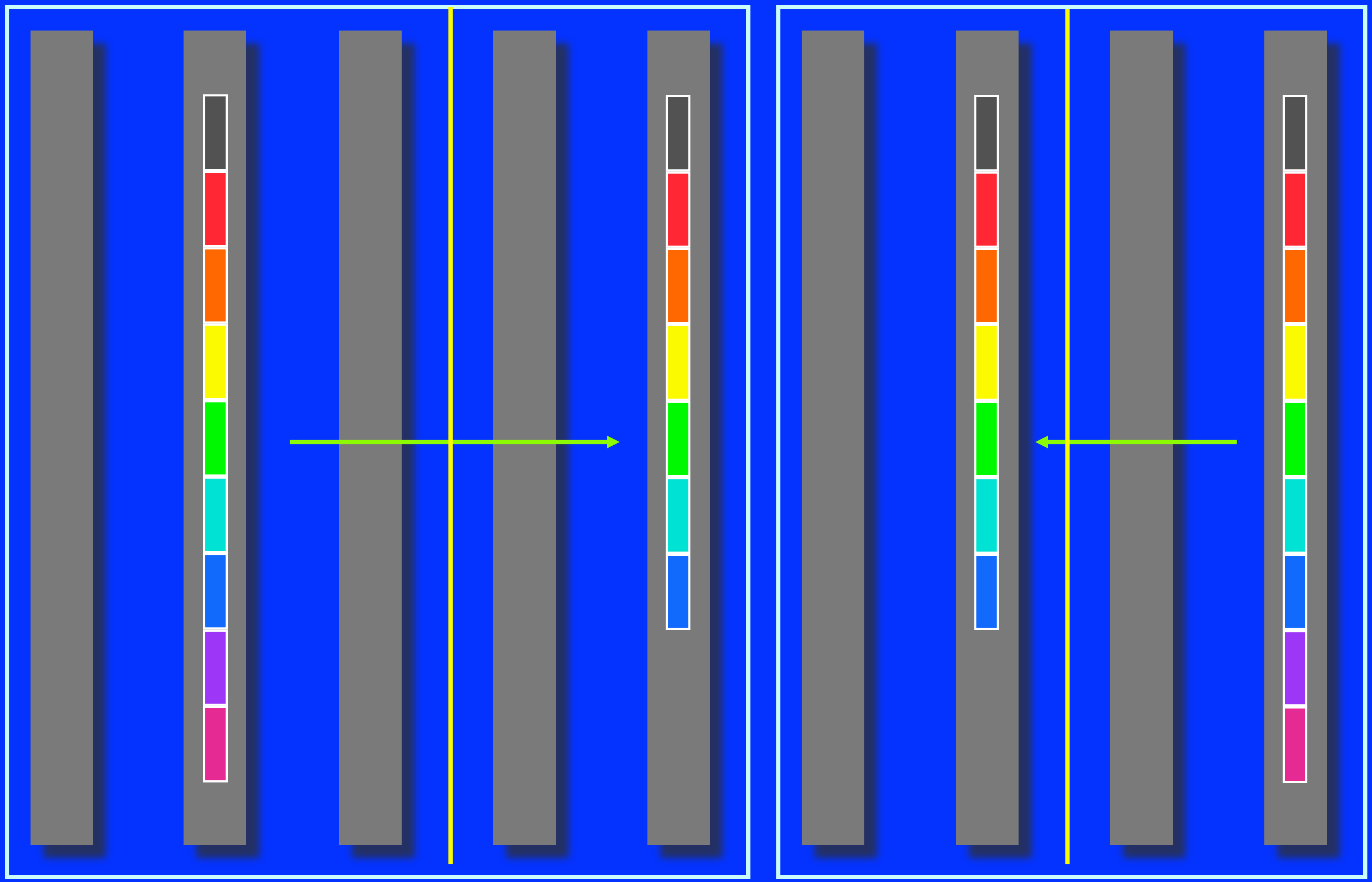


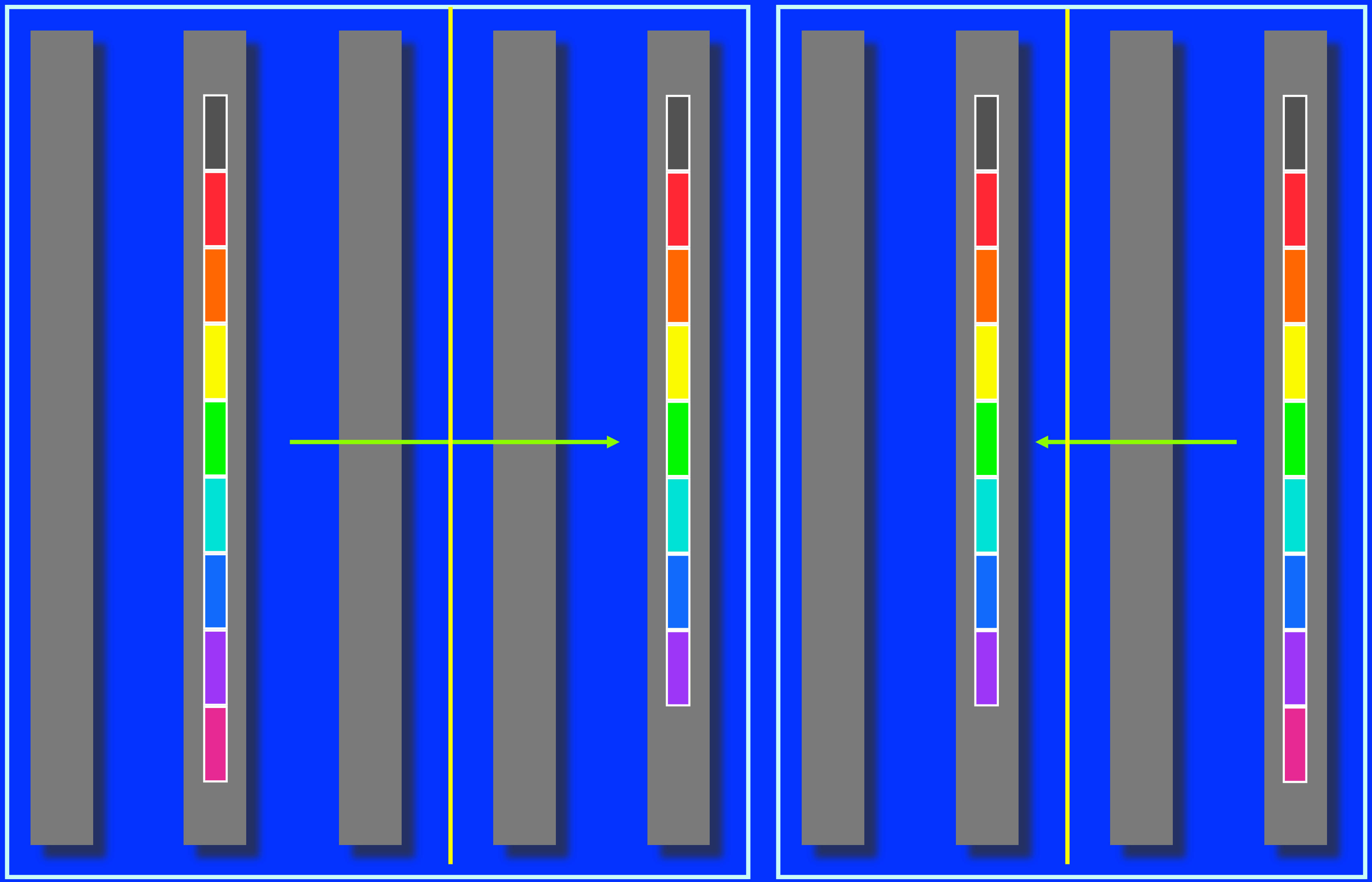


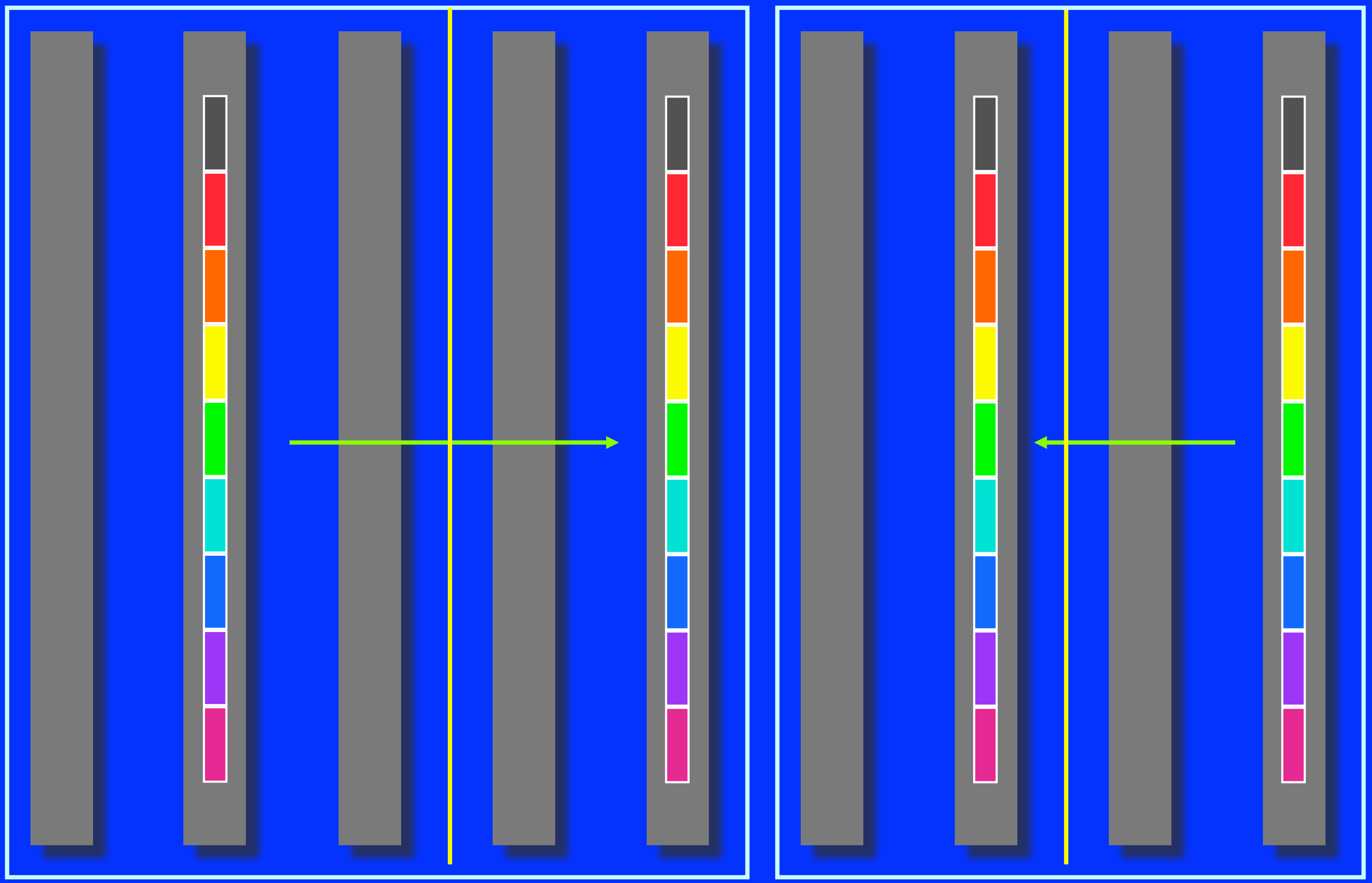


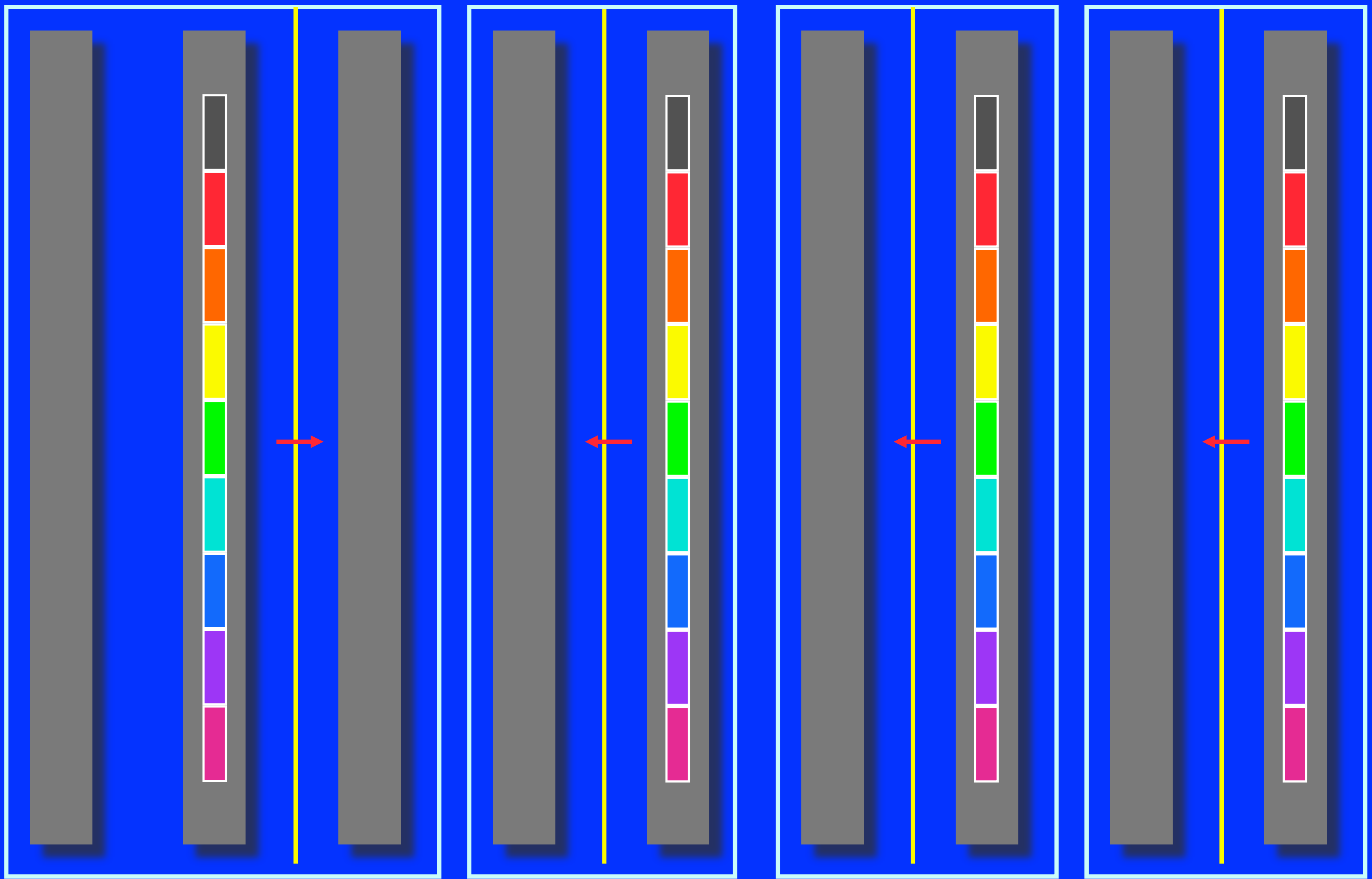


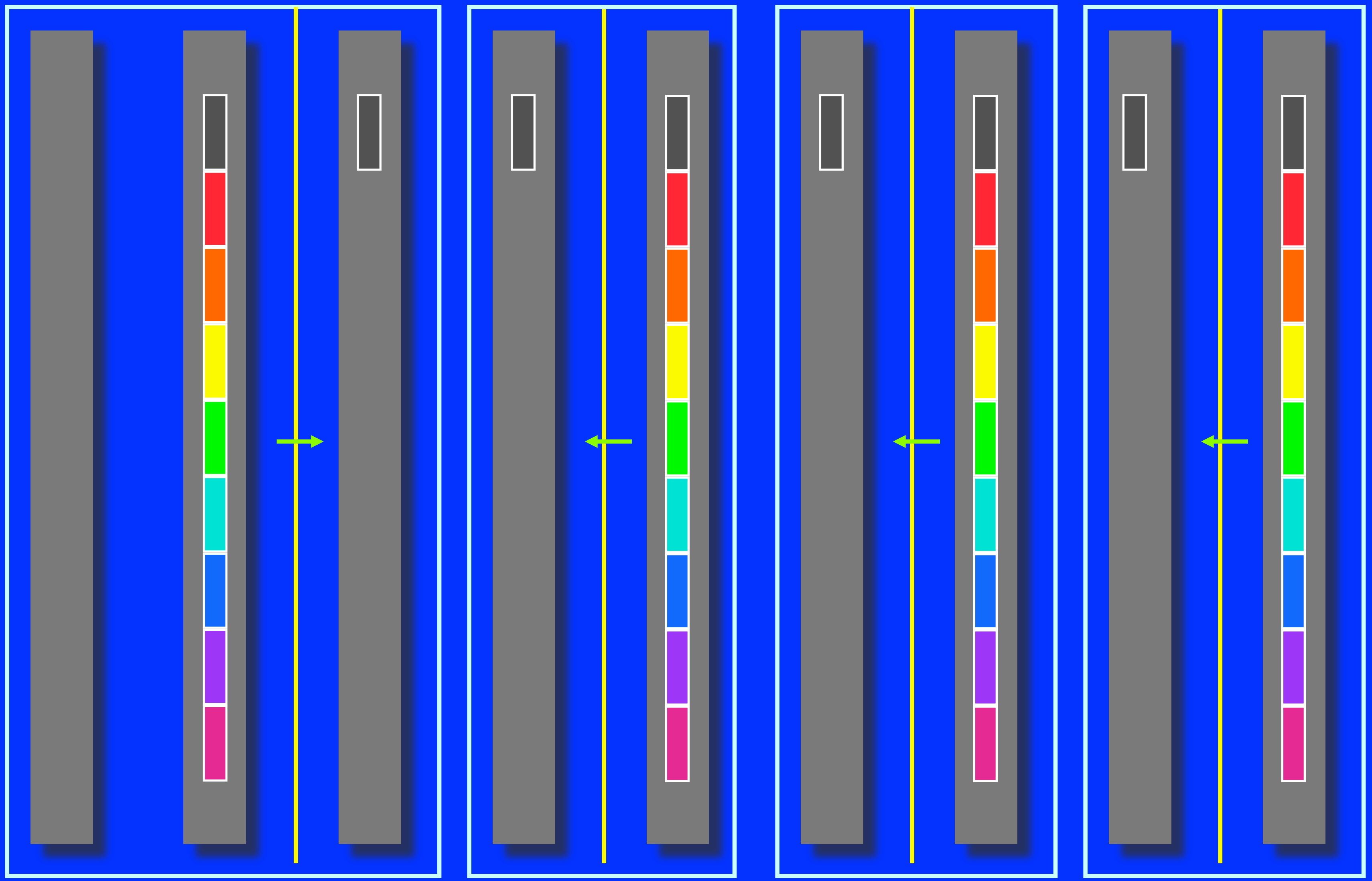


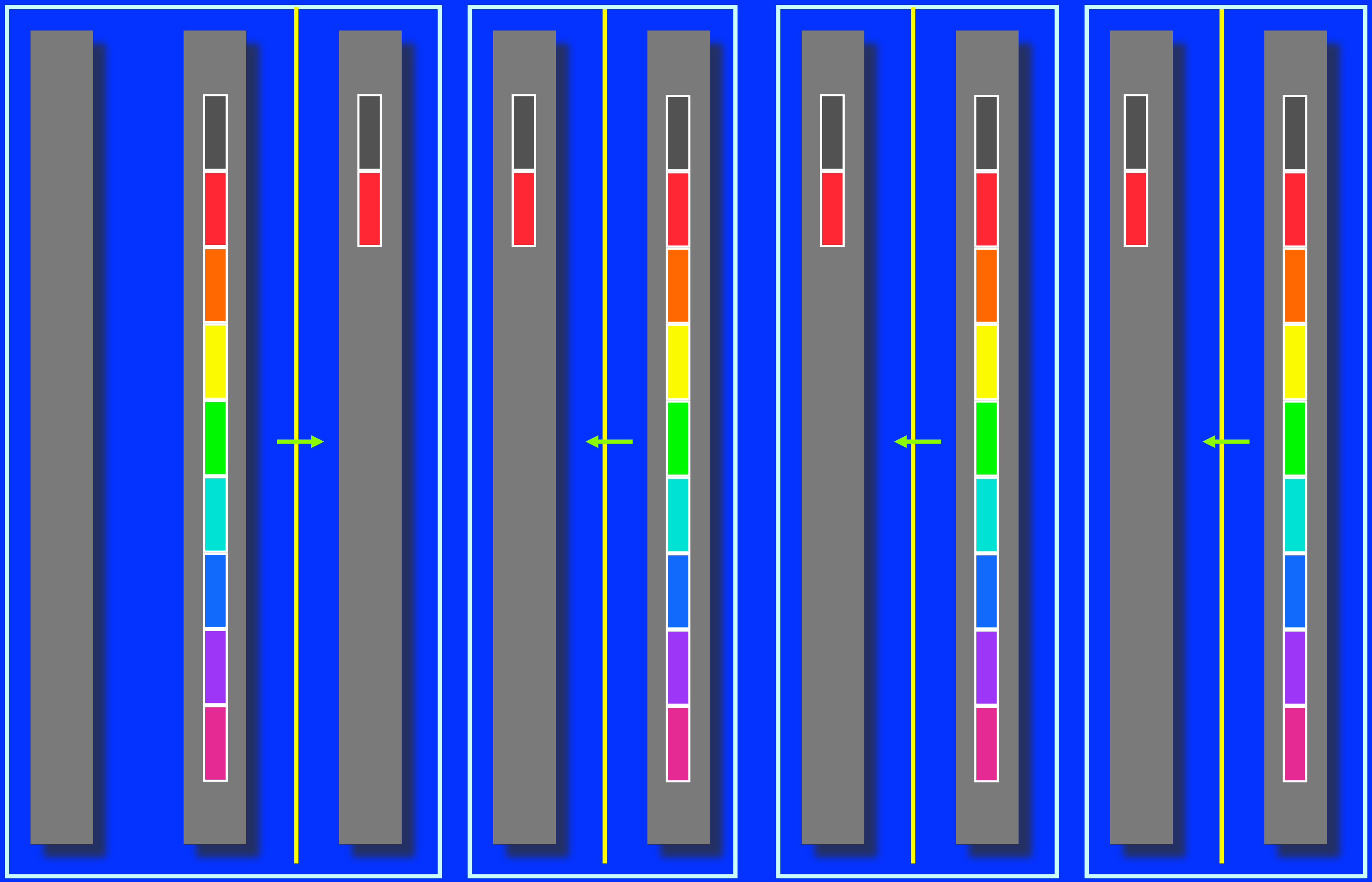


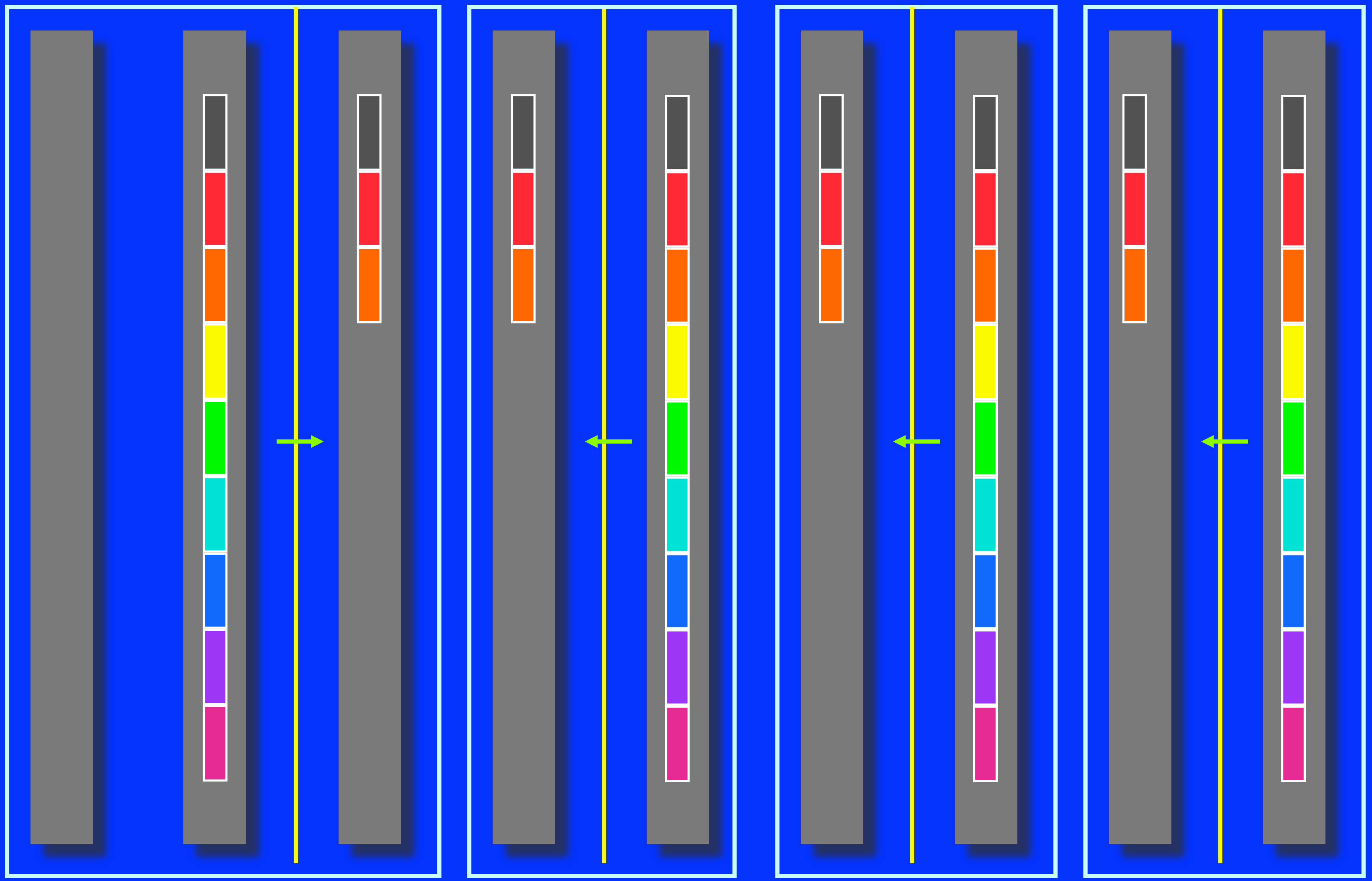


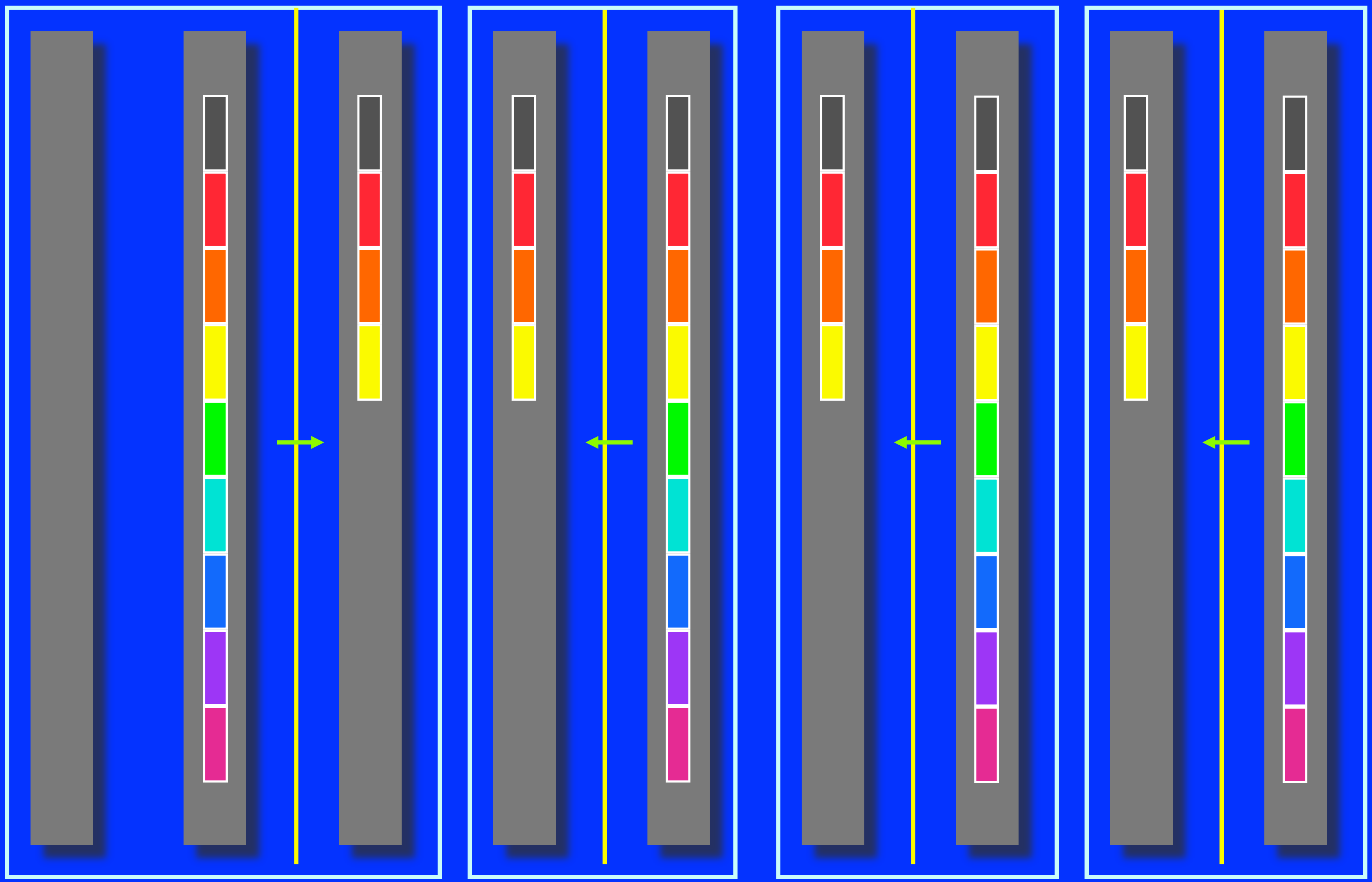


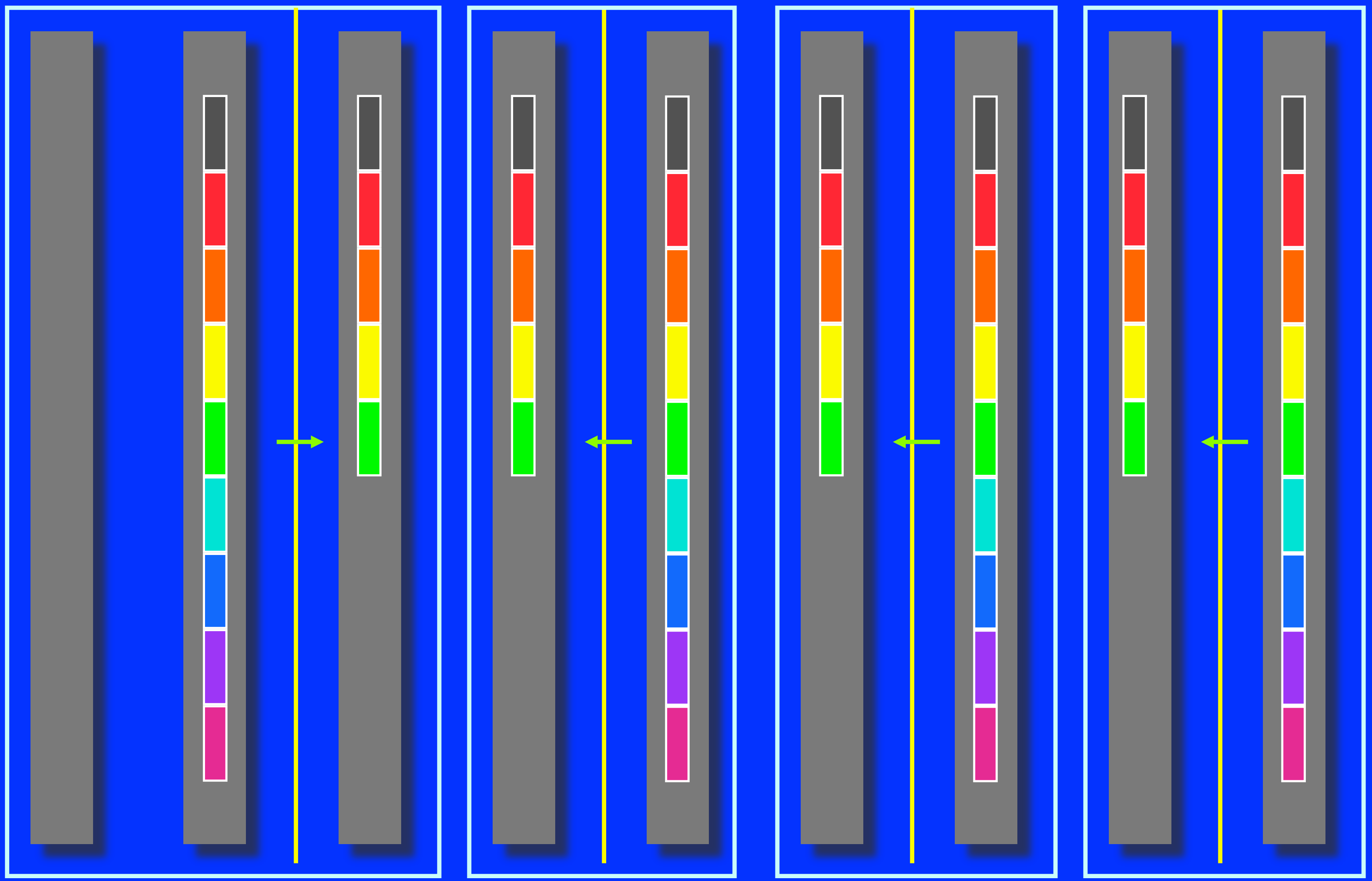


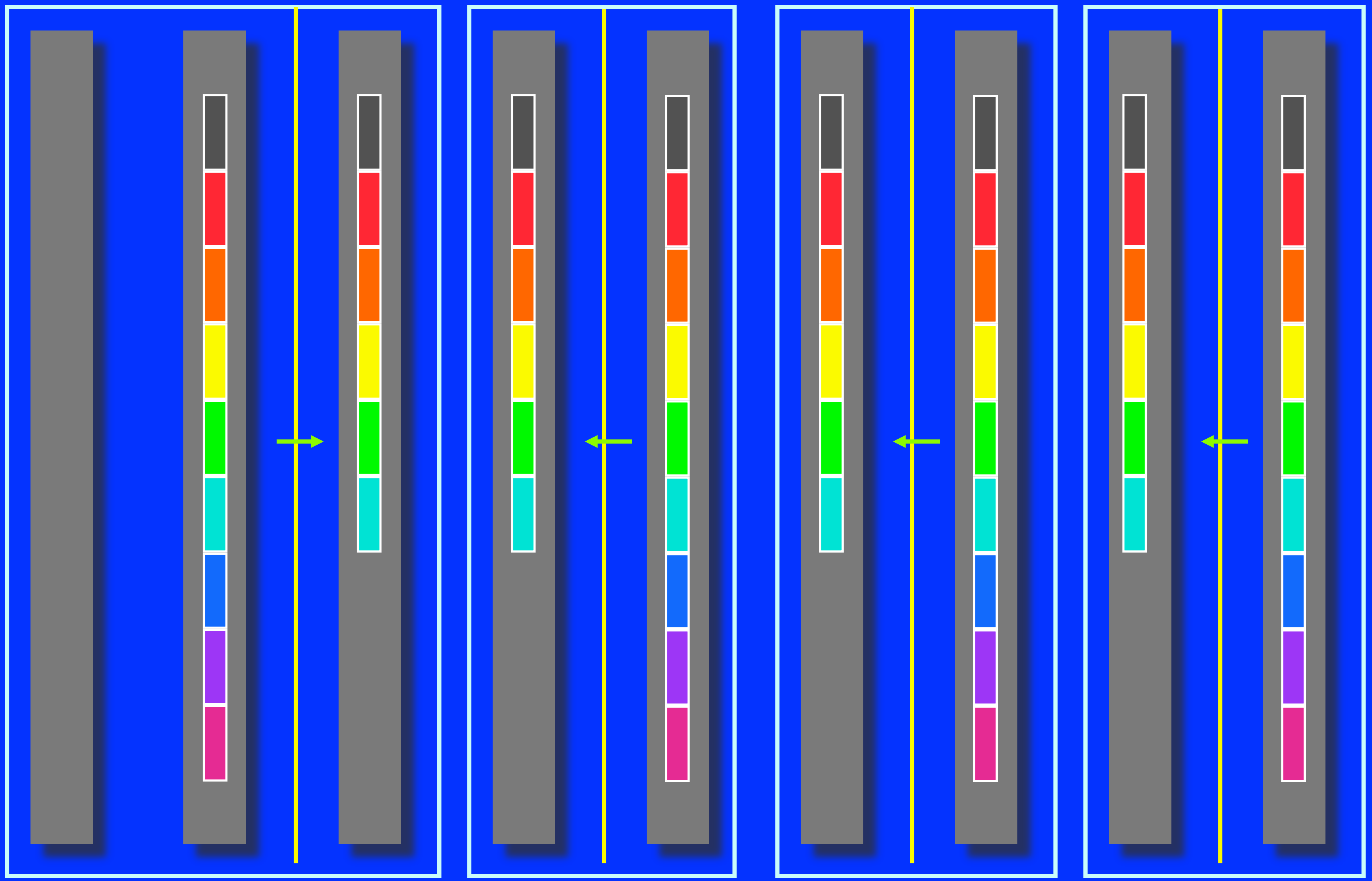


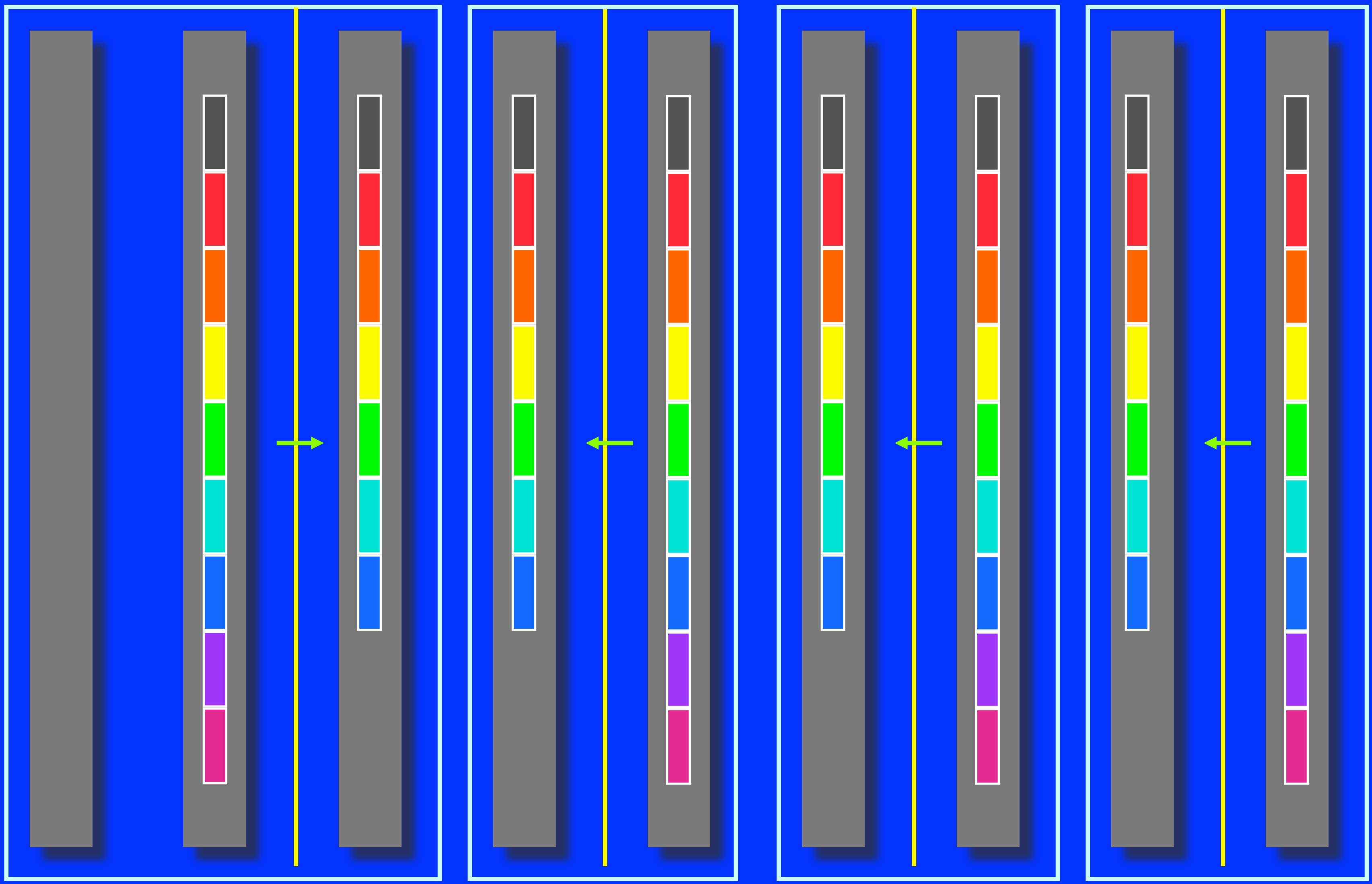


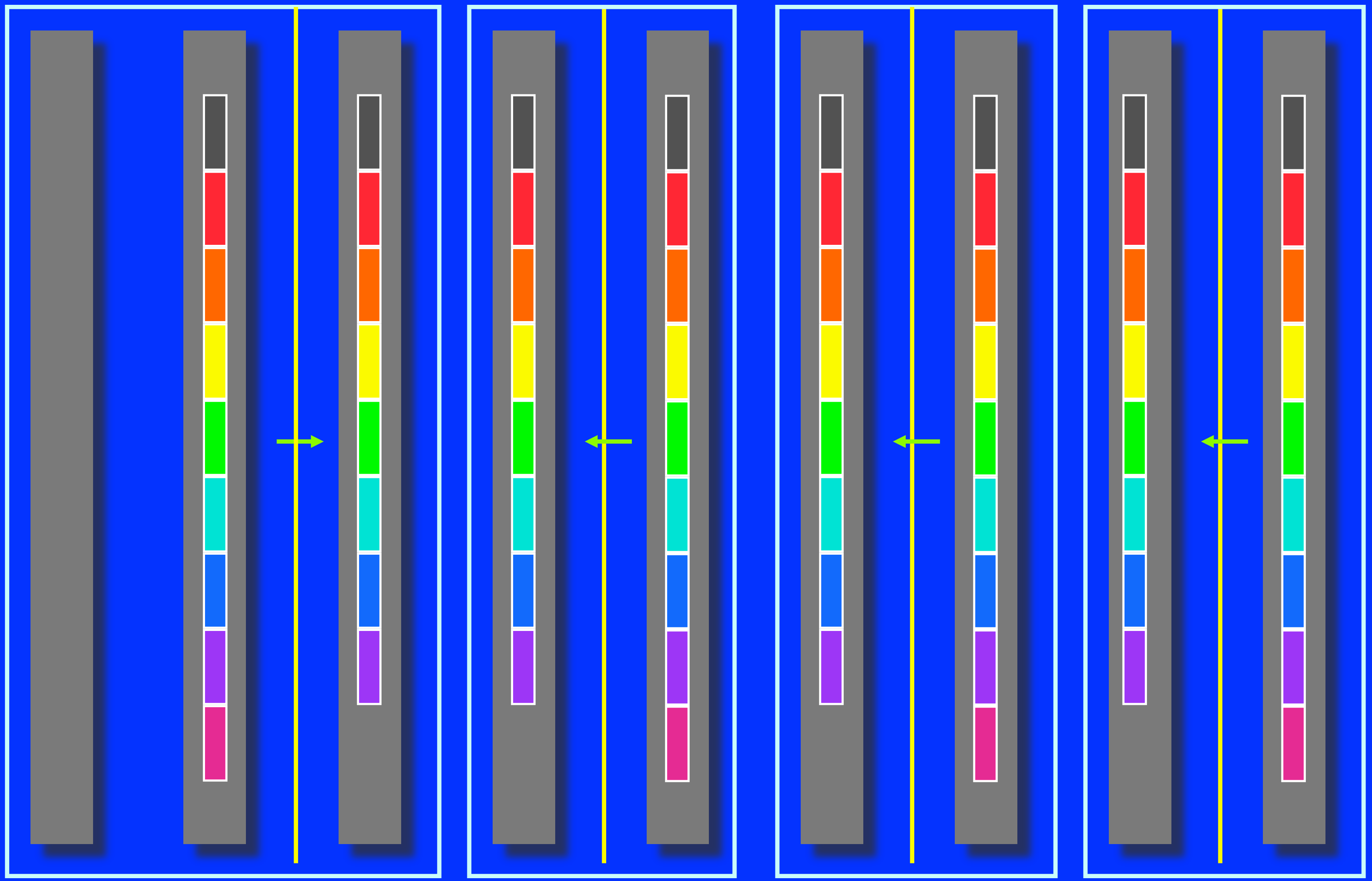


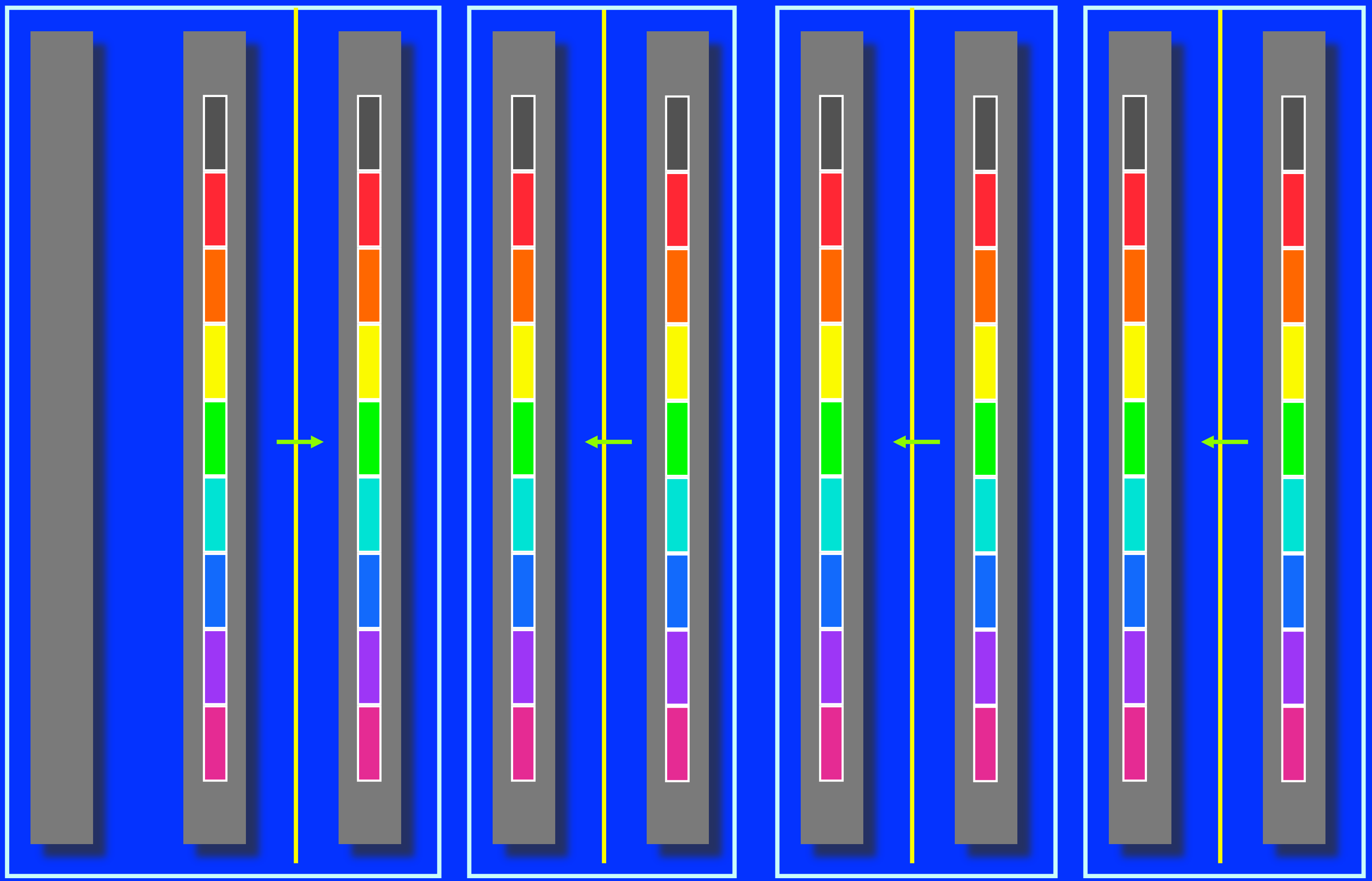


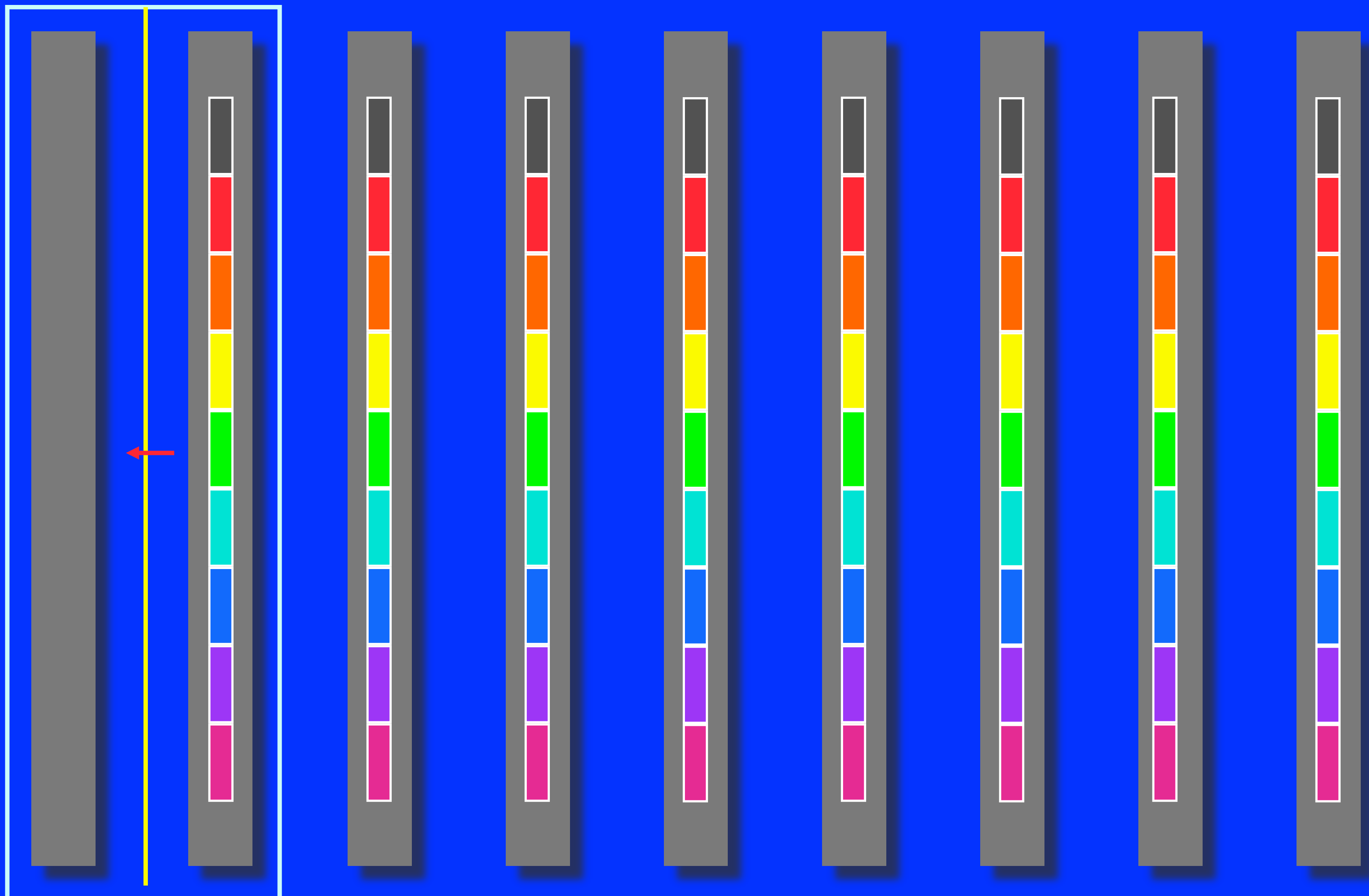


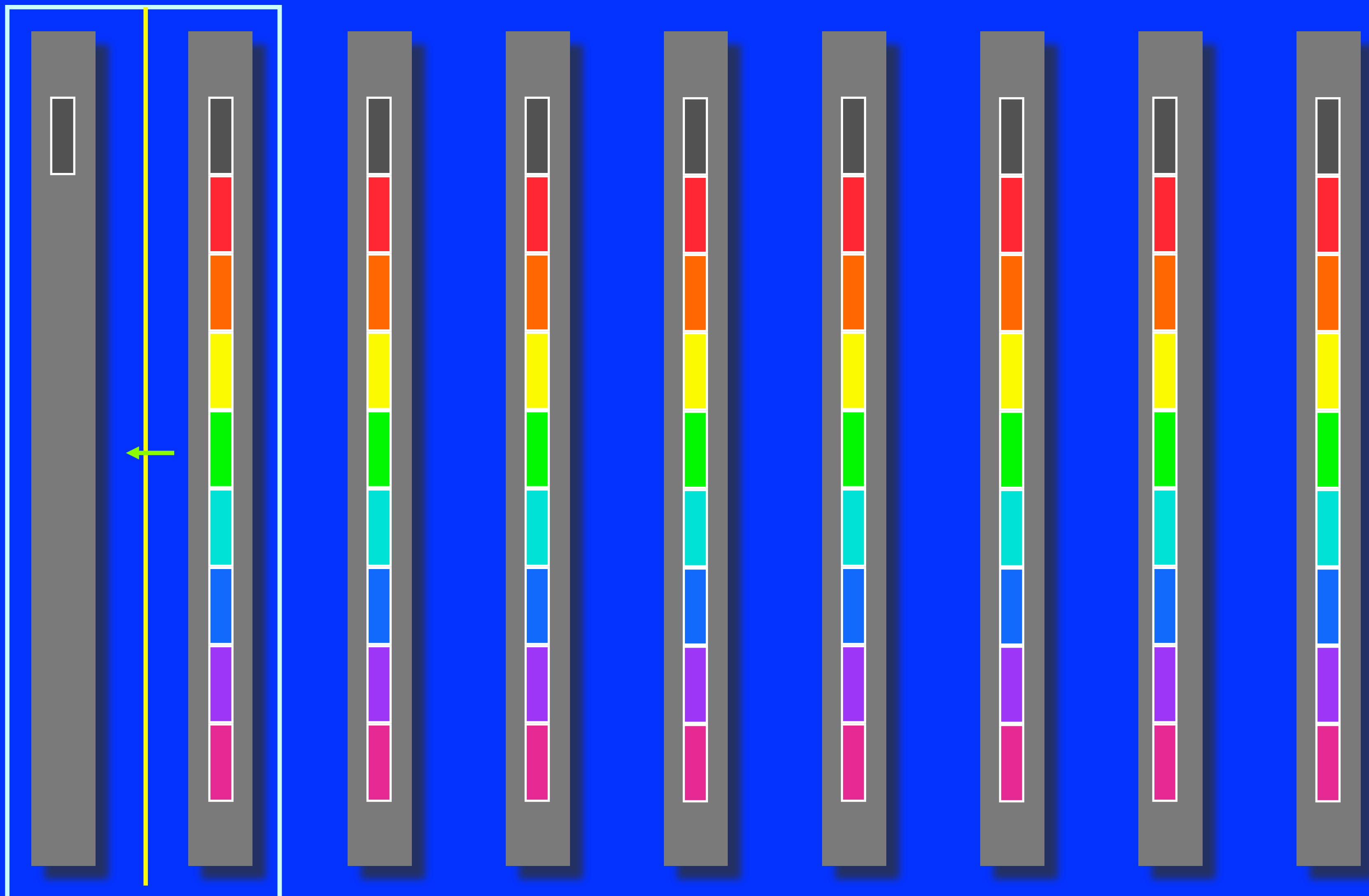


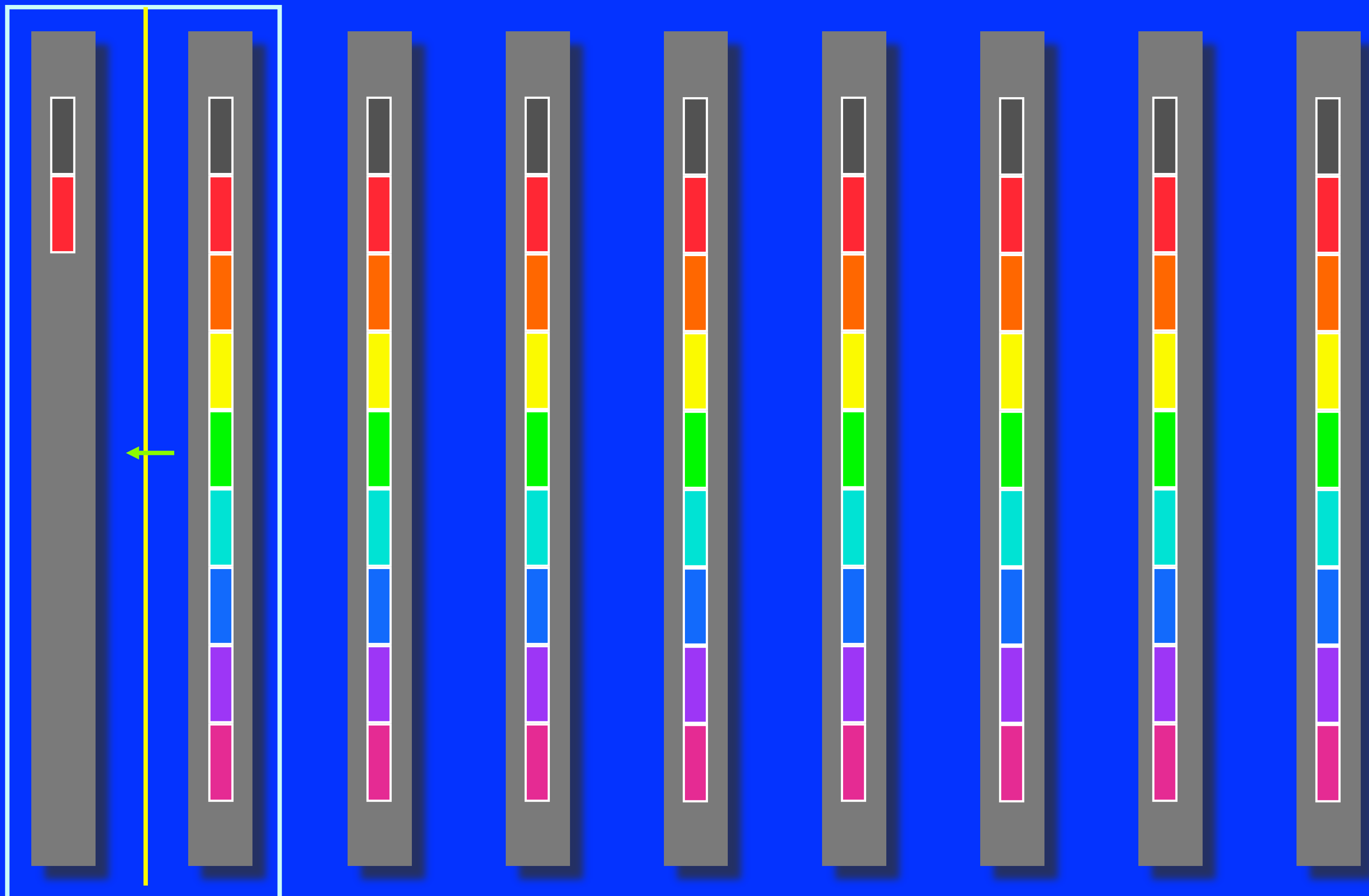


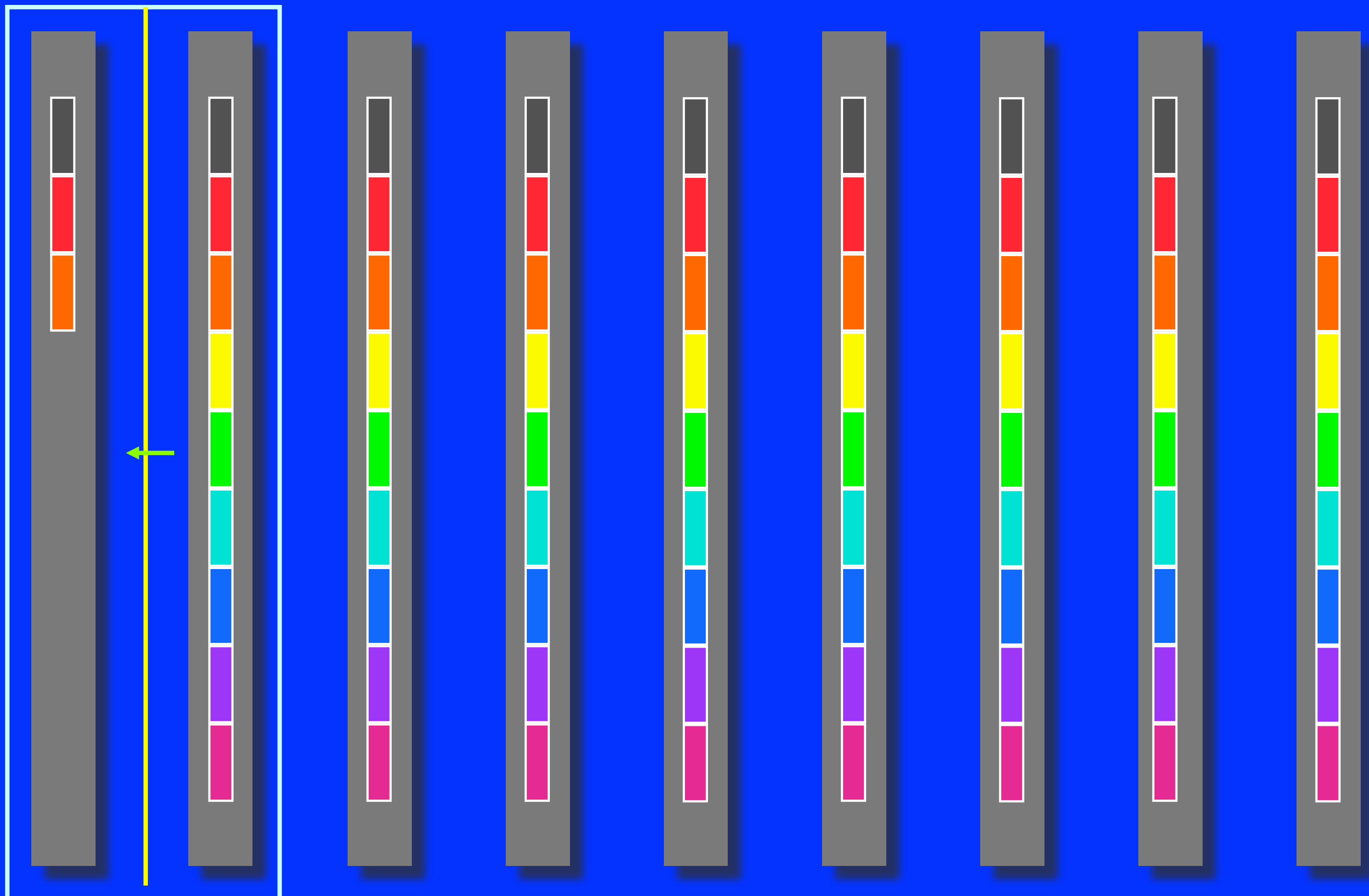


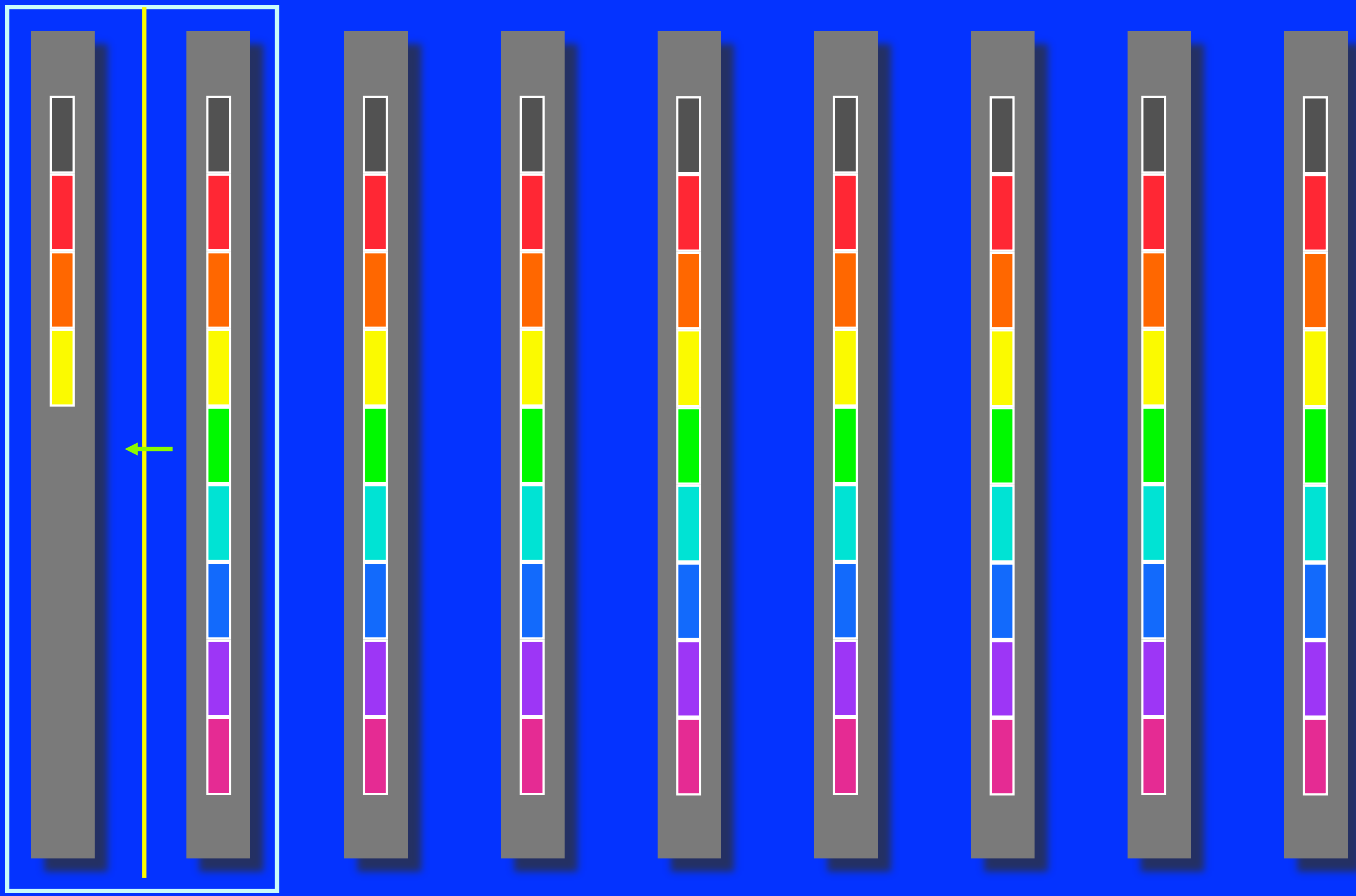


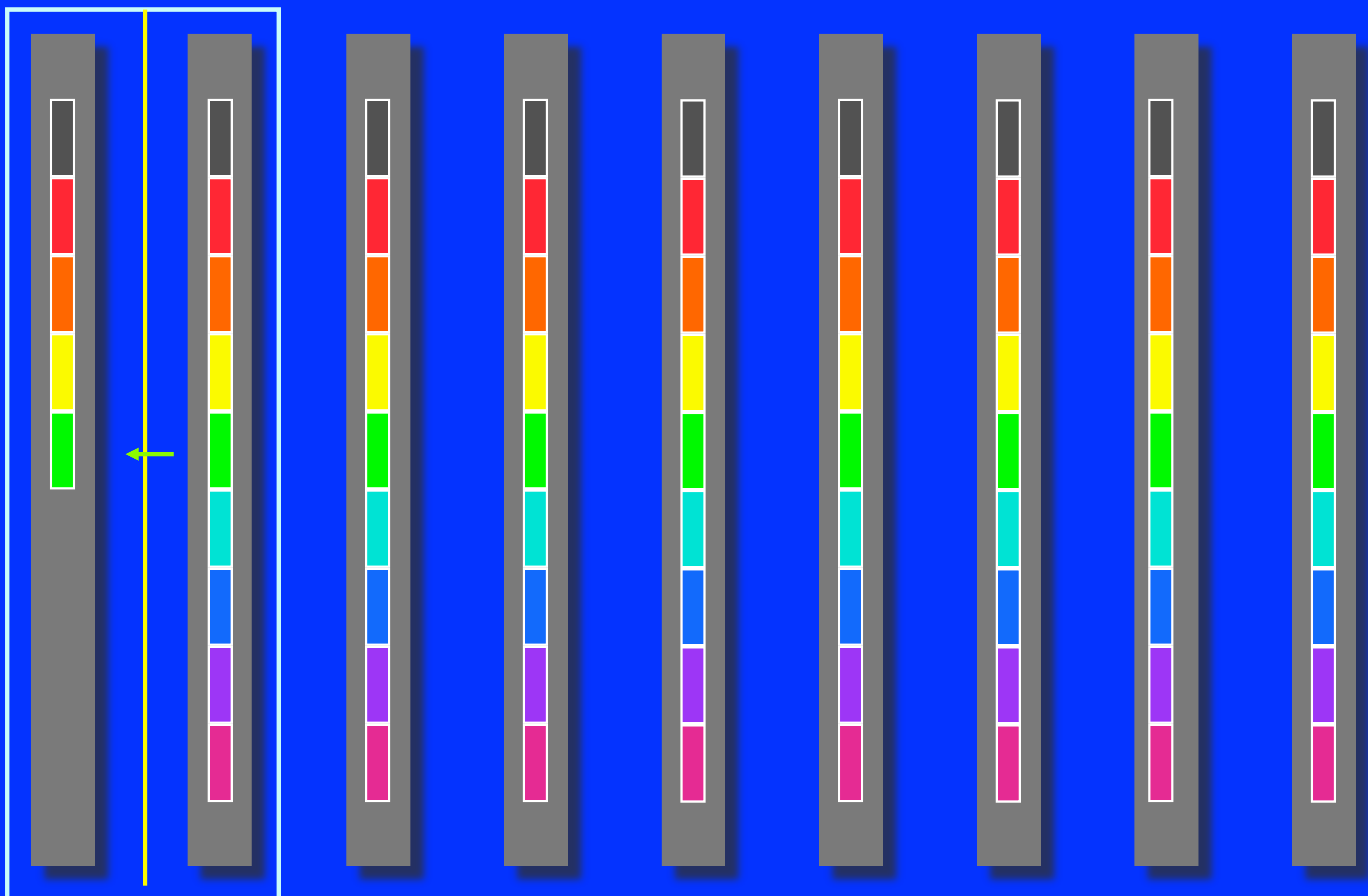


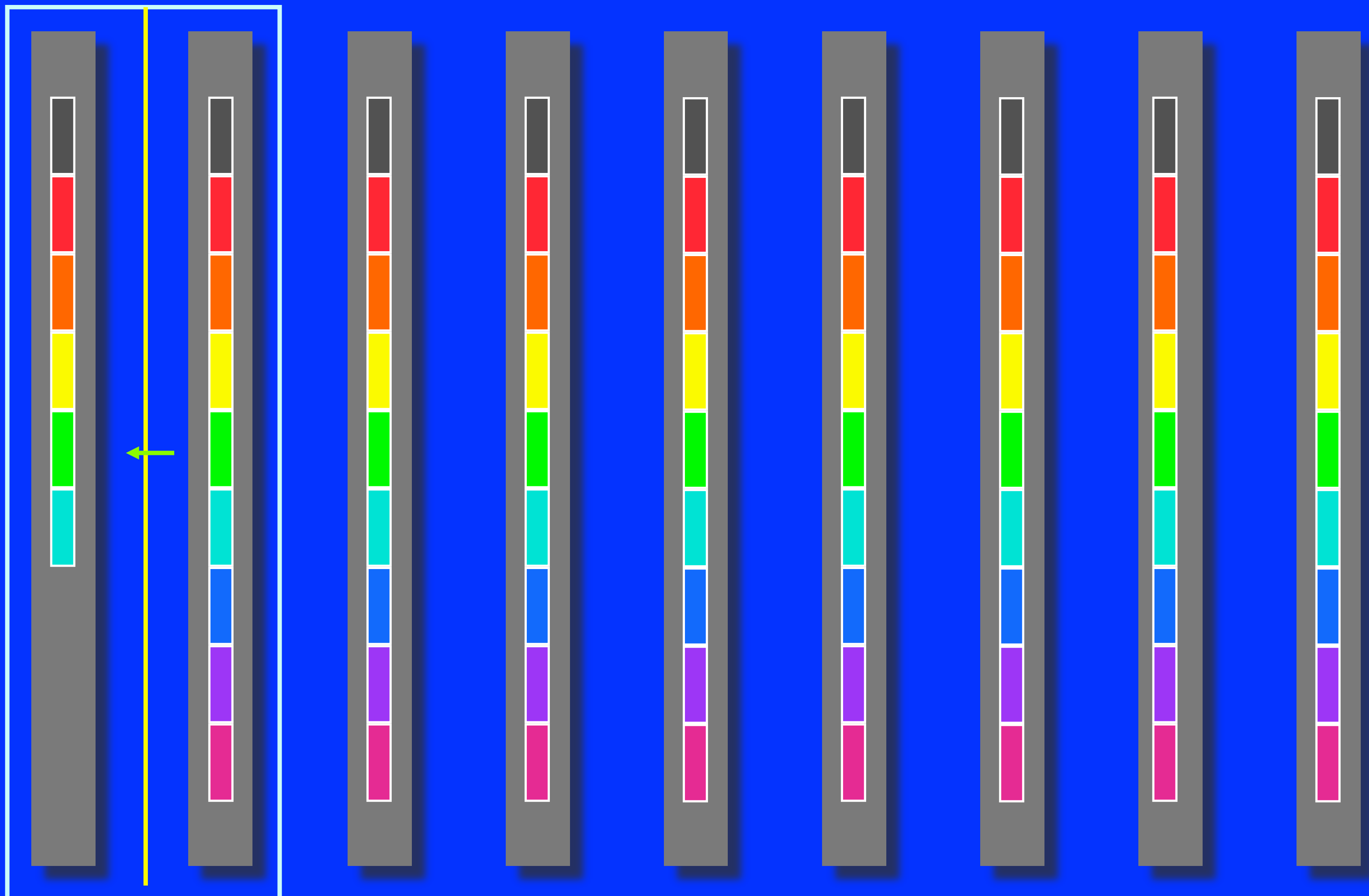


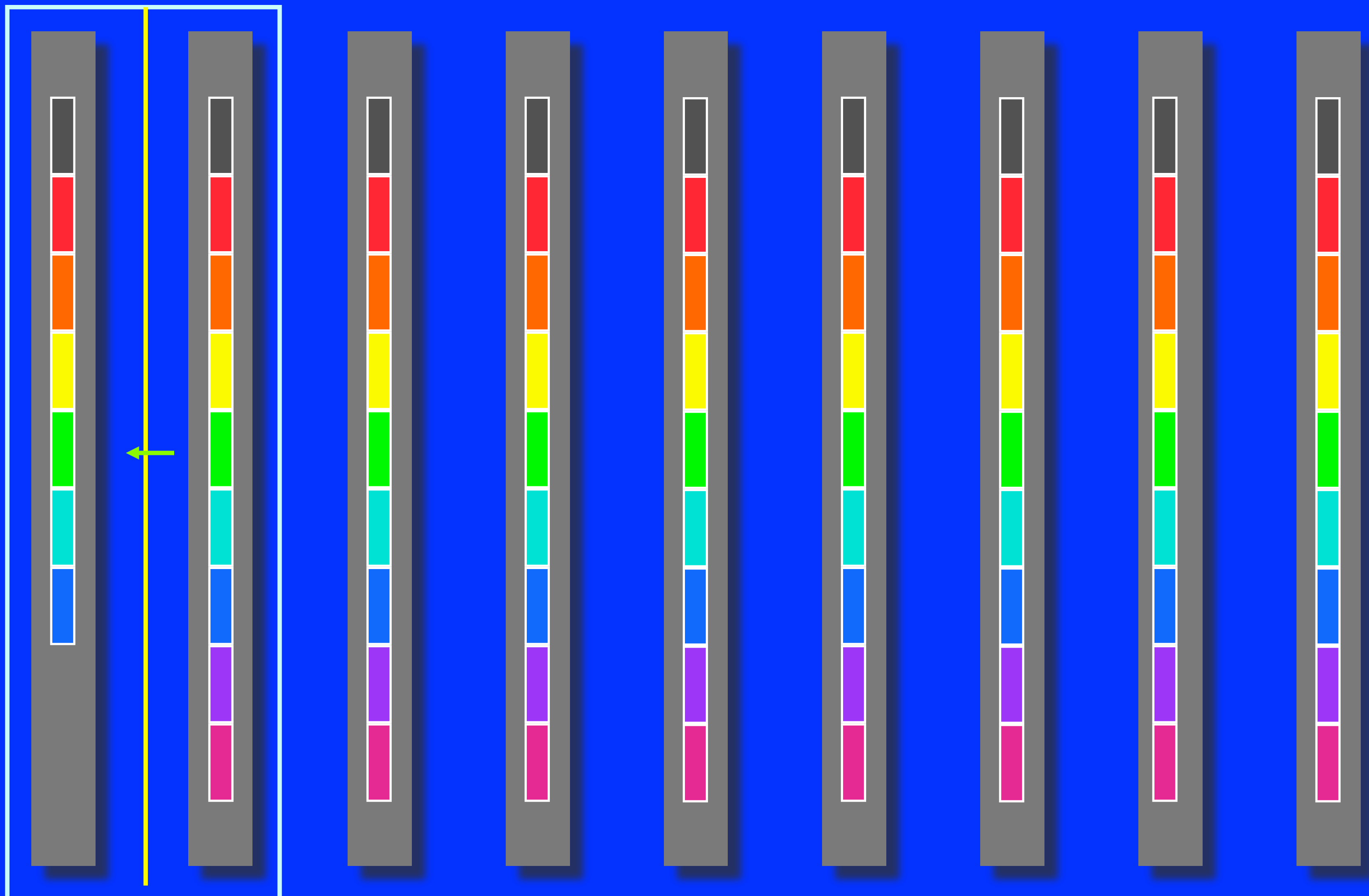


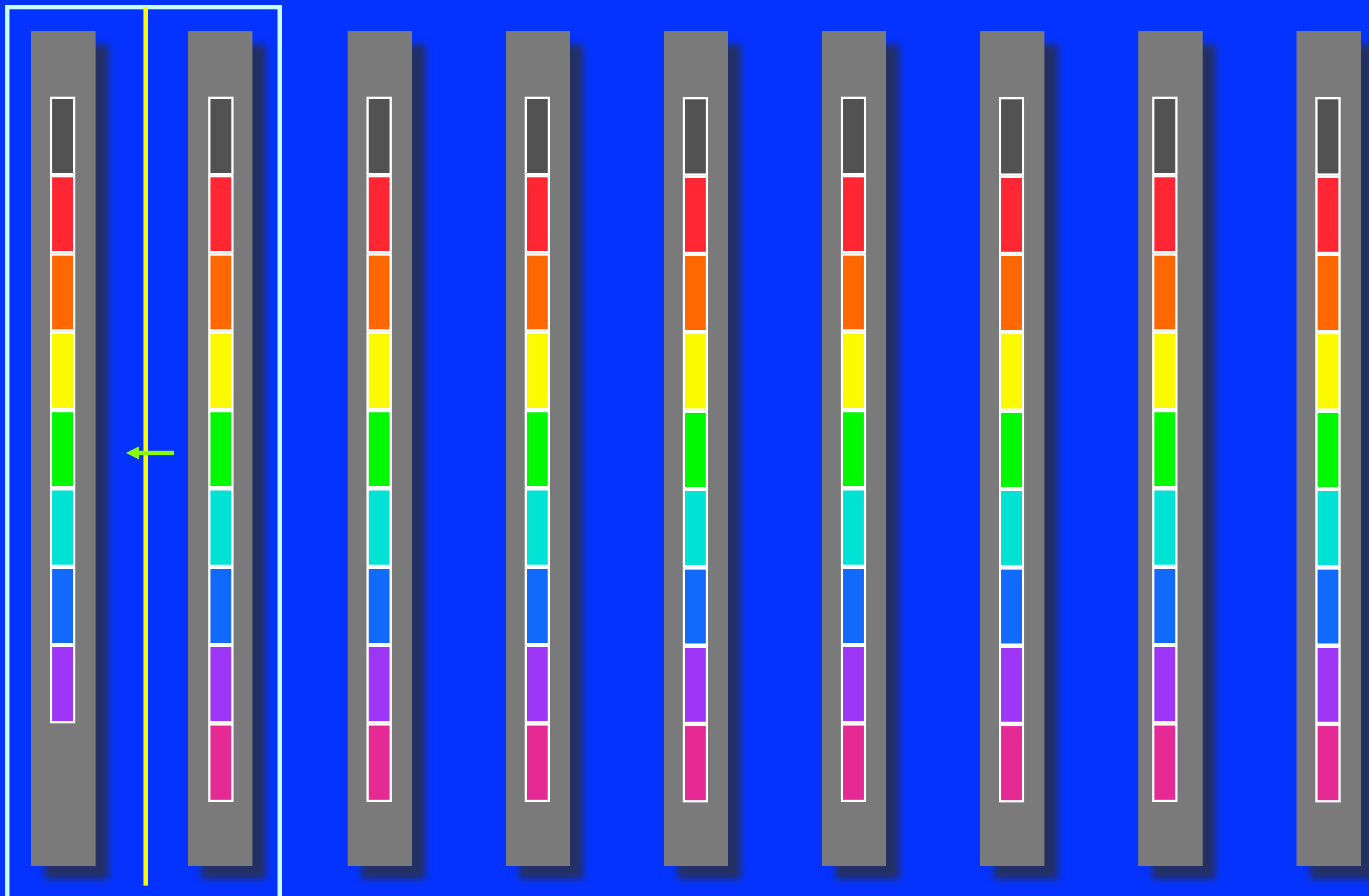


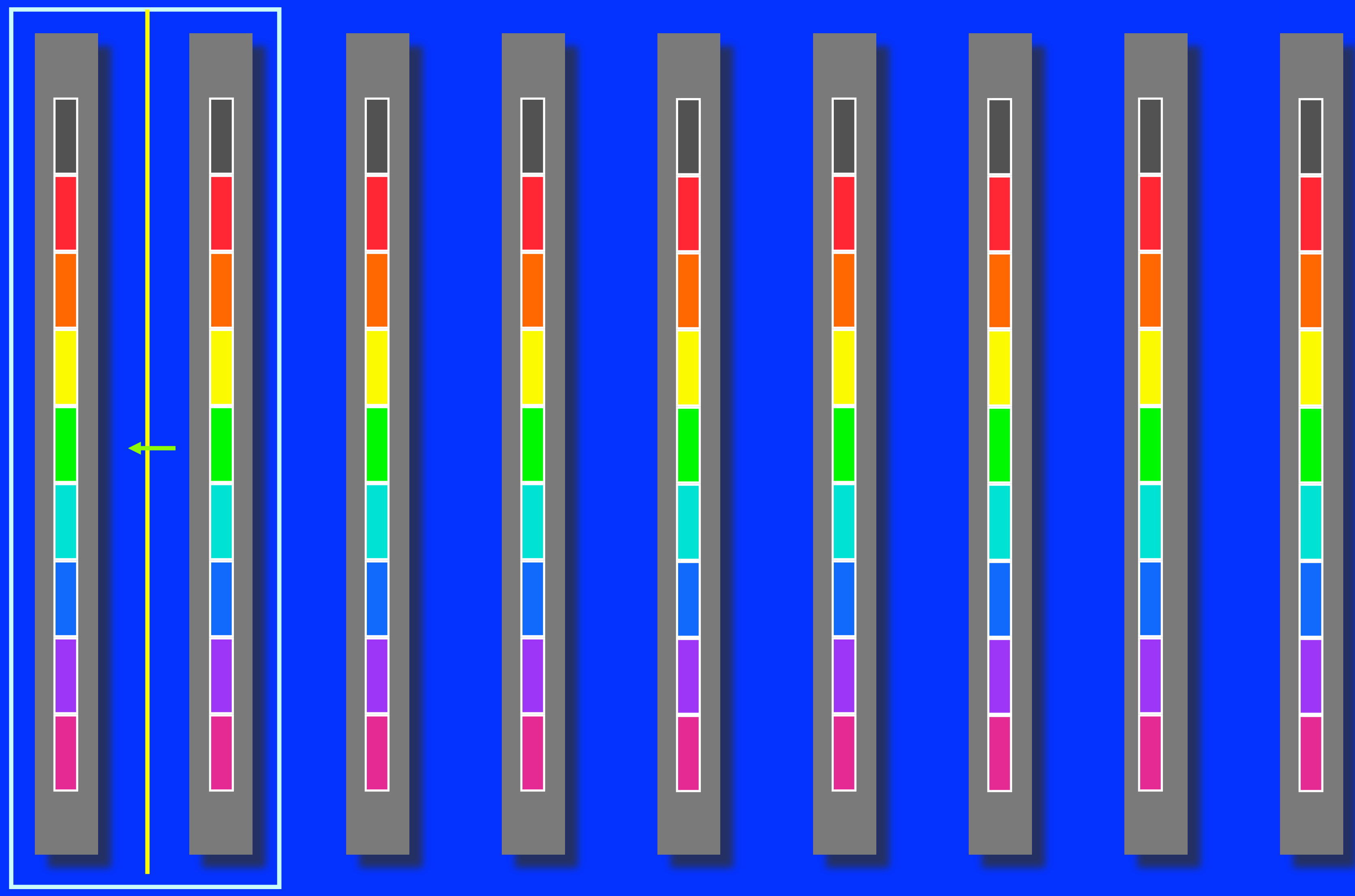


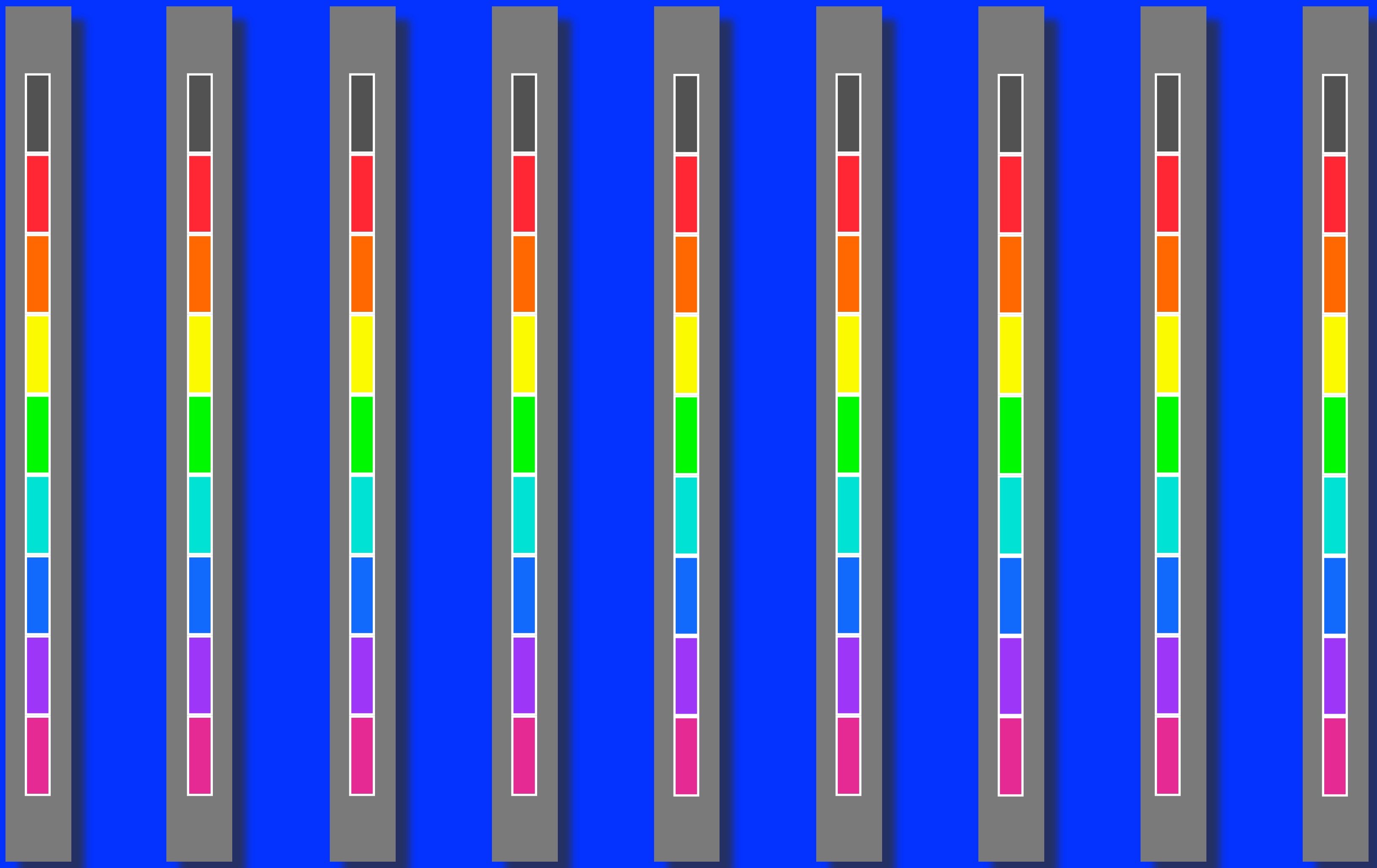












Cost of minimum spanning tree broadcast

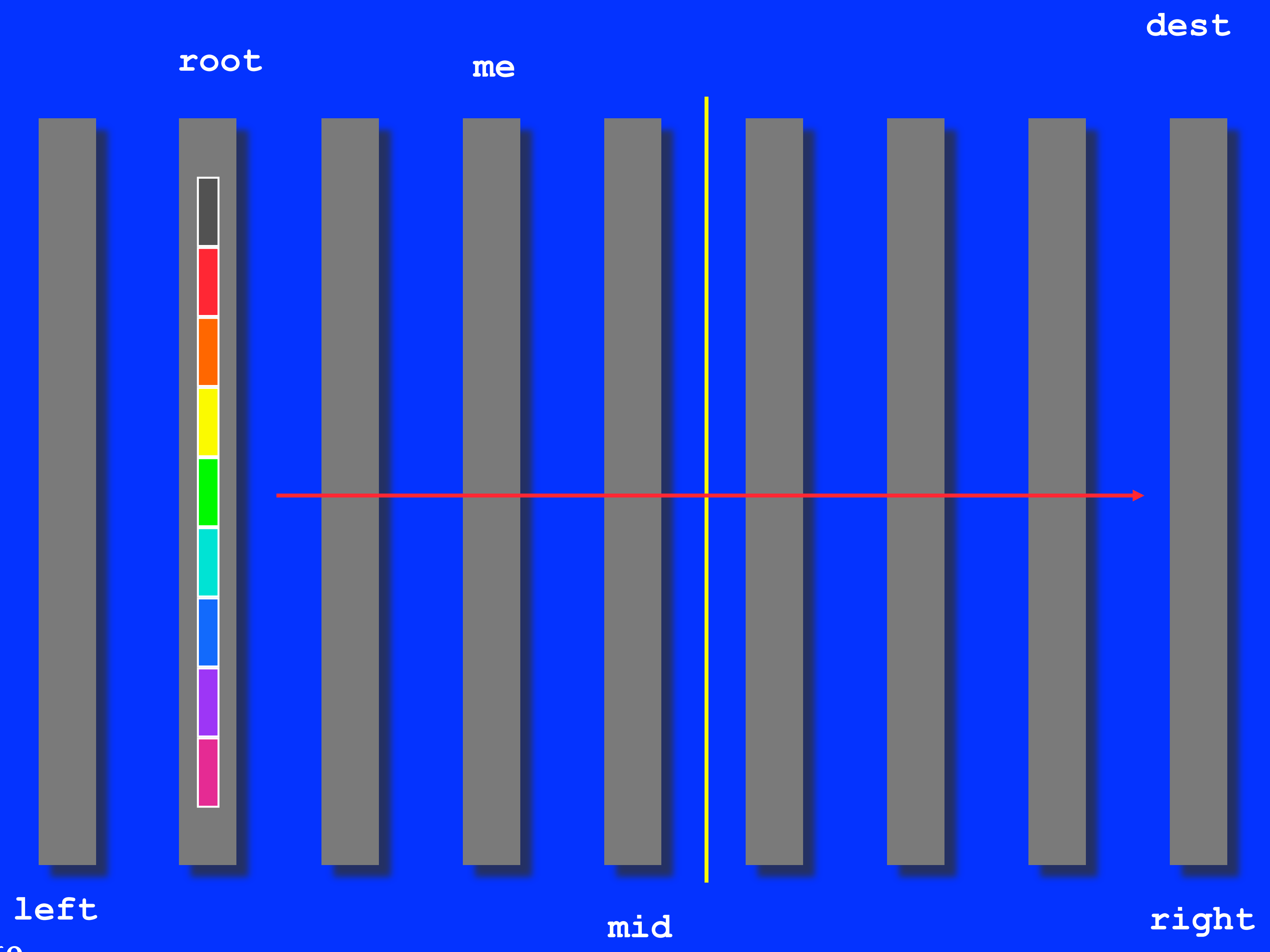
$$\underbrace{\lceil \log(p) \rceil}_{\text{number of steps}} \underbrace{(\alpha + n\beta)}_{\text{cost per steps}}$$

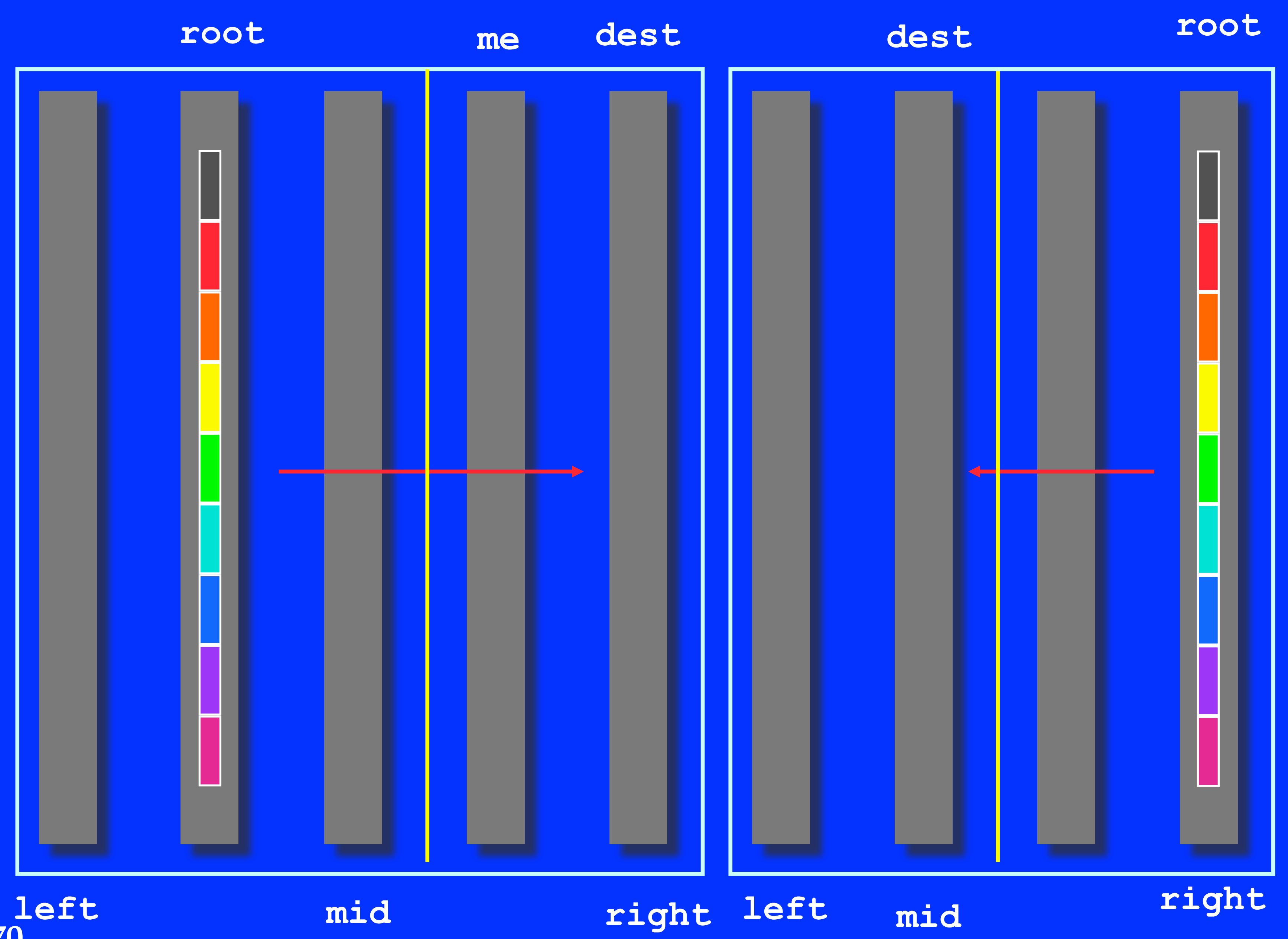
Cost of minimum spanning tree broadcast

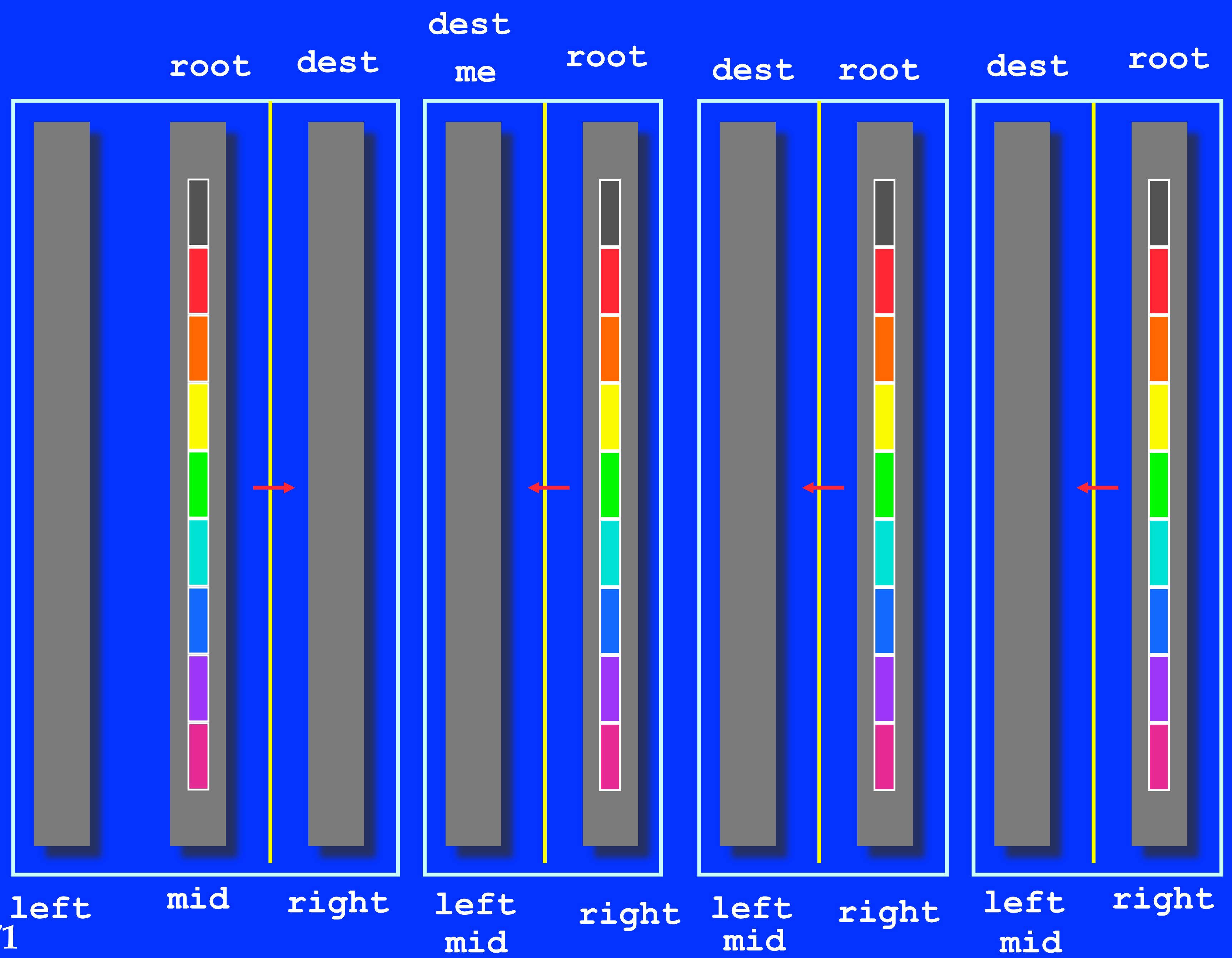
$$\underbrace{\lceil log(p) \rceil}_{\text{number of steps}} \underbrace{(\alpha + n\beta)}_{\text{cost per steps}}$$

Notice: attains lower bound for latency component

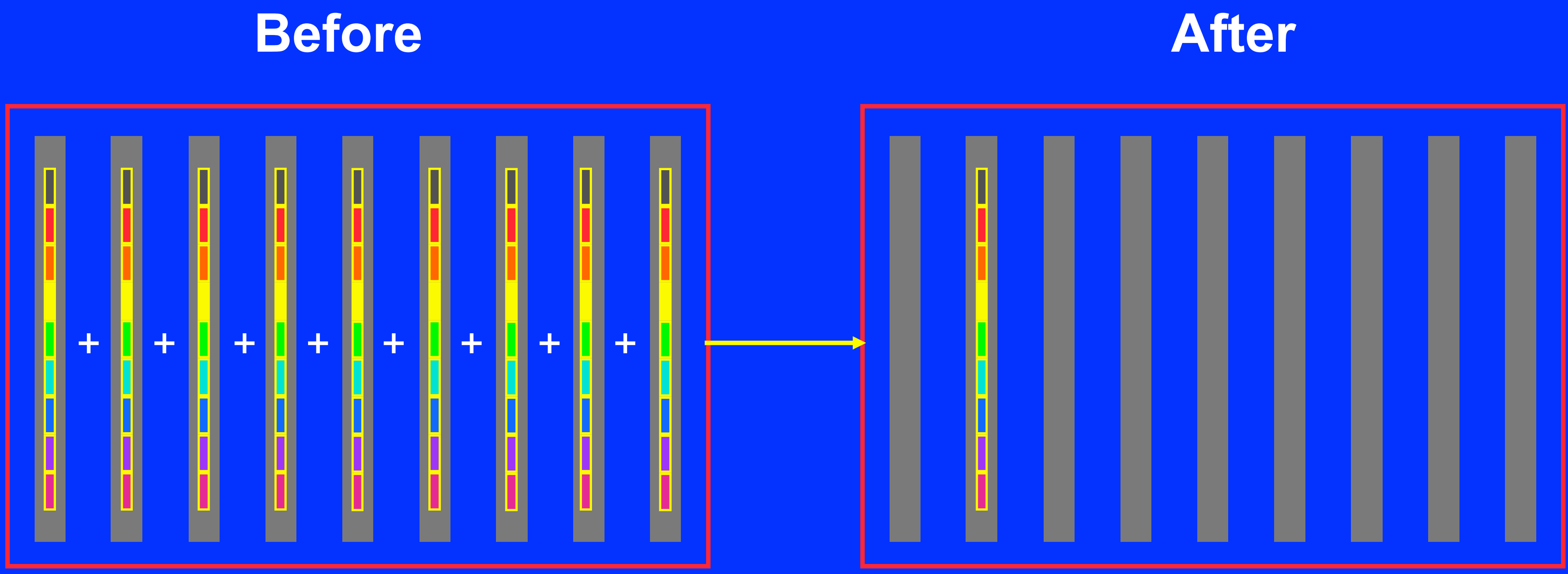
```
MSTBCAST( x, root, left, right )

if left = right return
mid = ⌊(left + right)/2⌋
if root ≤ mid then dest = right else dest = left

if me == root SEND( x, dest )
if me == dest RECV( x, root )

if me ≤ mid and root ≤ mid
    MSTBCAST( x, root, left, mid )
else if me ≤ mid and root > mid
    MSTBCAST( x, dest, left, mid )
else if me > mid and root ≤ mid
    MSTBCAST( x, dest, mid+1, right )
else if me > mid and root > mid
    MSTBCAST( x, root, mid+1, right )
```

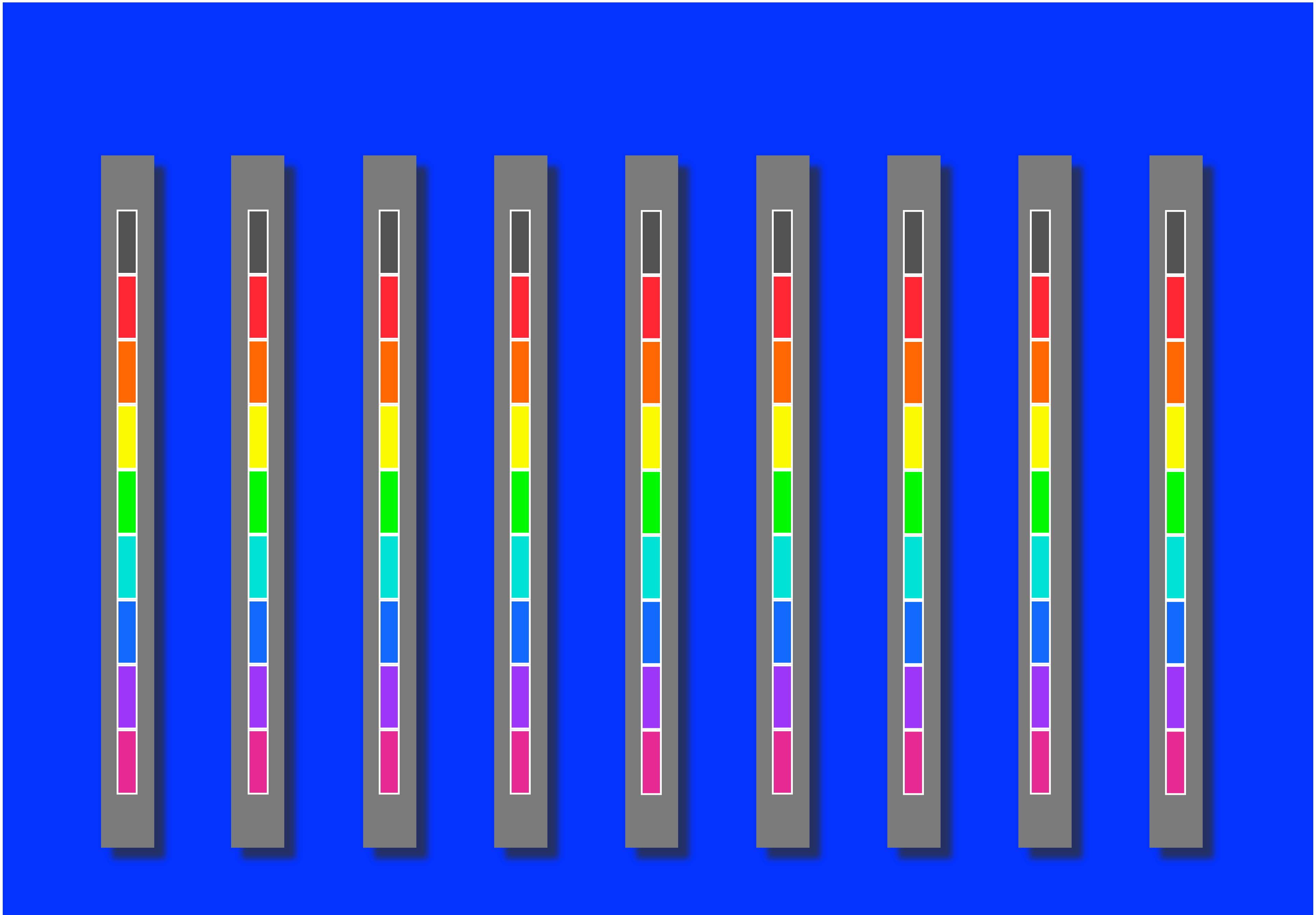

root
me
dest
left
mid
right

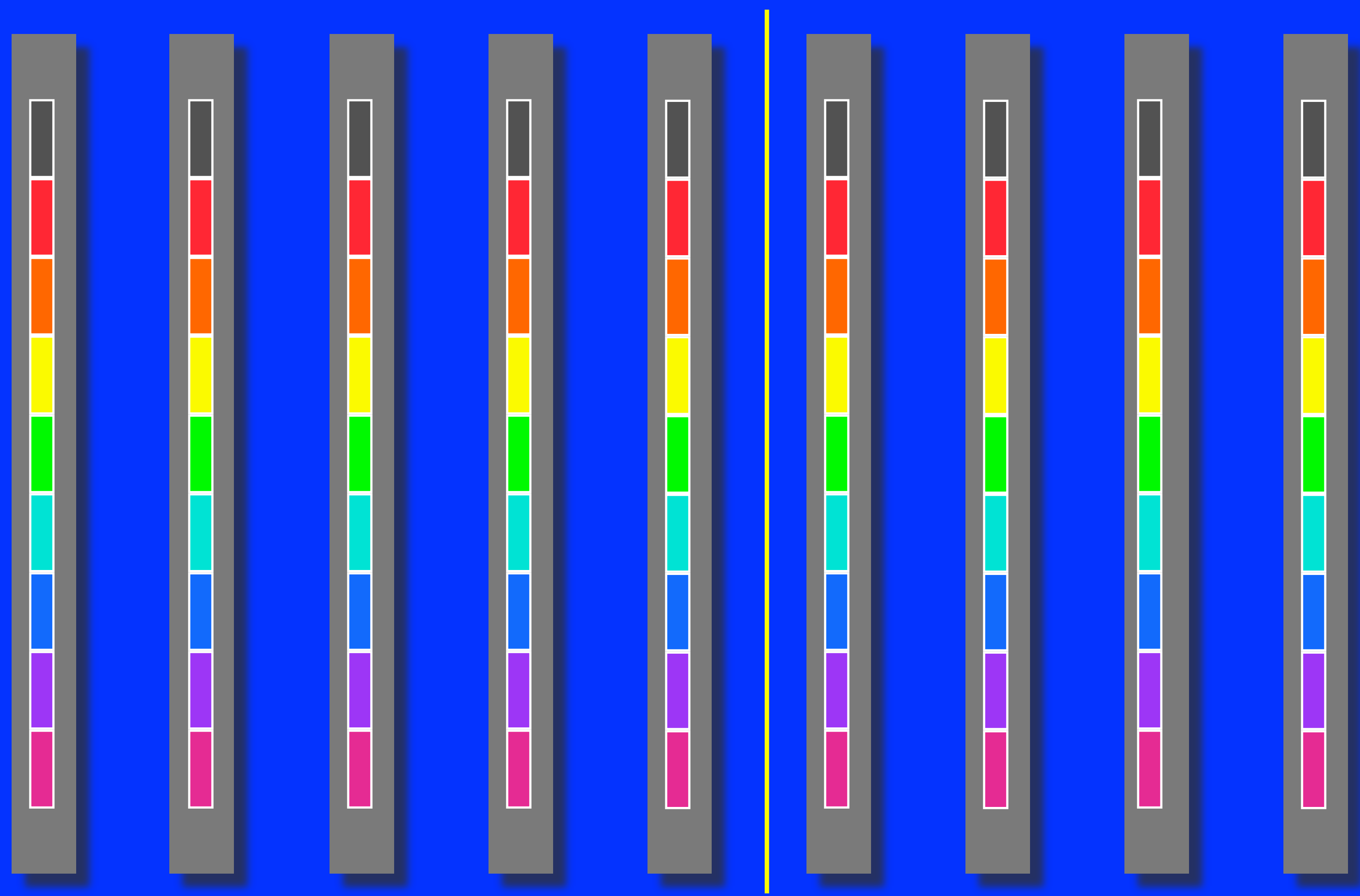

root
me
dest
dest
root
left
mid
right
left
mid
right

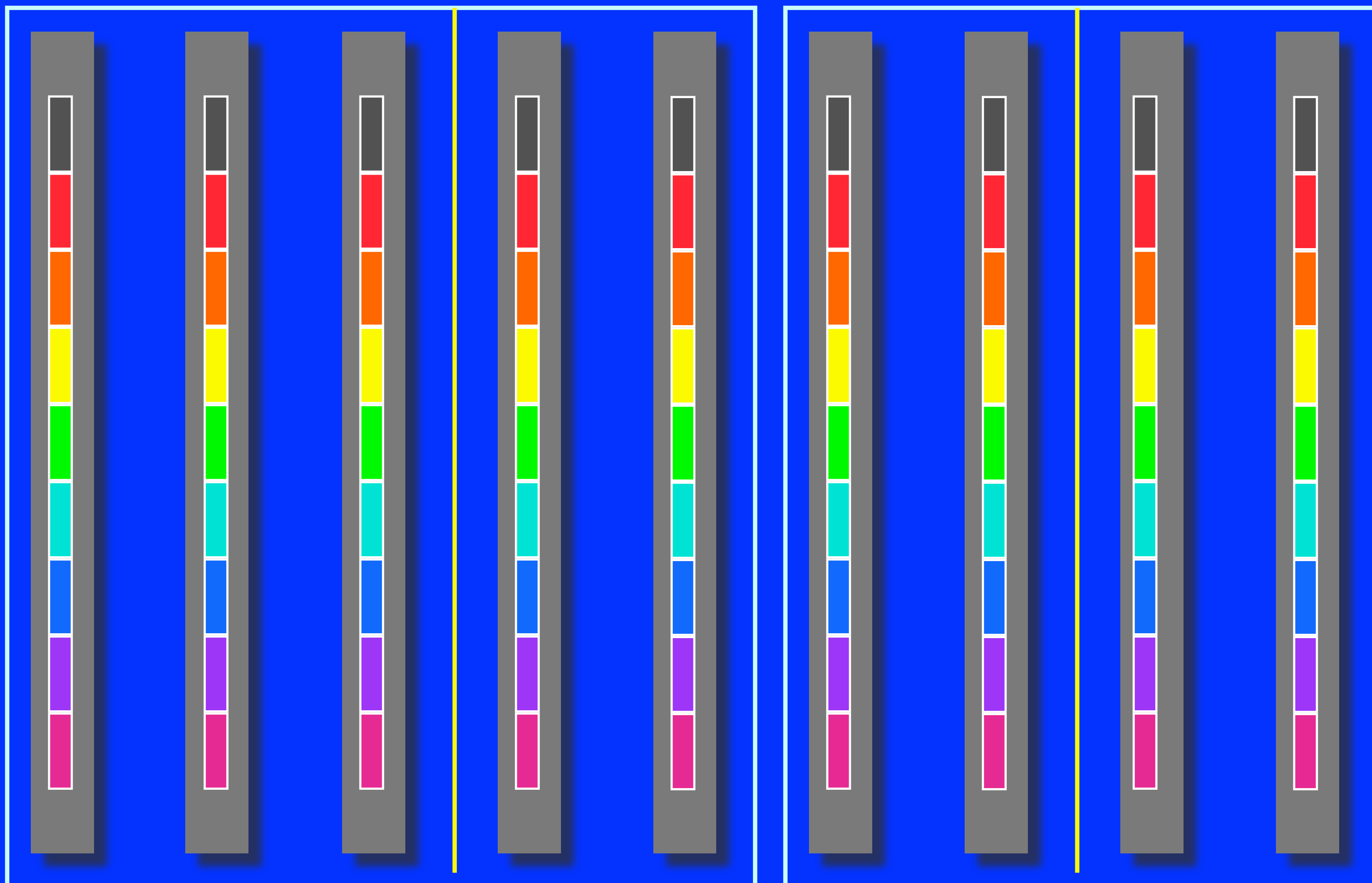

root
dest
dest
me
root
dest
root
dest
root
left
mid
right
left
mid
right
left
mid
right
left
mid
right

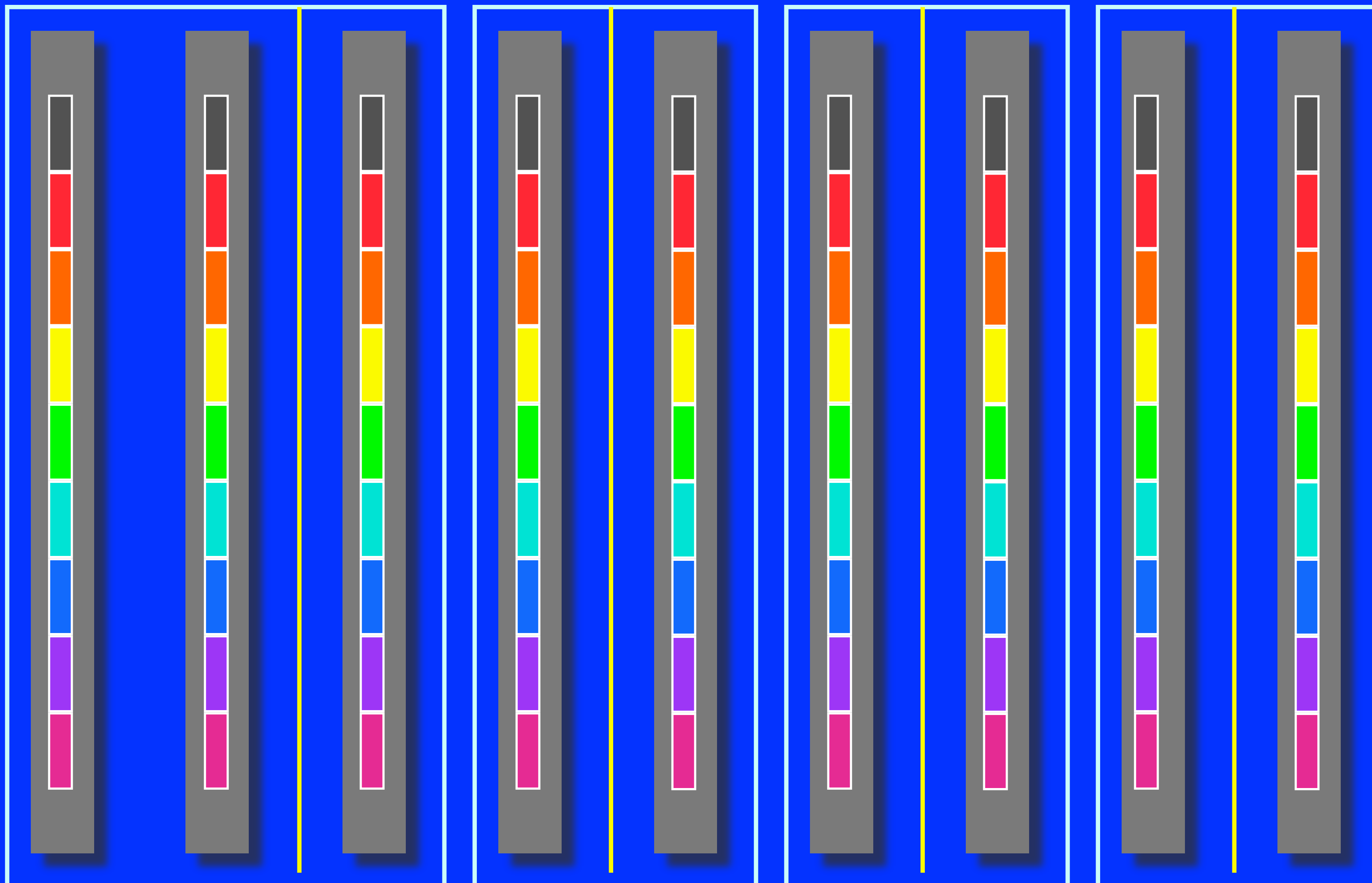
Reduce(-to-one)

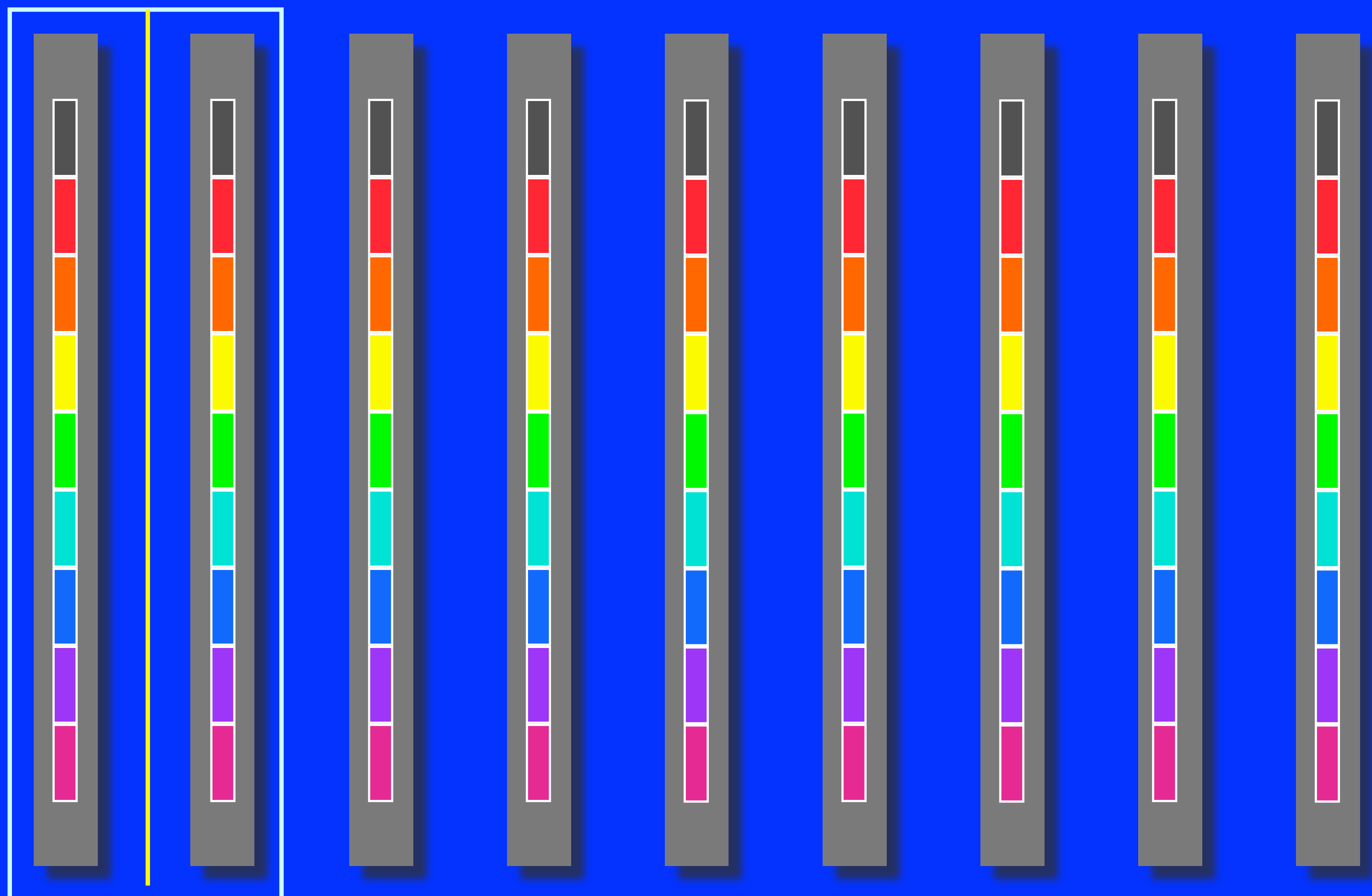


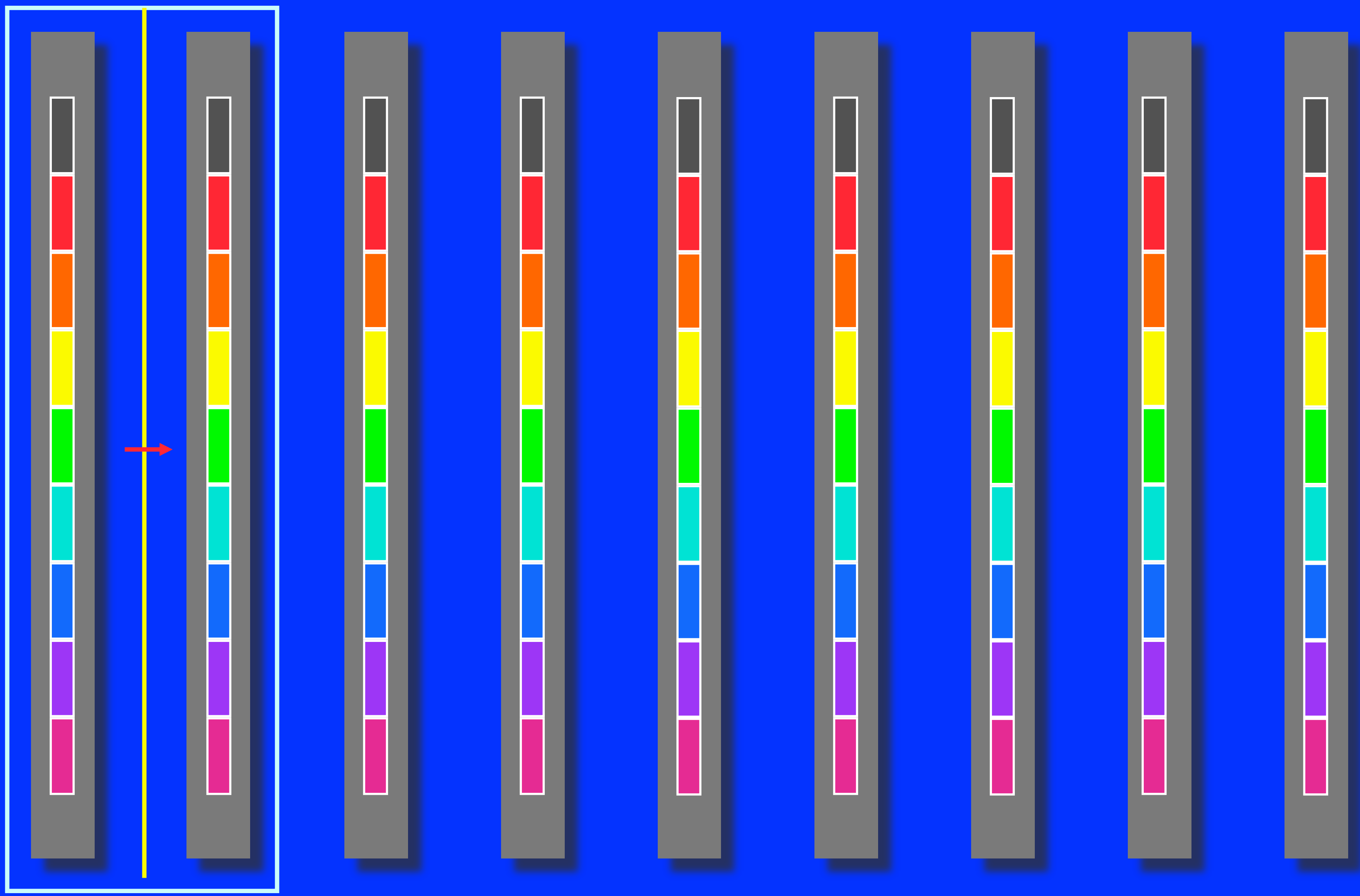


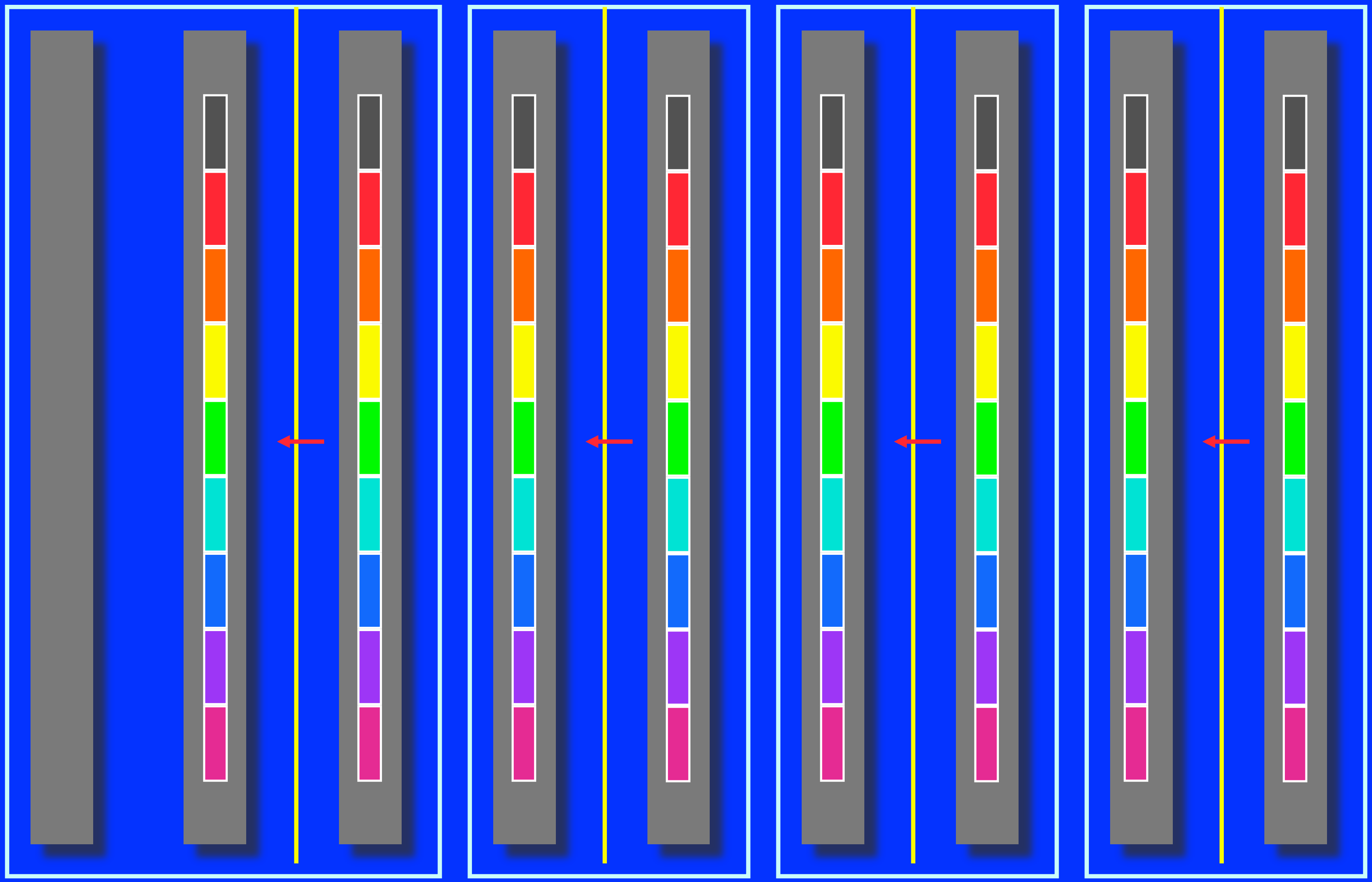


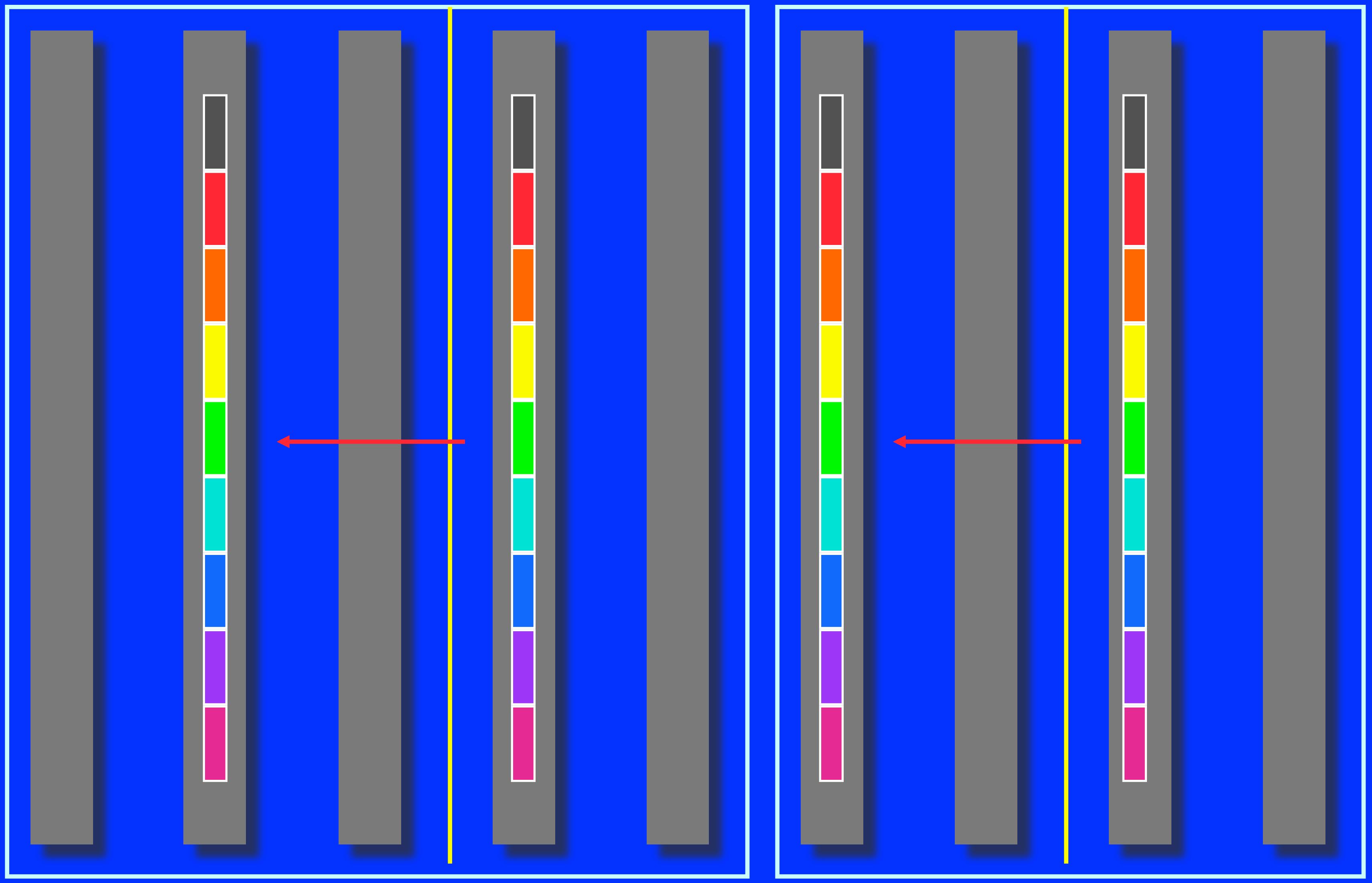


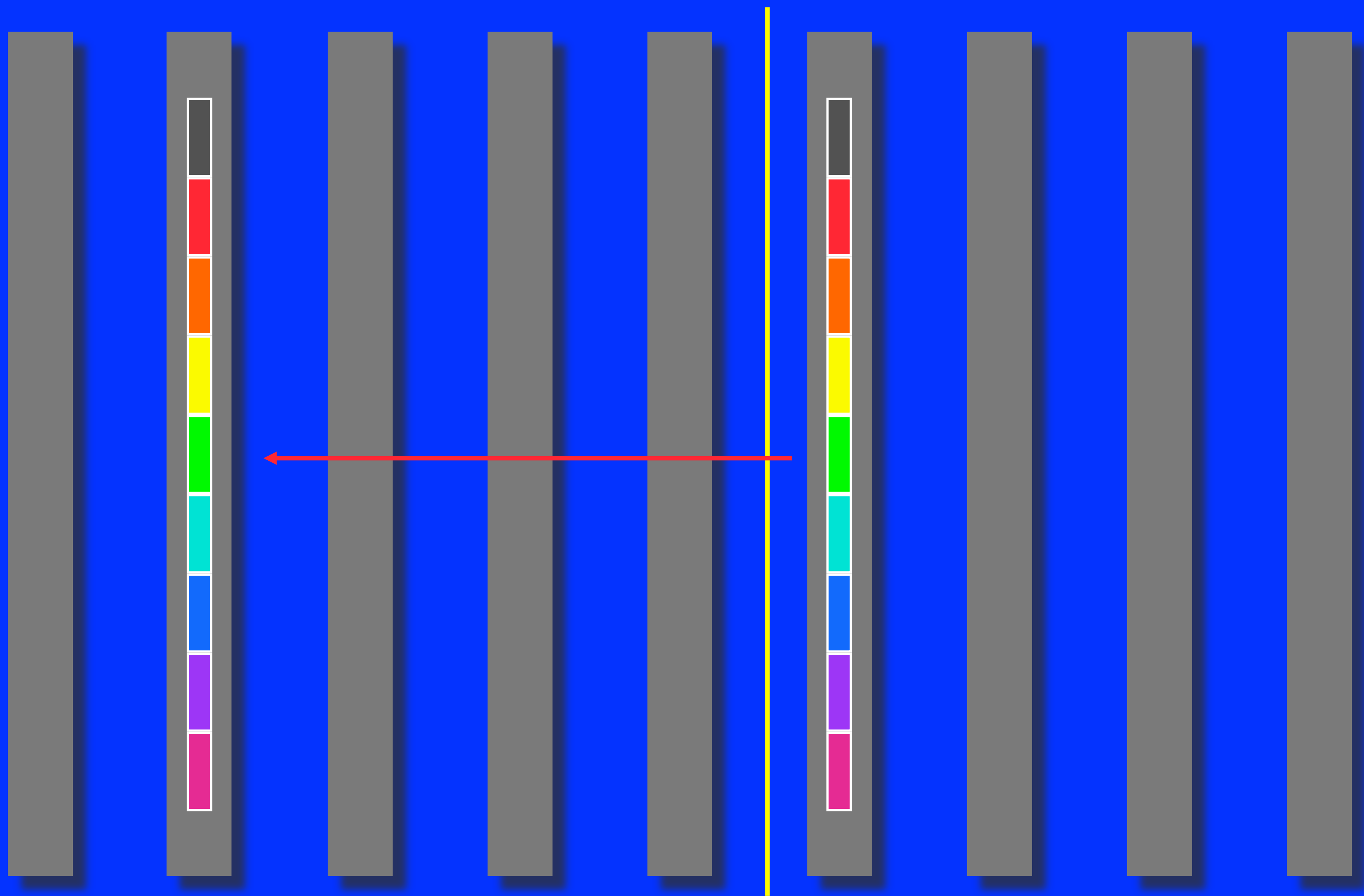












Cost of minimum spanning tree reduce(-to-one)

$$\underbrace{\lceil \log(p) \rceil}_{\text{number of steps}} \underbrace{(\alpha + n\beta + n\gamma)}_{\text{cost per steps}}$$

number of steps

cost per steps

Cost of minimum spanning tree reduce(-to-one)

$$\lceil log(p) \rceil \left(\alpha + n\beta + n\gamma\right)$$

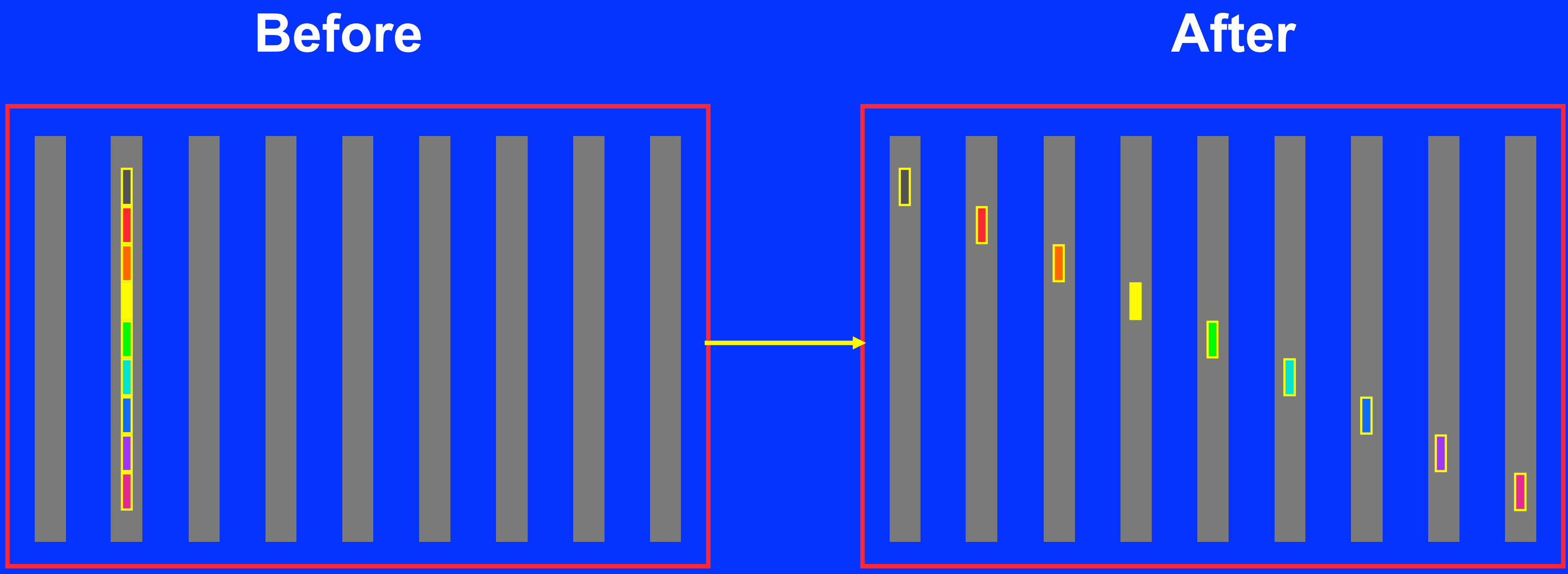
number of steps

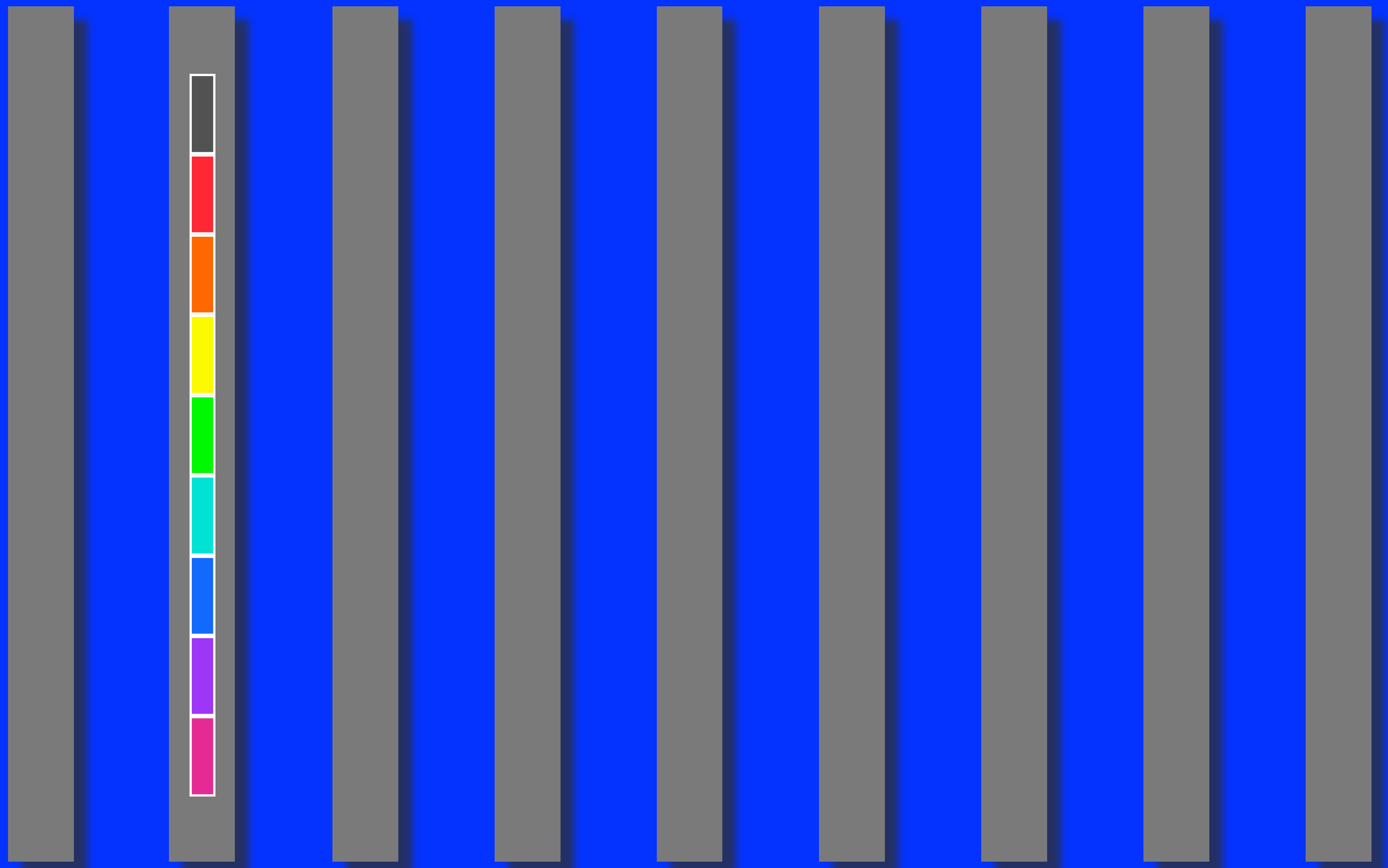
cost per steps

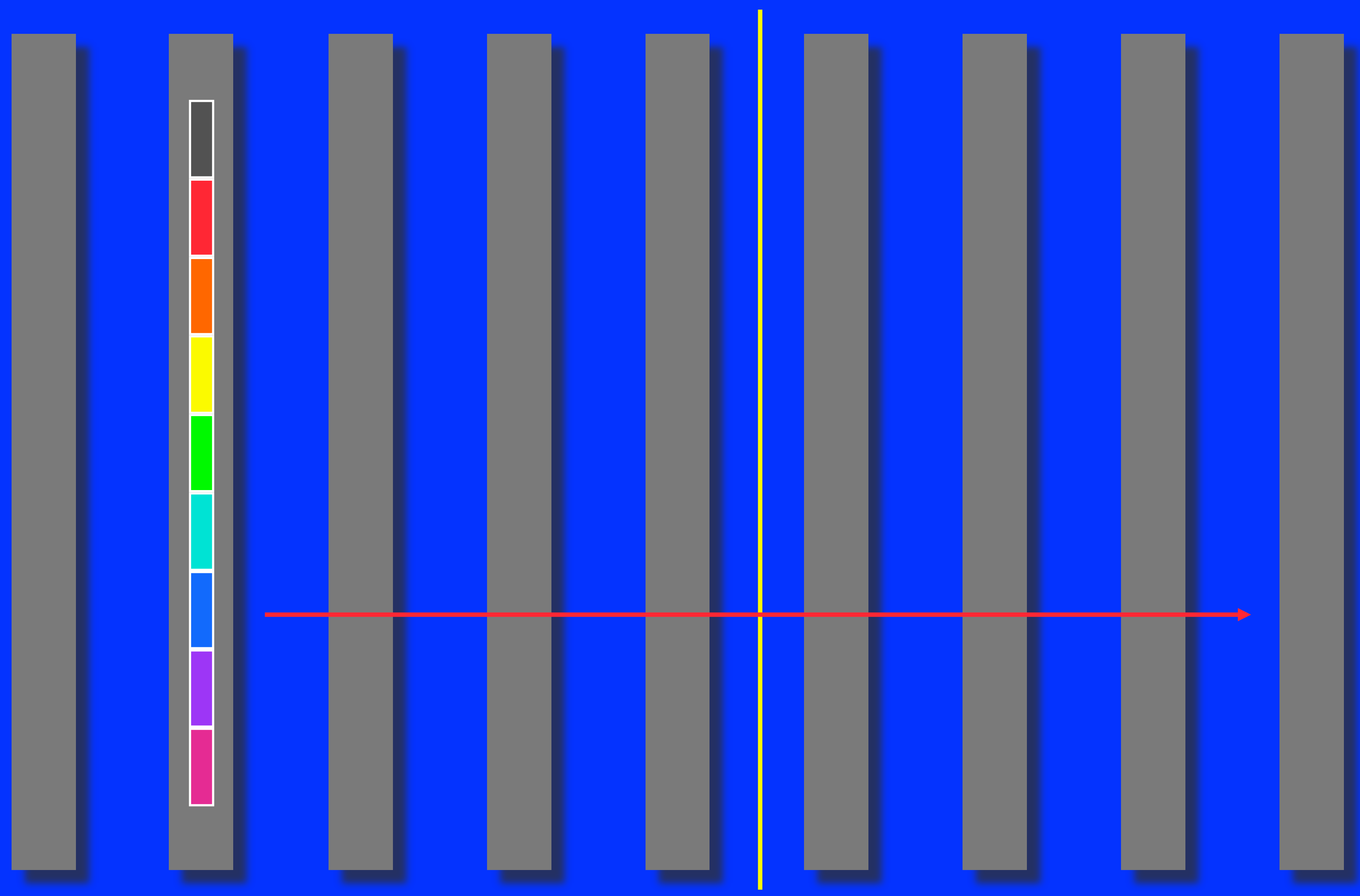
Notice: attains lower bound for latency component

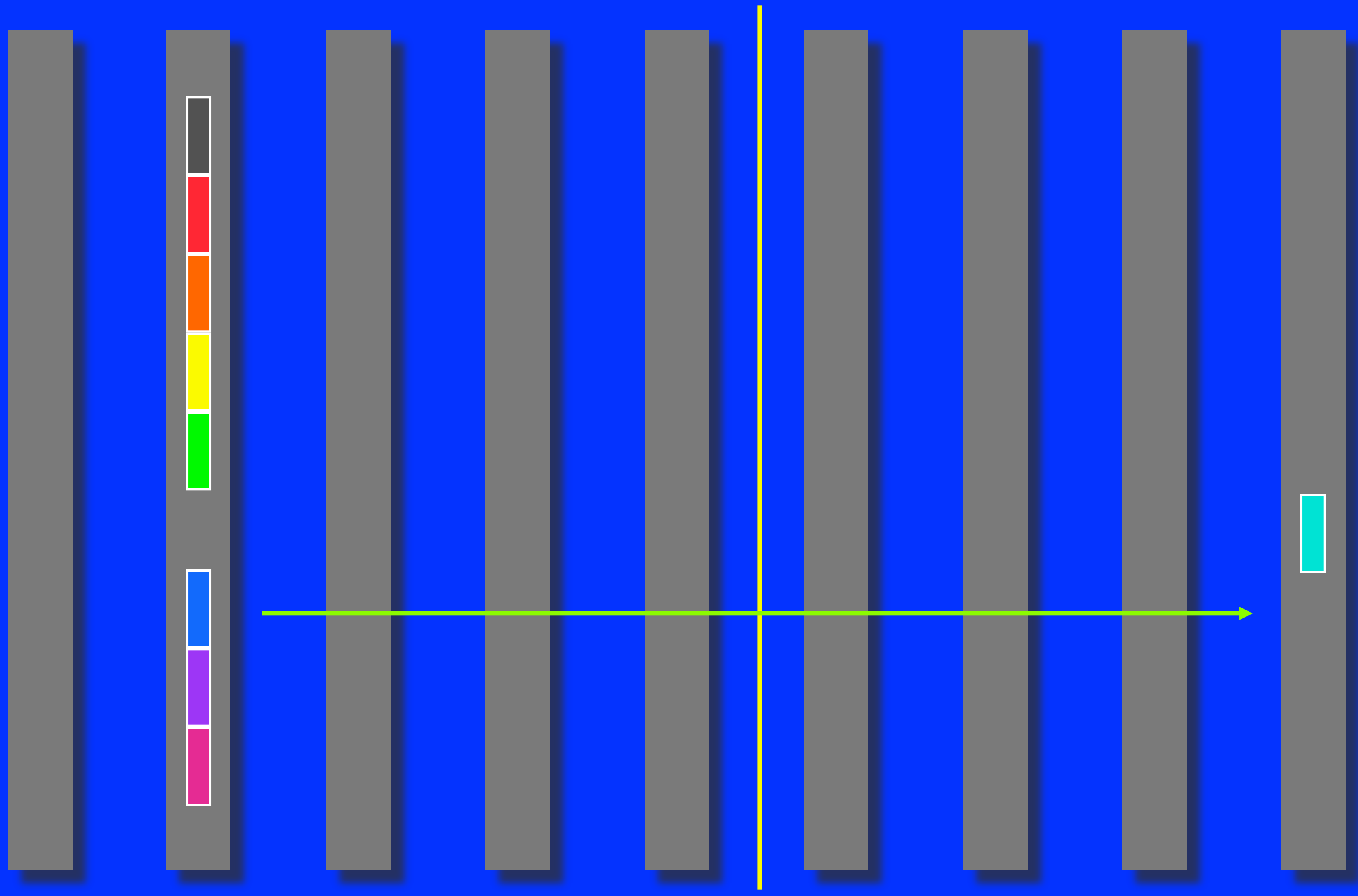
```
MSTReduce( x, root, left, right )

if left = right return
mid = ⌊(left + right)/2⌋
if root ≤ mid then srce = right else srce = left

if me ≤ mid and root ≤ mid
    MSTReduce( x, root, left, mid )
else if me ≤ mid and root > mid
    MSTReduce( x, srce, left, mid )
else if me > mid and root ≤ mid
    MSTReduce( x, srce, mid+1, right )
else if me > mid and root > mid
    MSTReduce( x, root, mid+1, right )

if me == srce Send( x, root )
if me == root Recv( tmp, srce ) and x = x + tmp
```

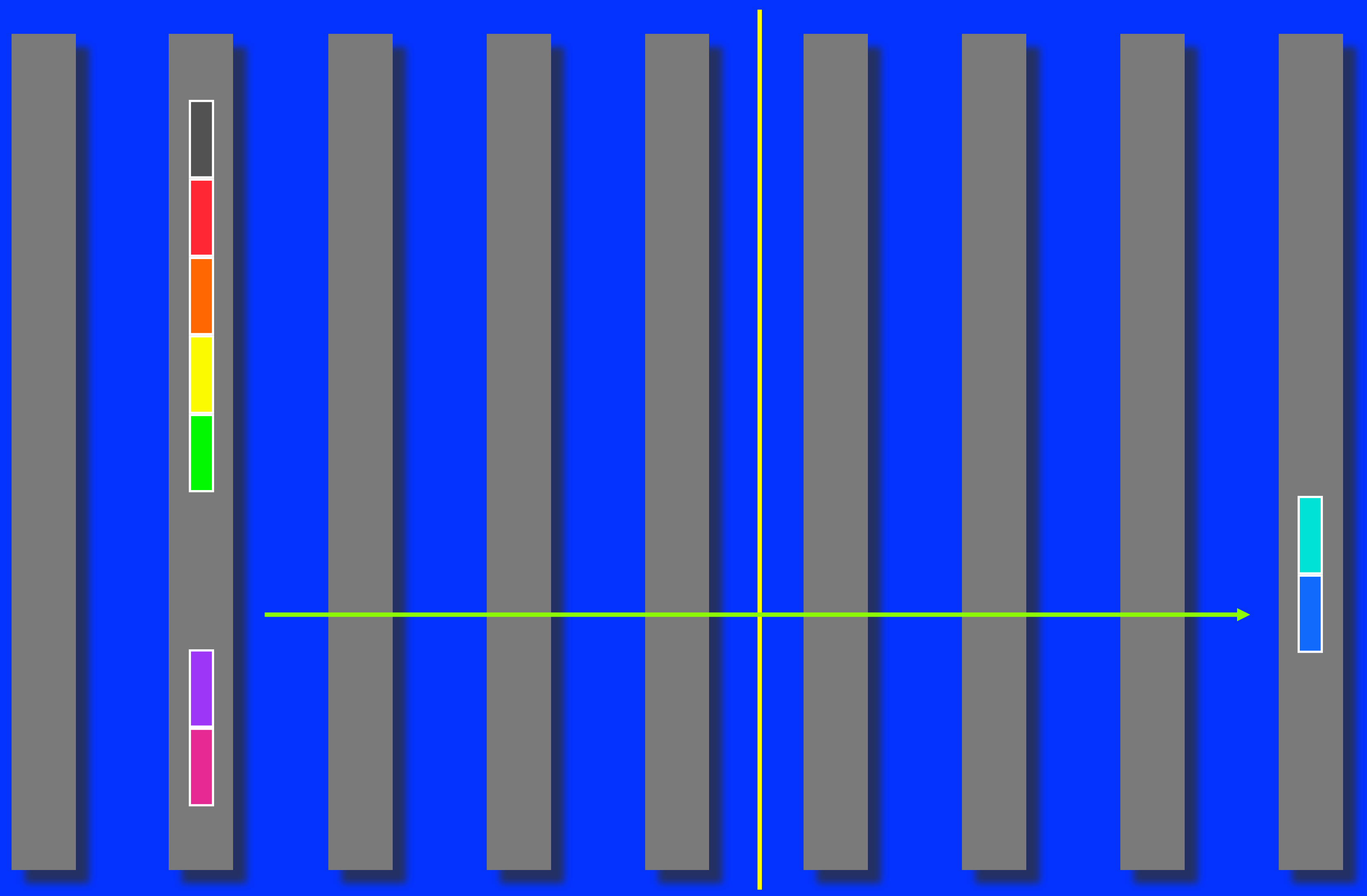
Scatter

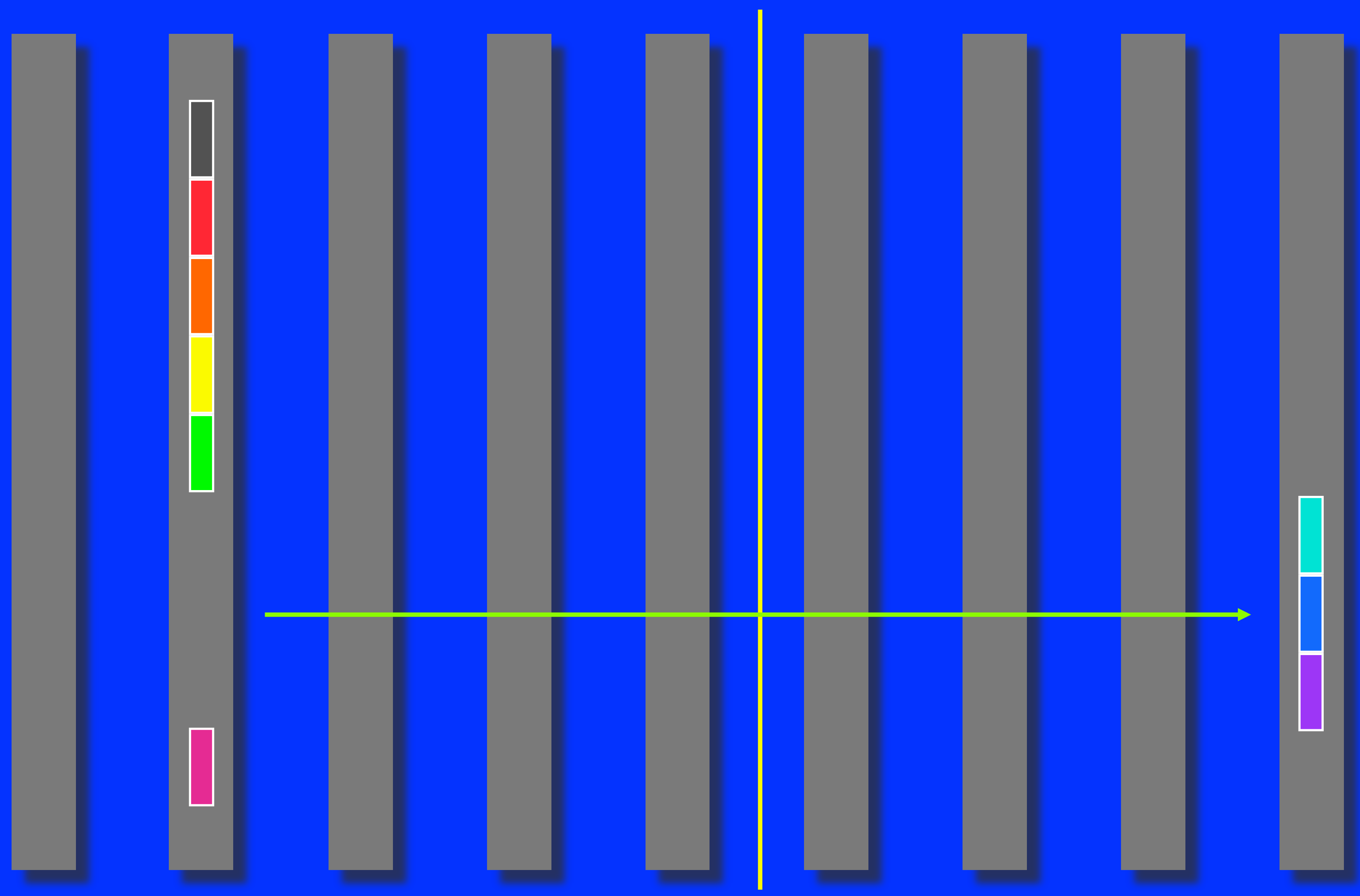


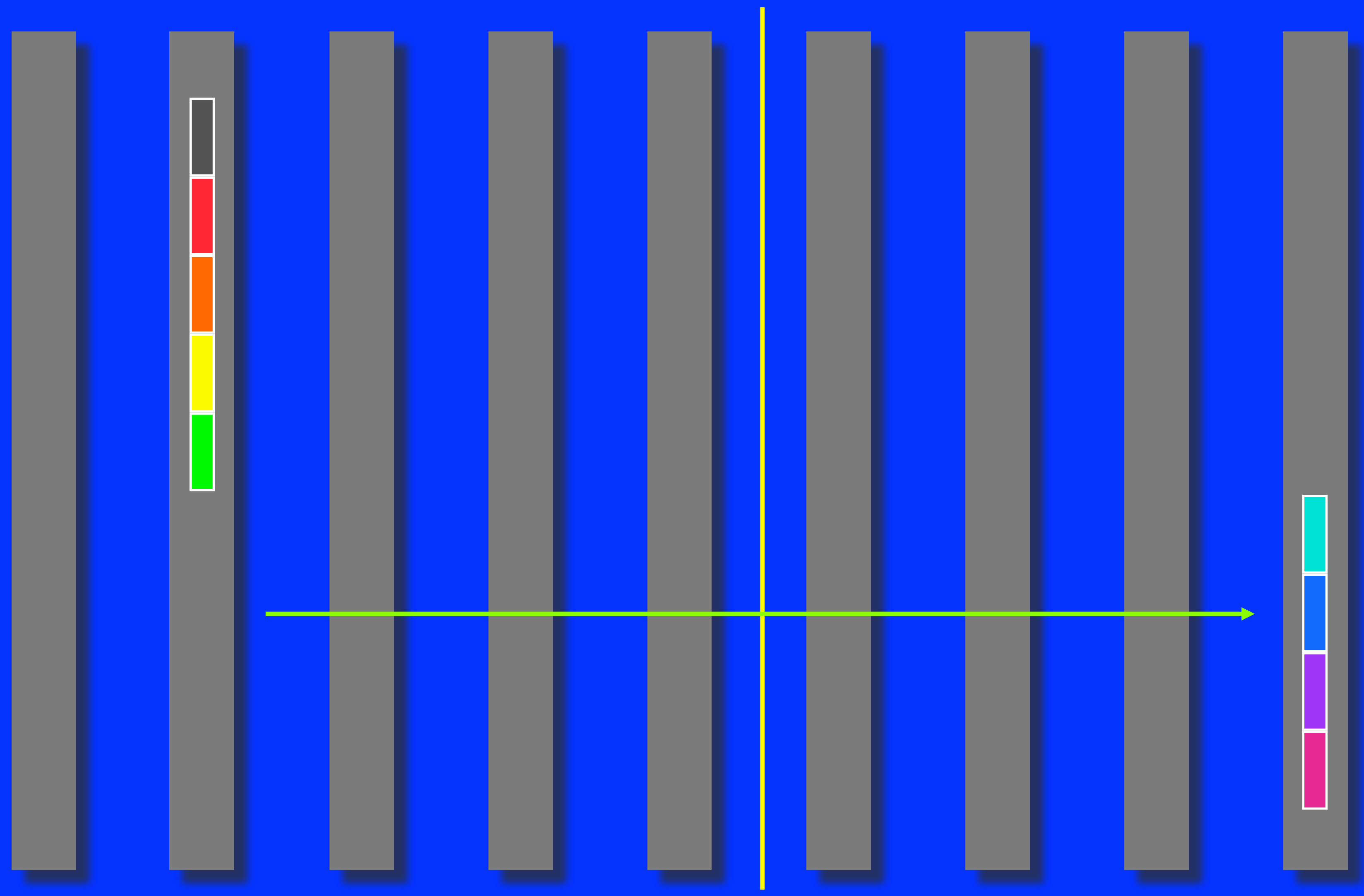


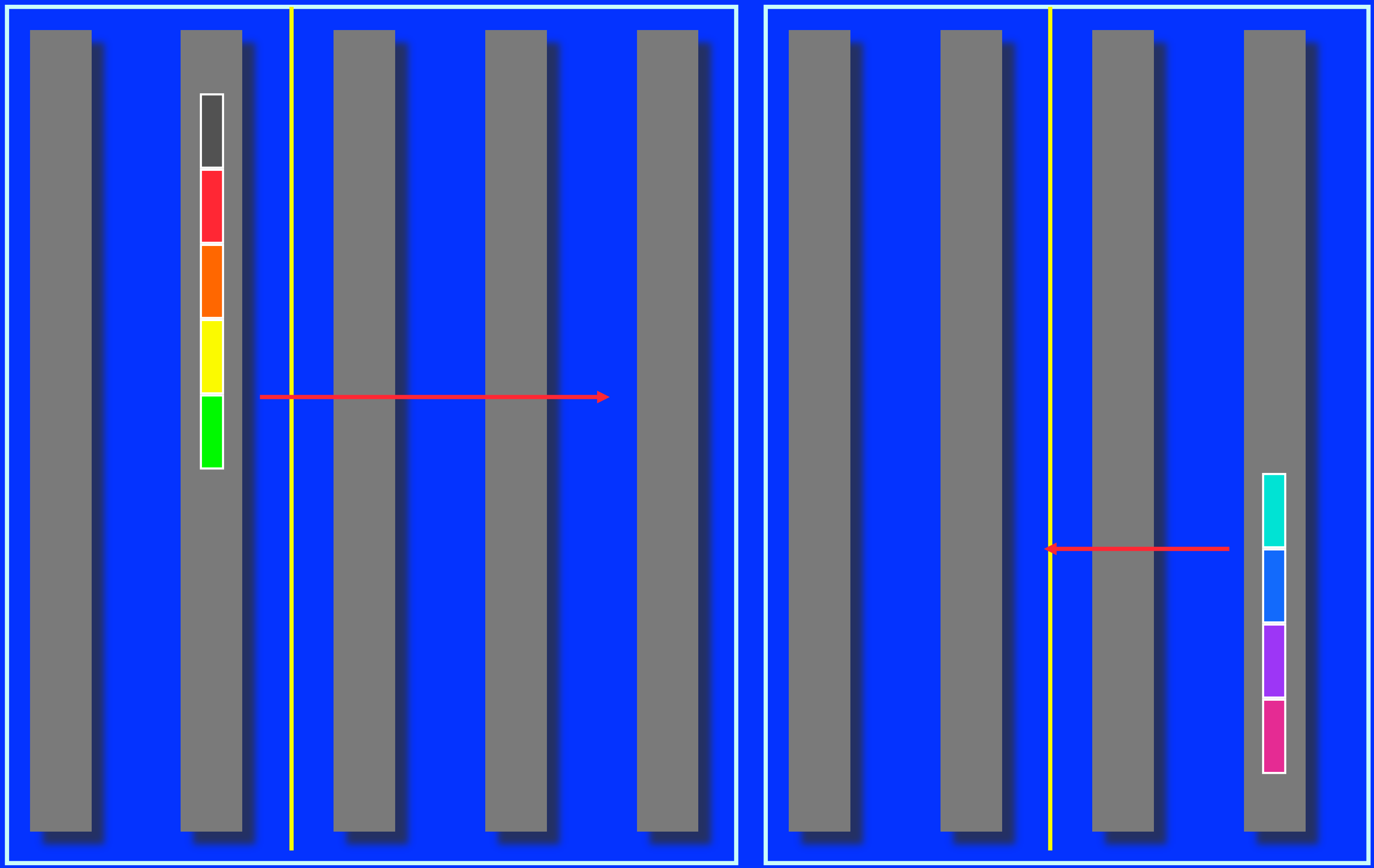


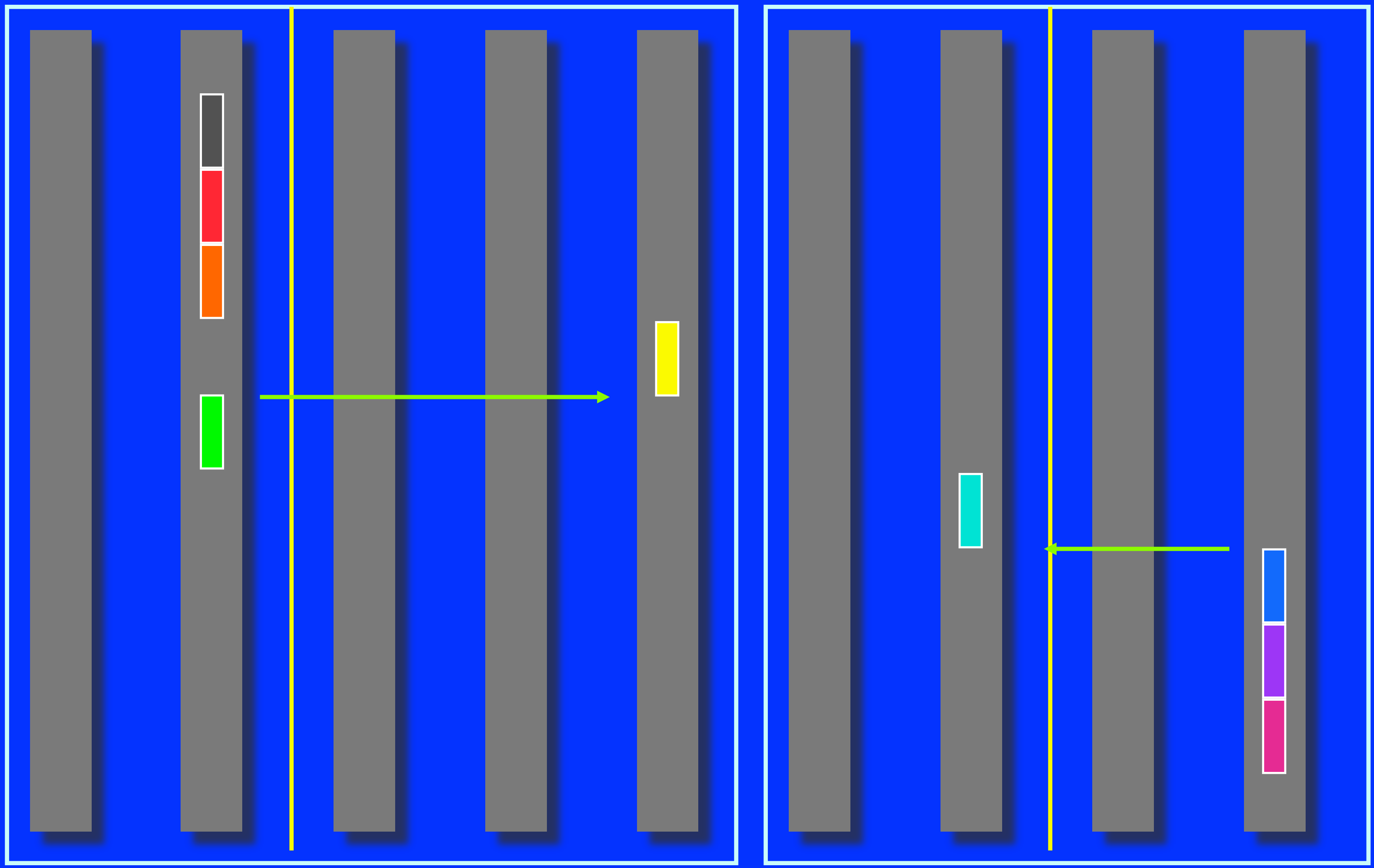


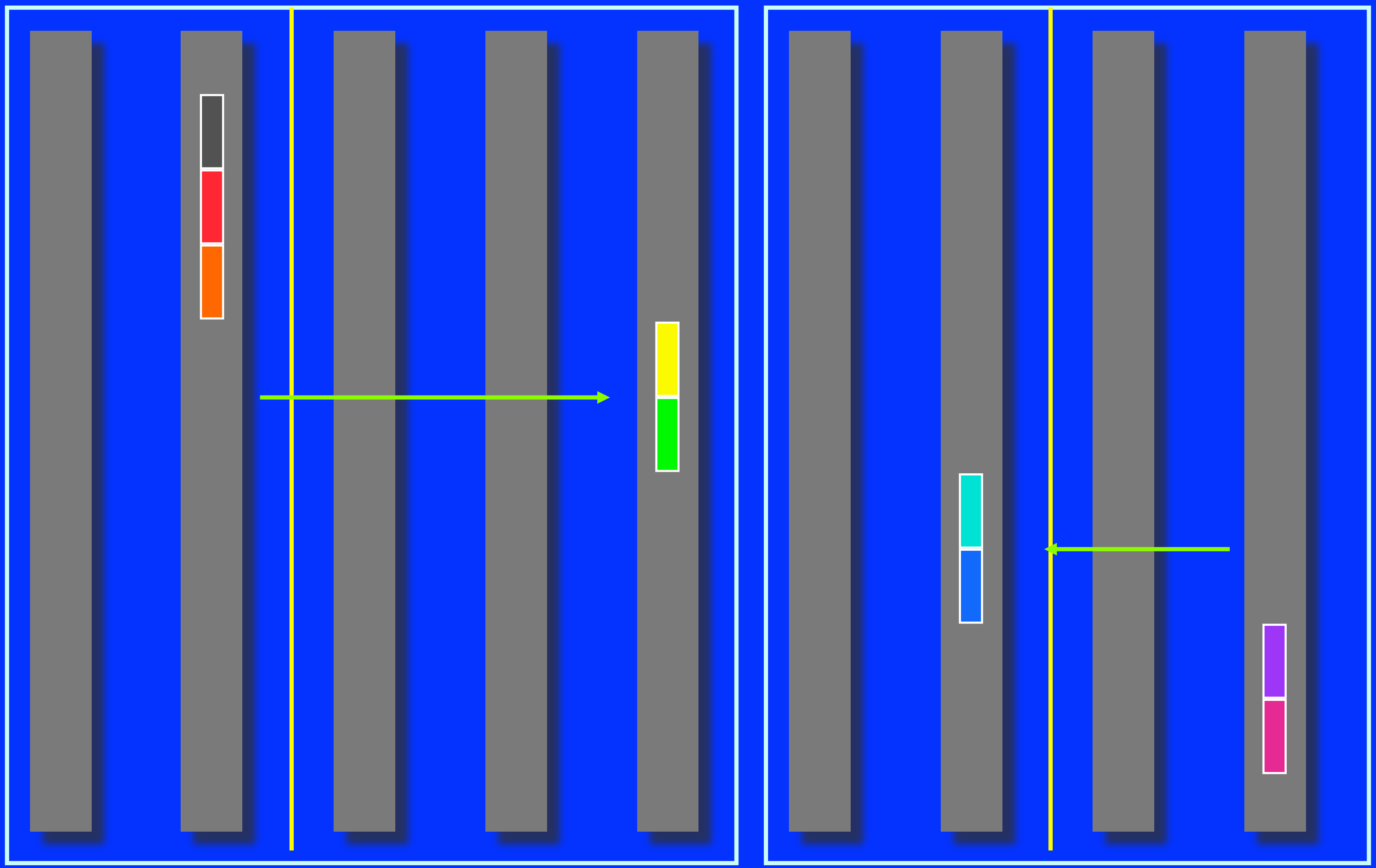


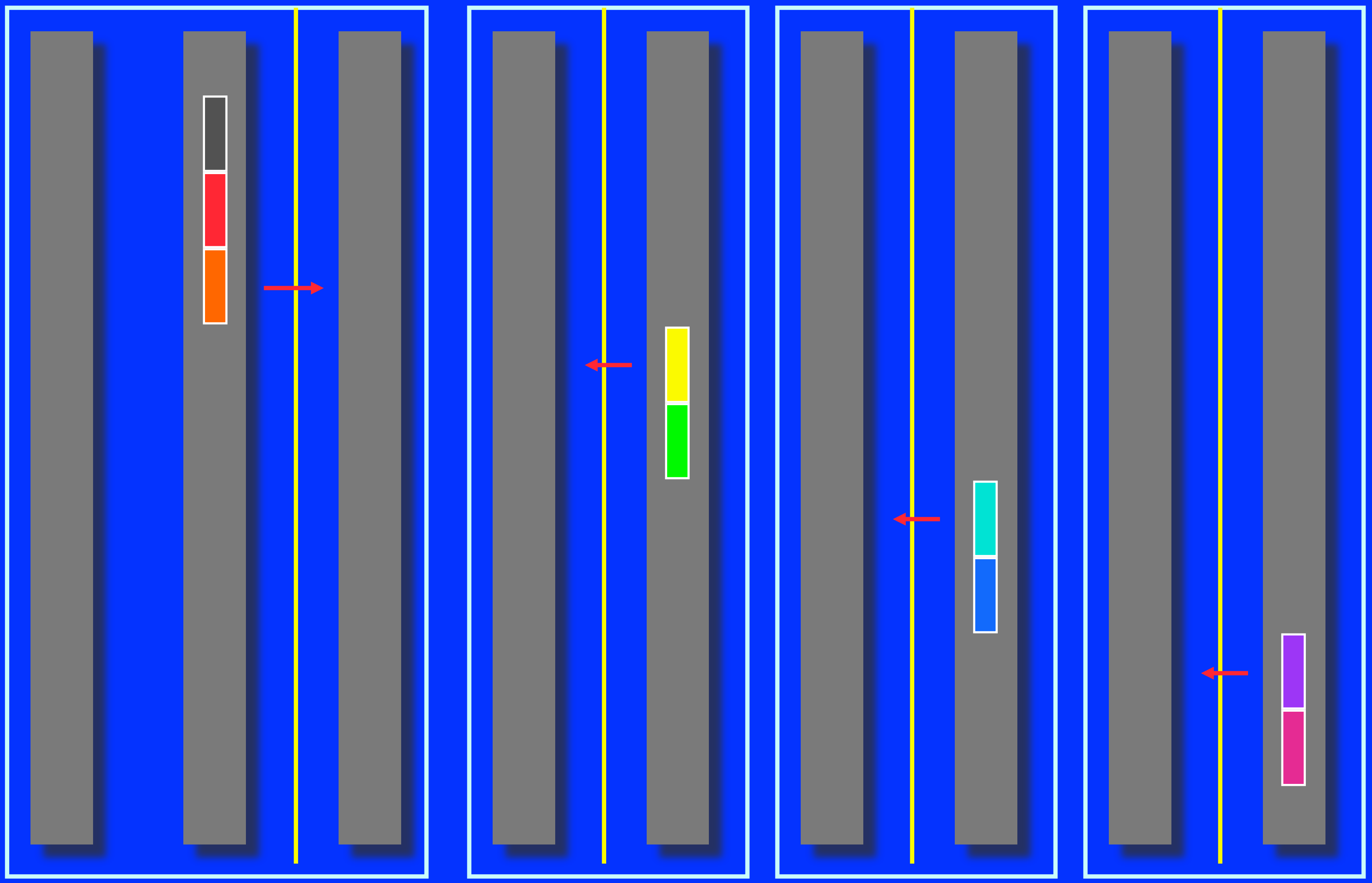


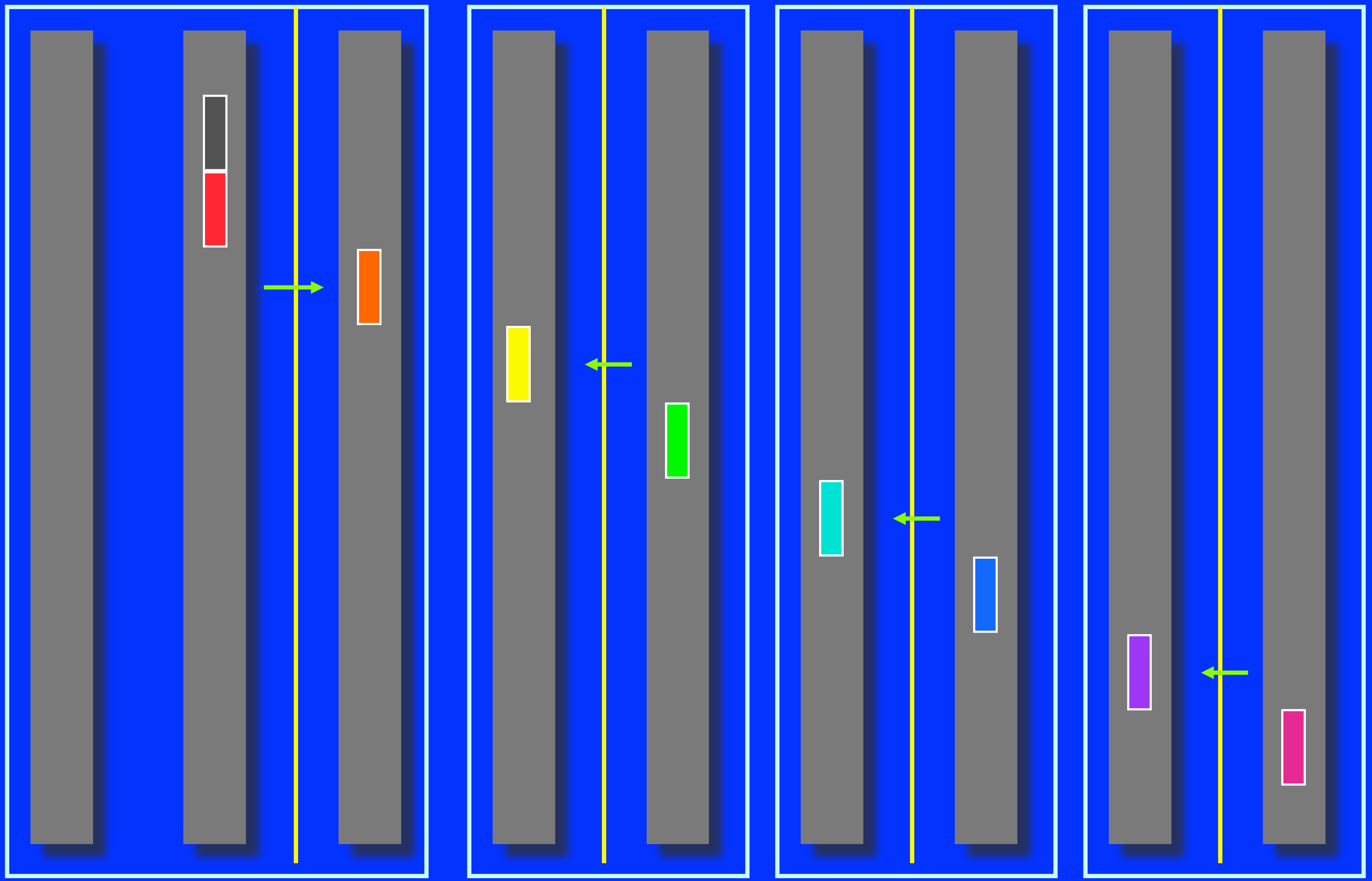


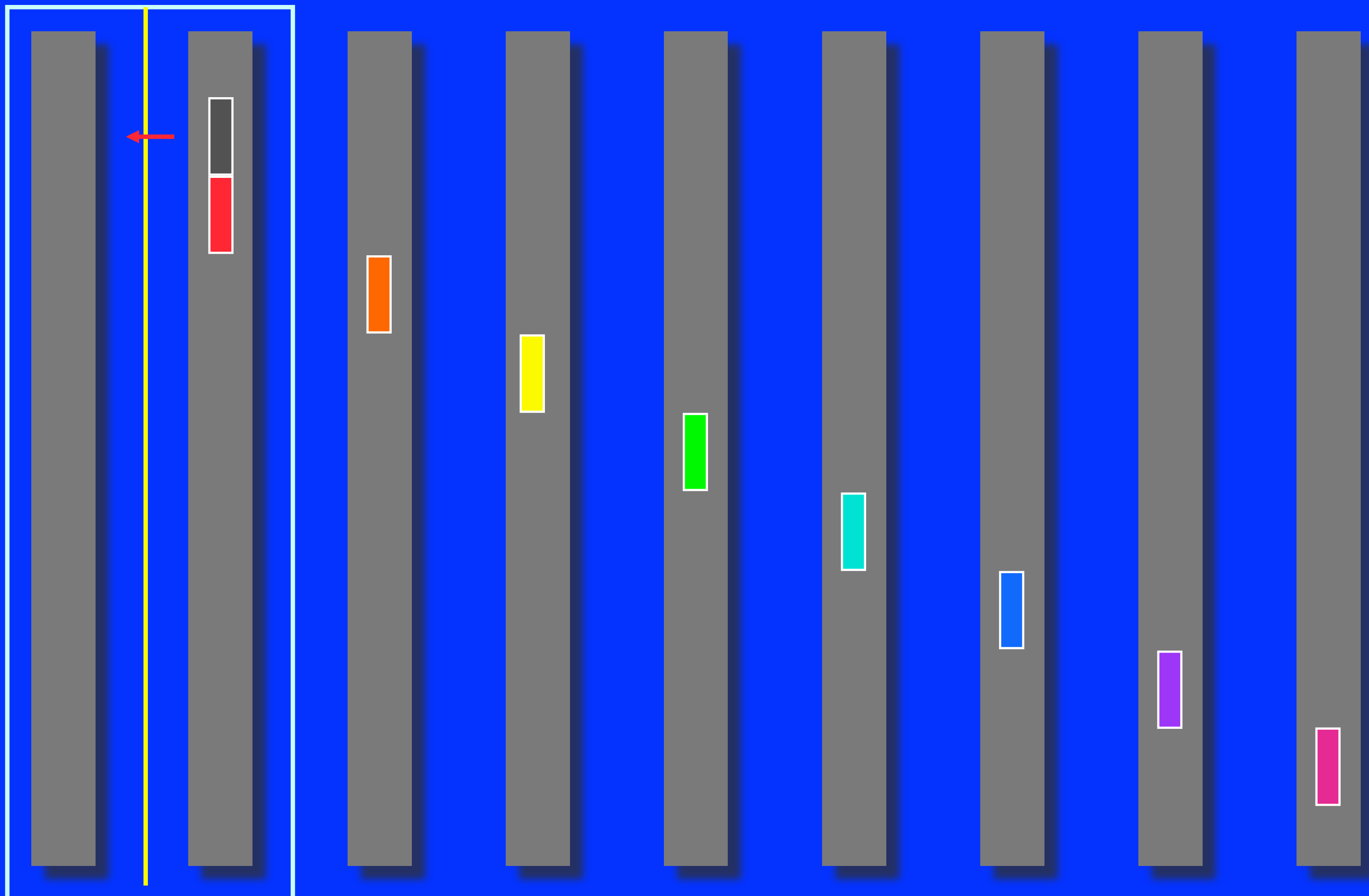


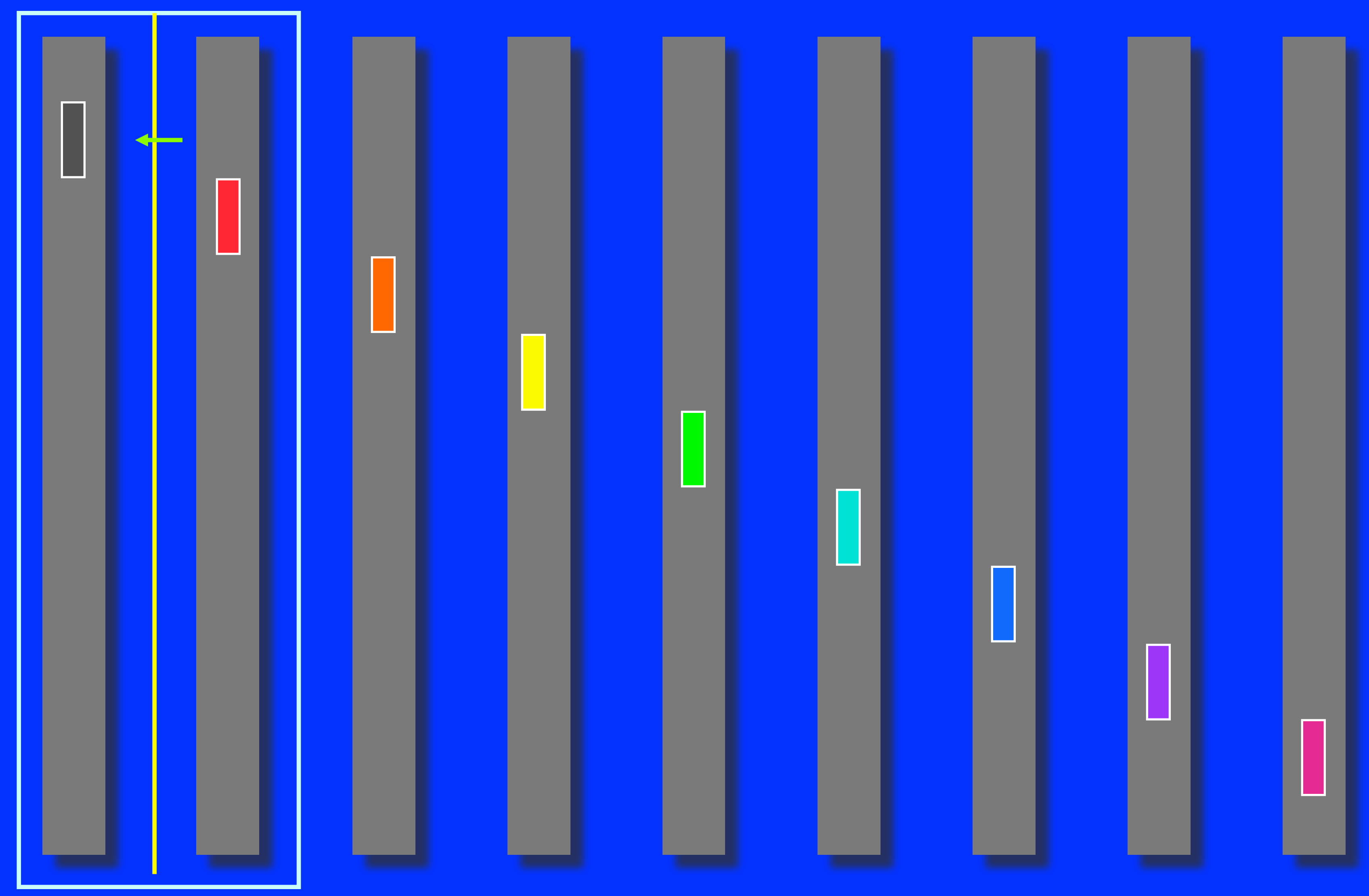


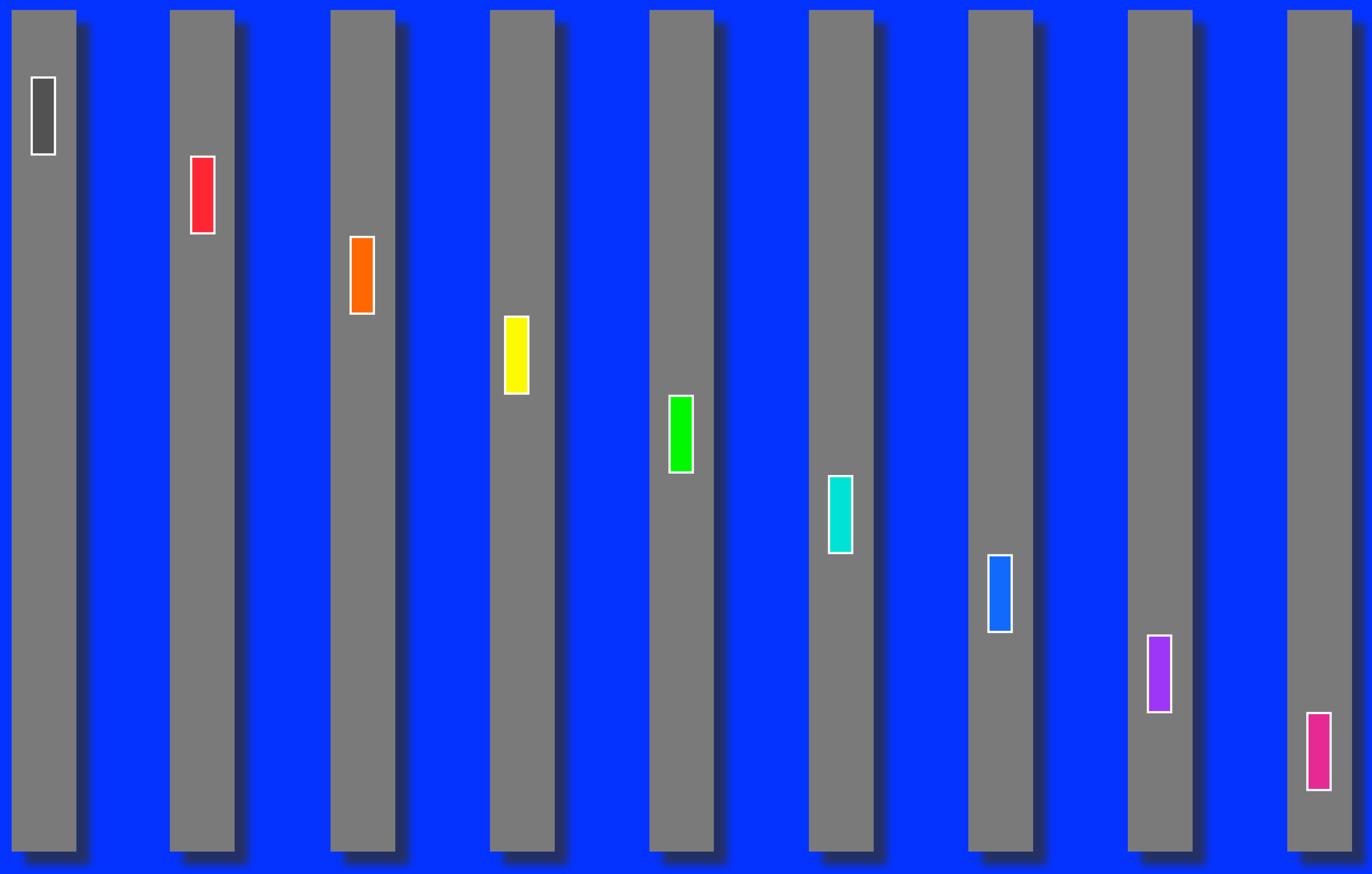












Cost of minimum spanning tree scatter

- Assumption: power of two number of nodes

$$\sum_{k=1}^{log(p)} \left(\alpha + \frac{n}{2^k} \beta \right)$$

$$=$$

$$log(p) \quad \alpha + \frac{p-1}{p} n\beta$$

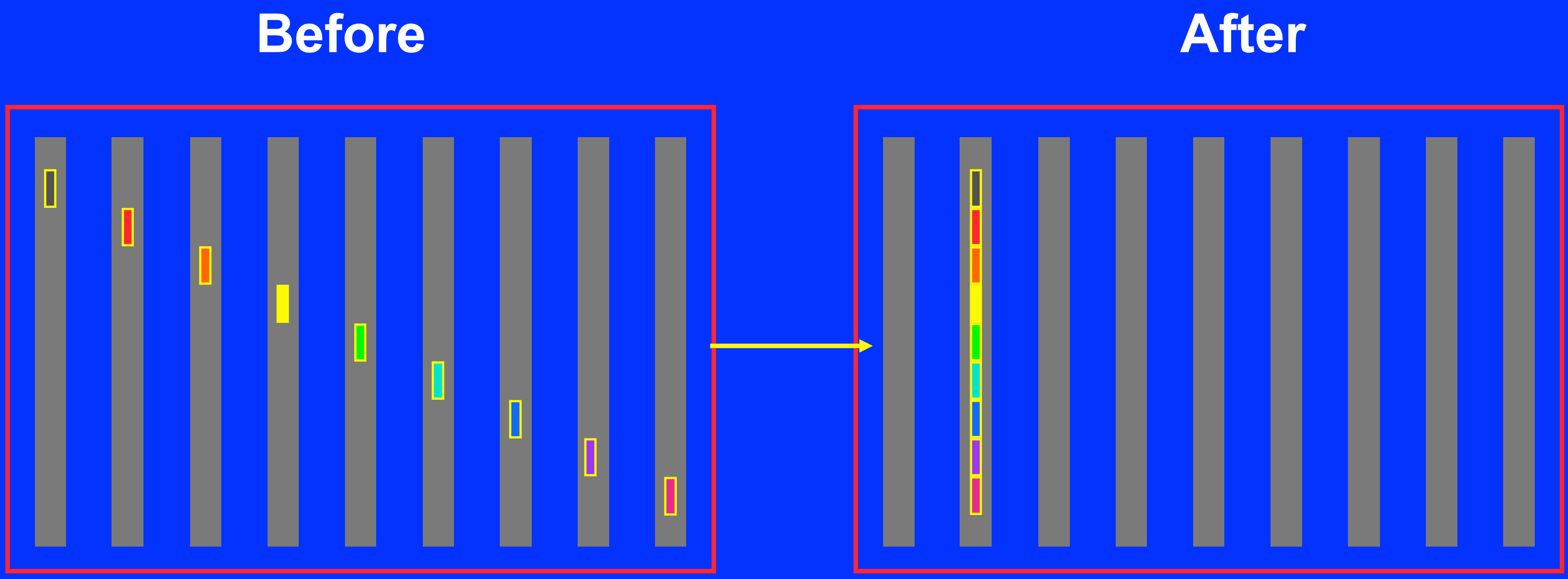
Cost of minimum spanning tree scatter

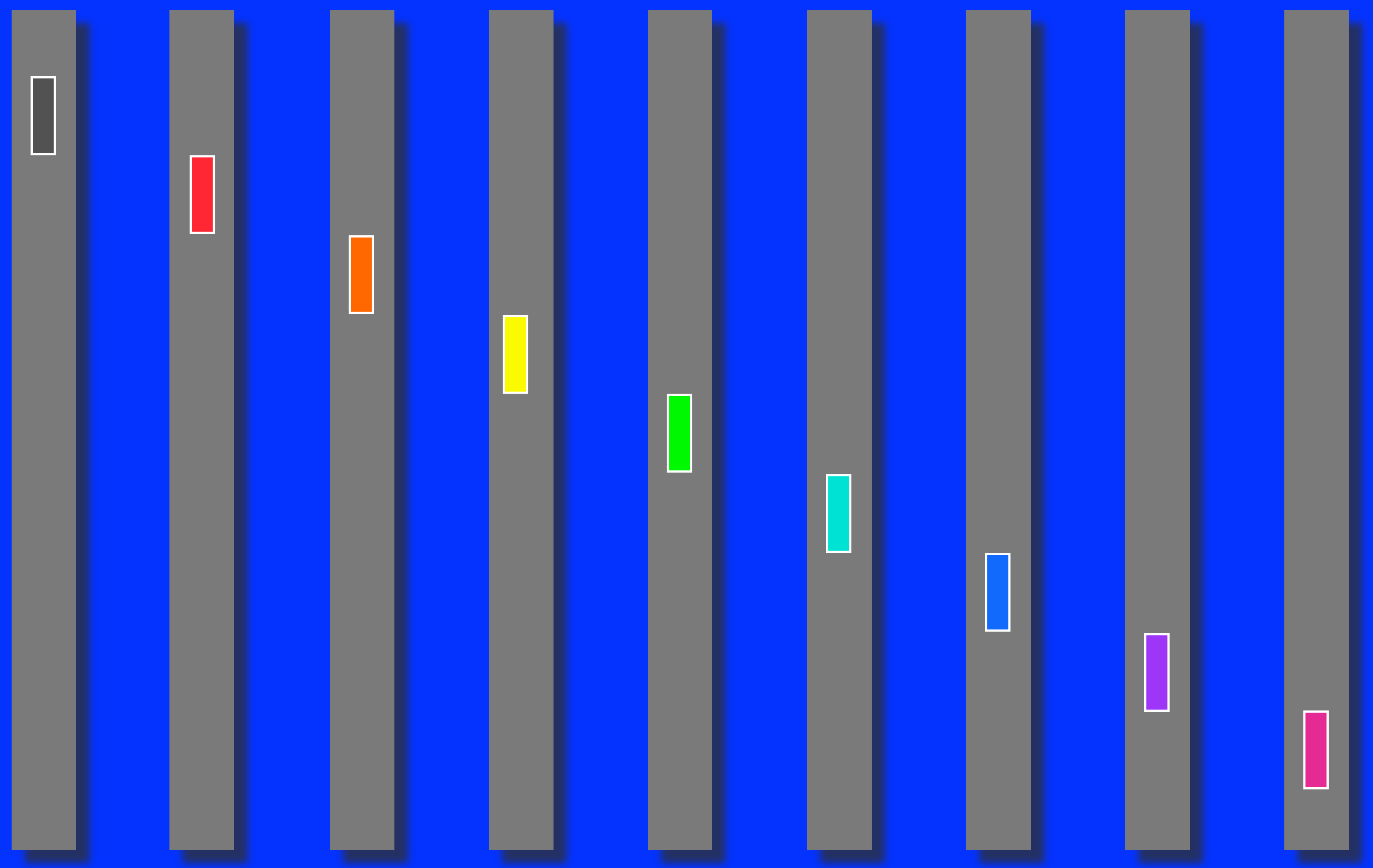
- **Assumption: power of two number of nodes**

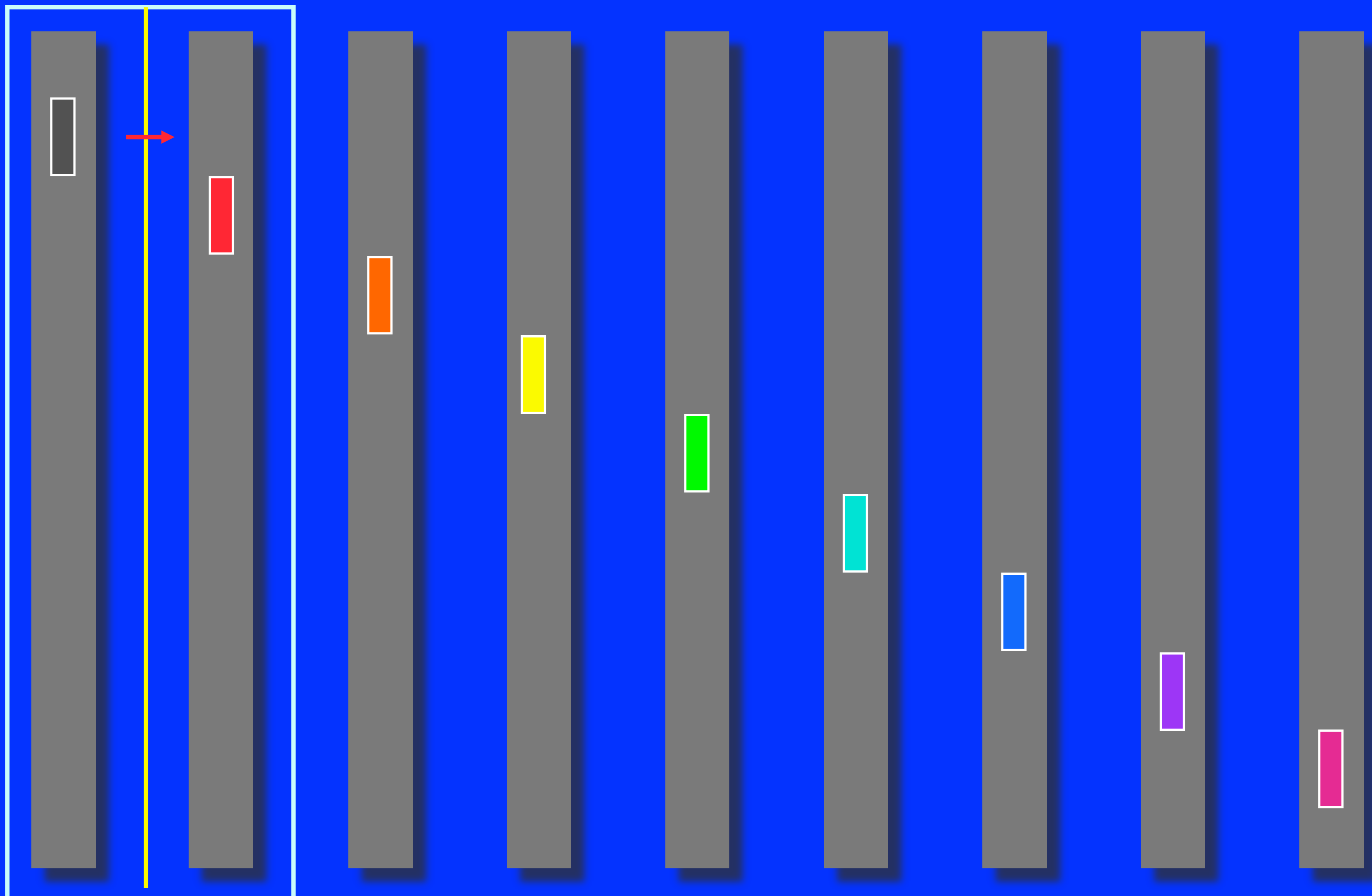
$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right) = \log(p) \; \alpha + \frac{p-1}{p} n\beta$$

Notice: attains lower bound for latency and bandwidth components

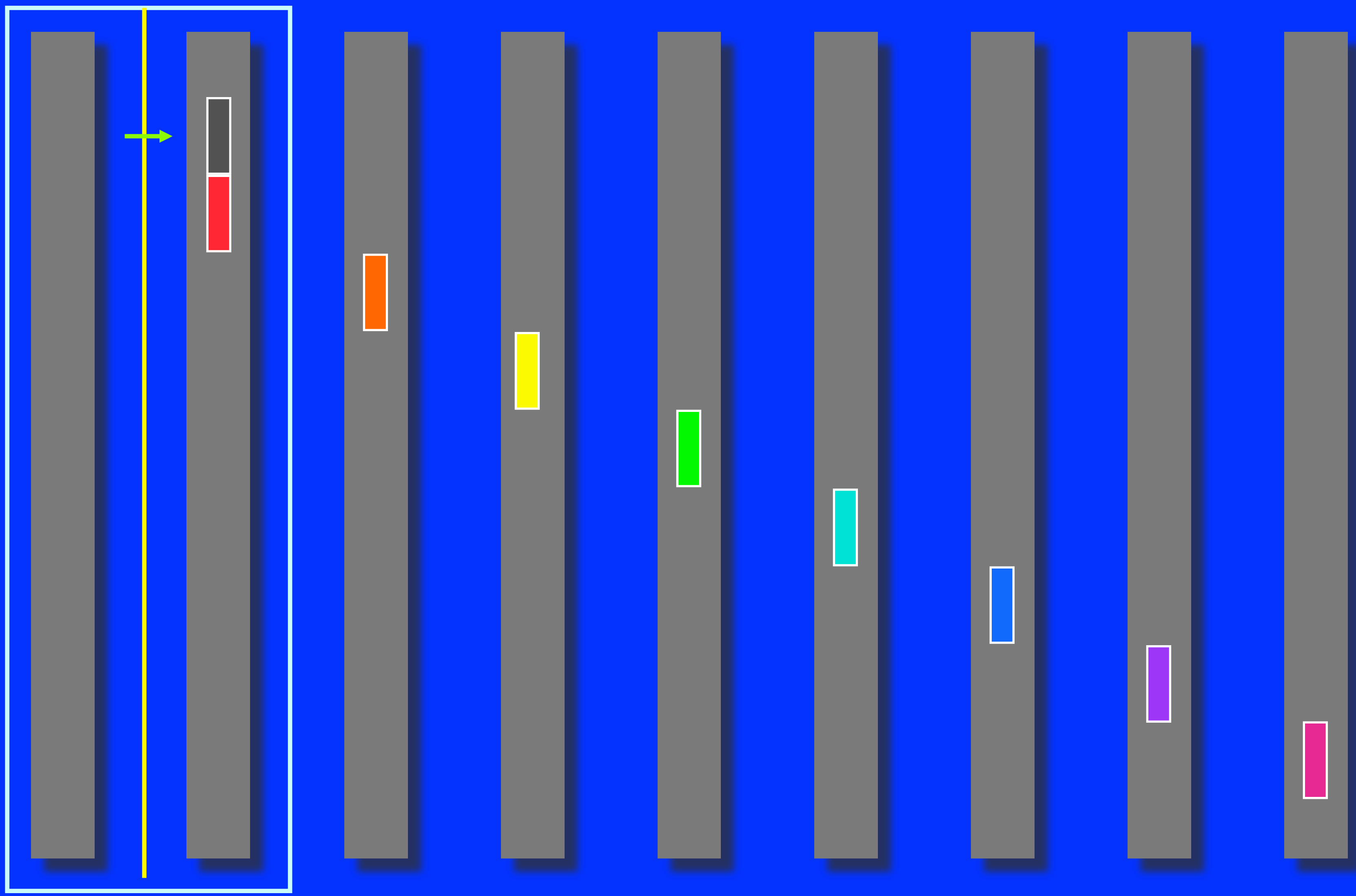
Gather

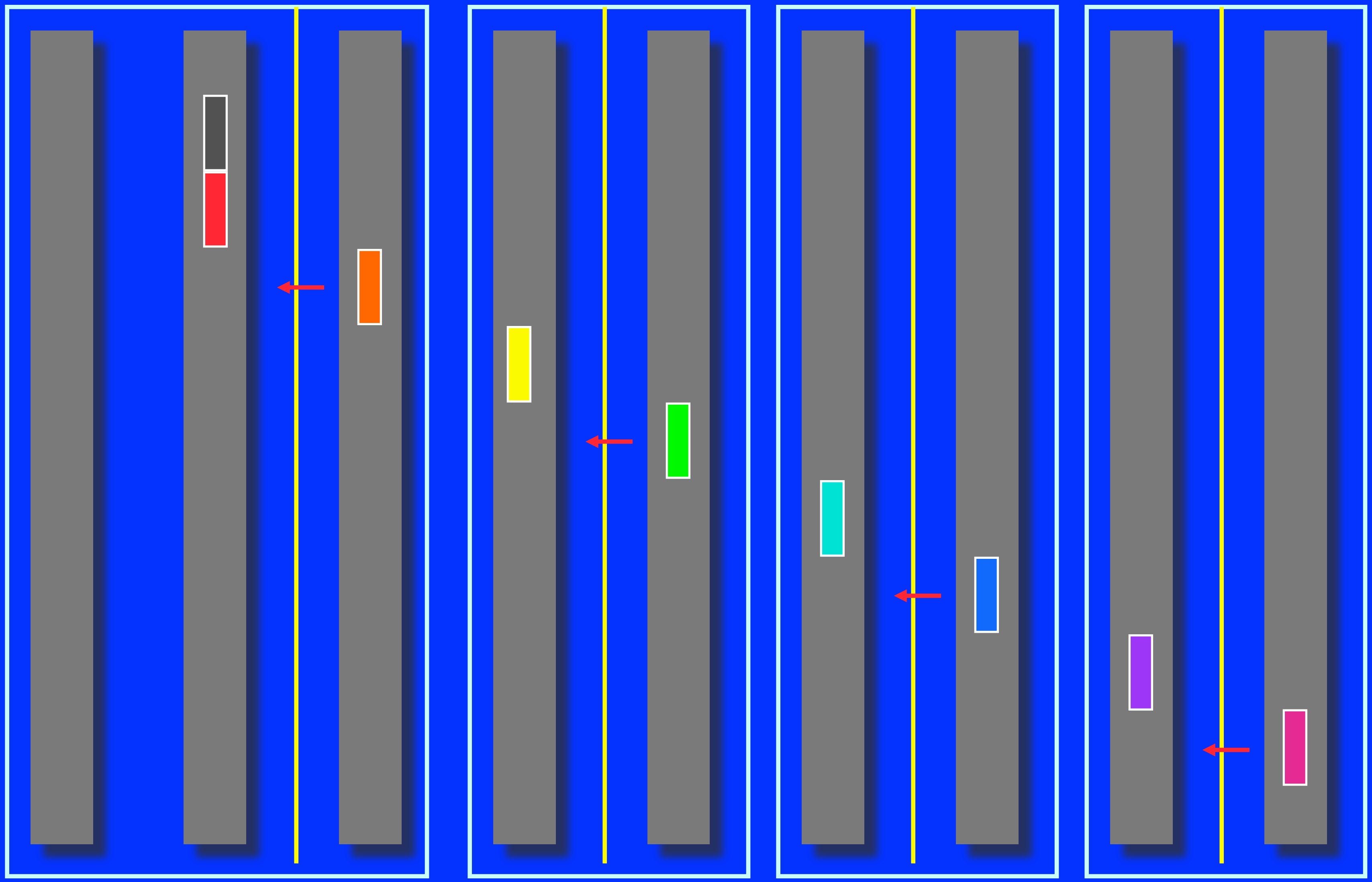


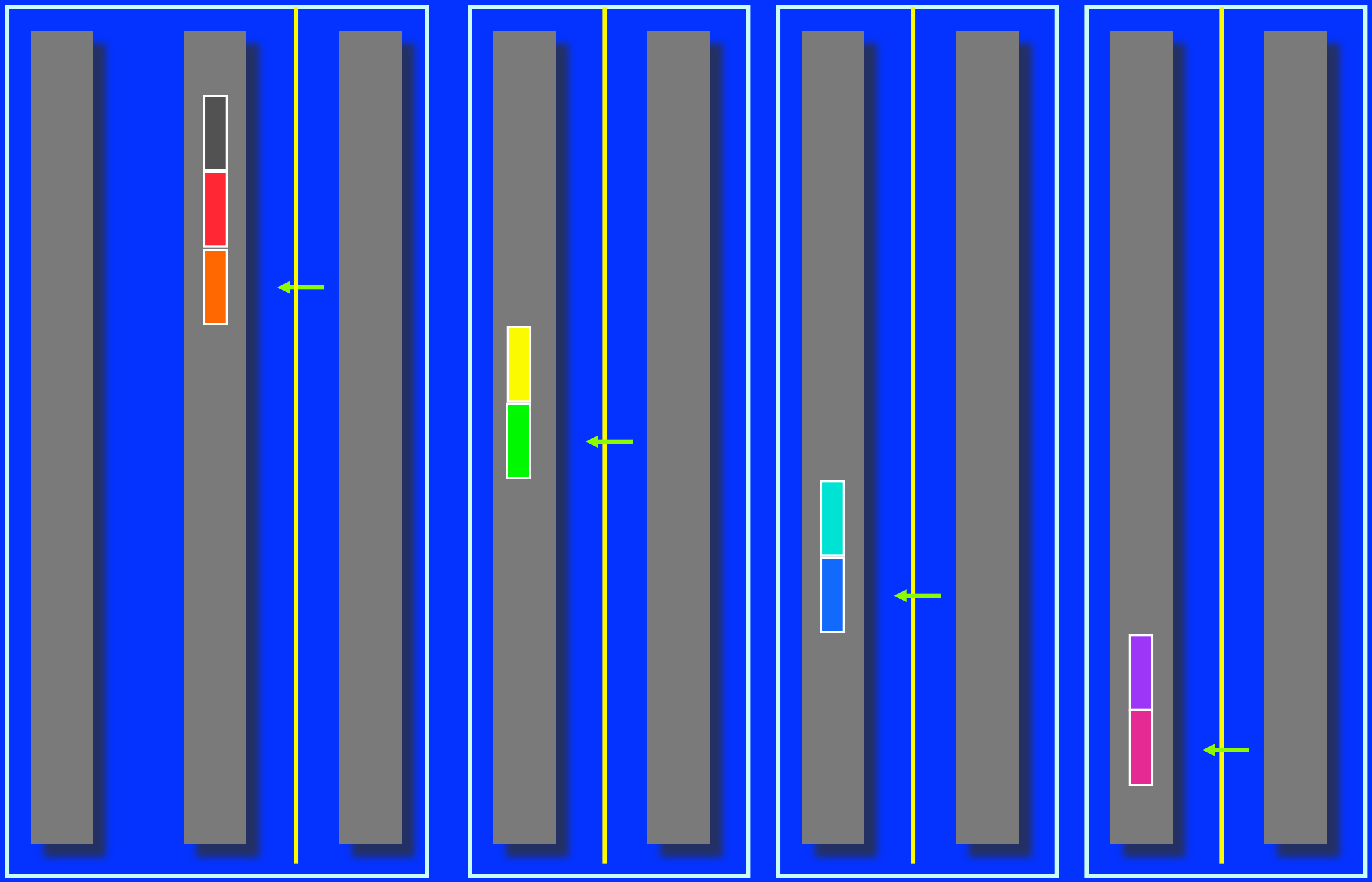


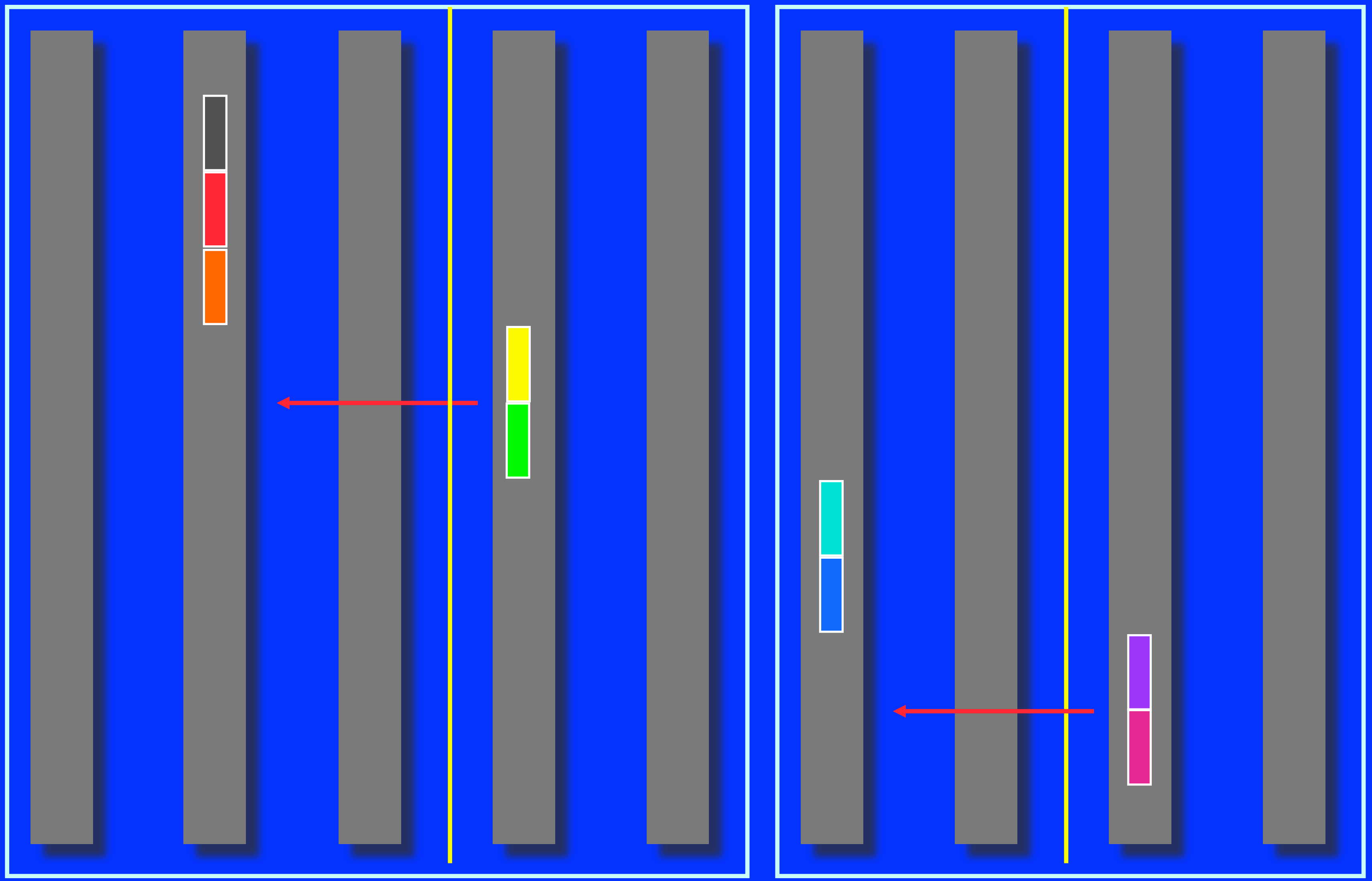


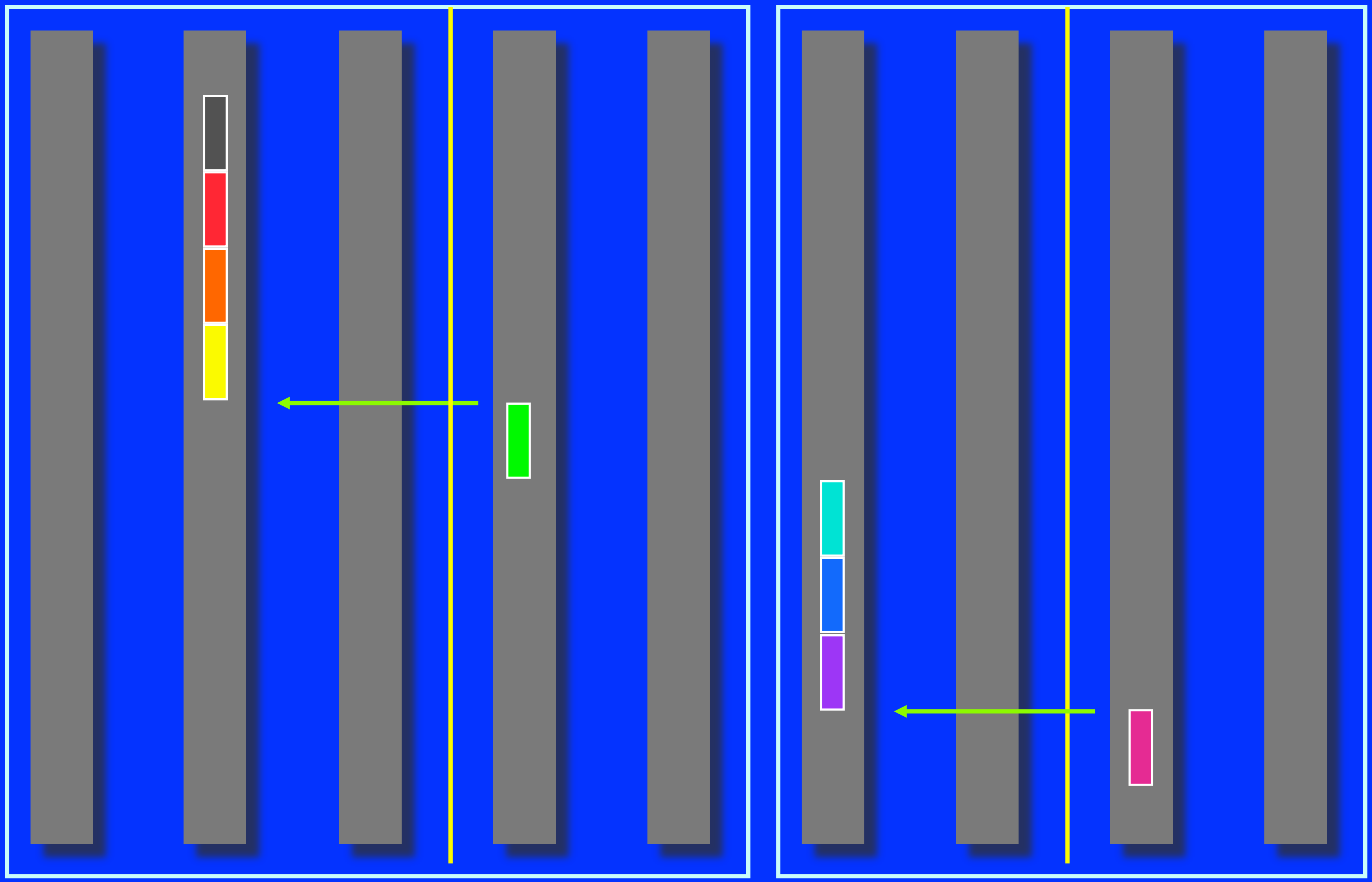
204

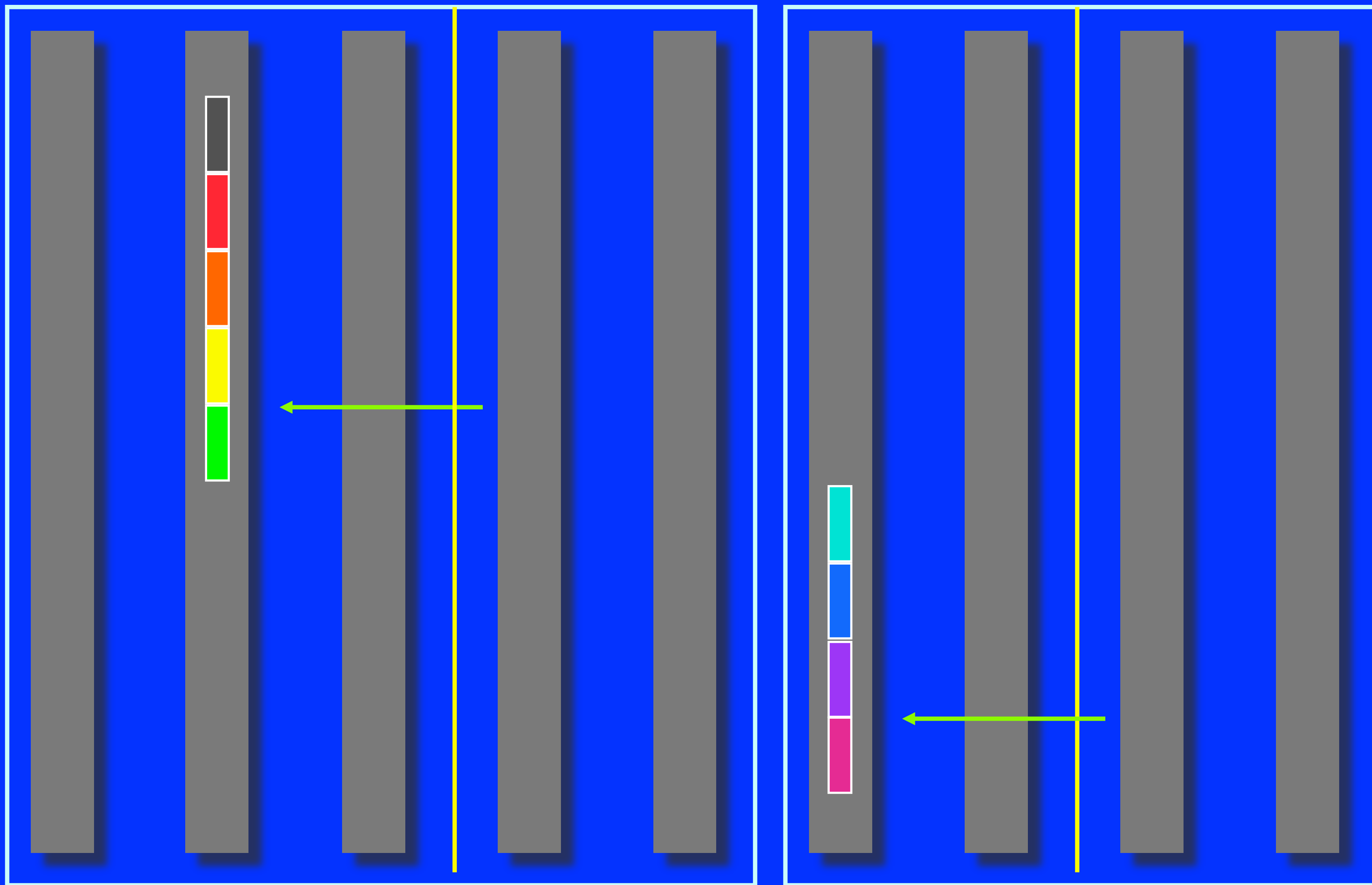


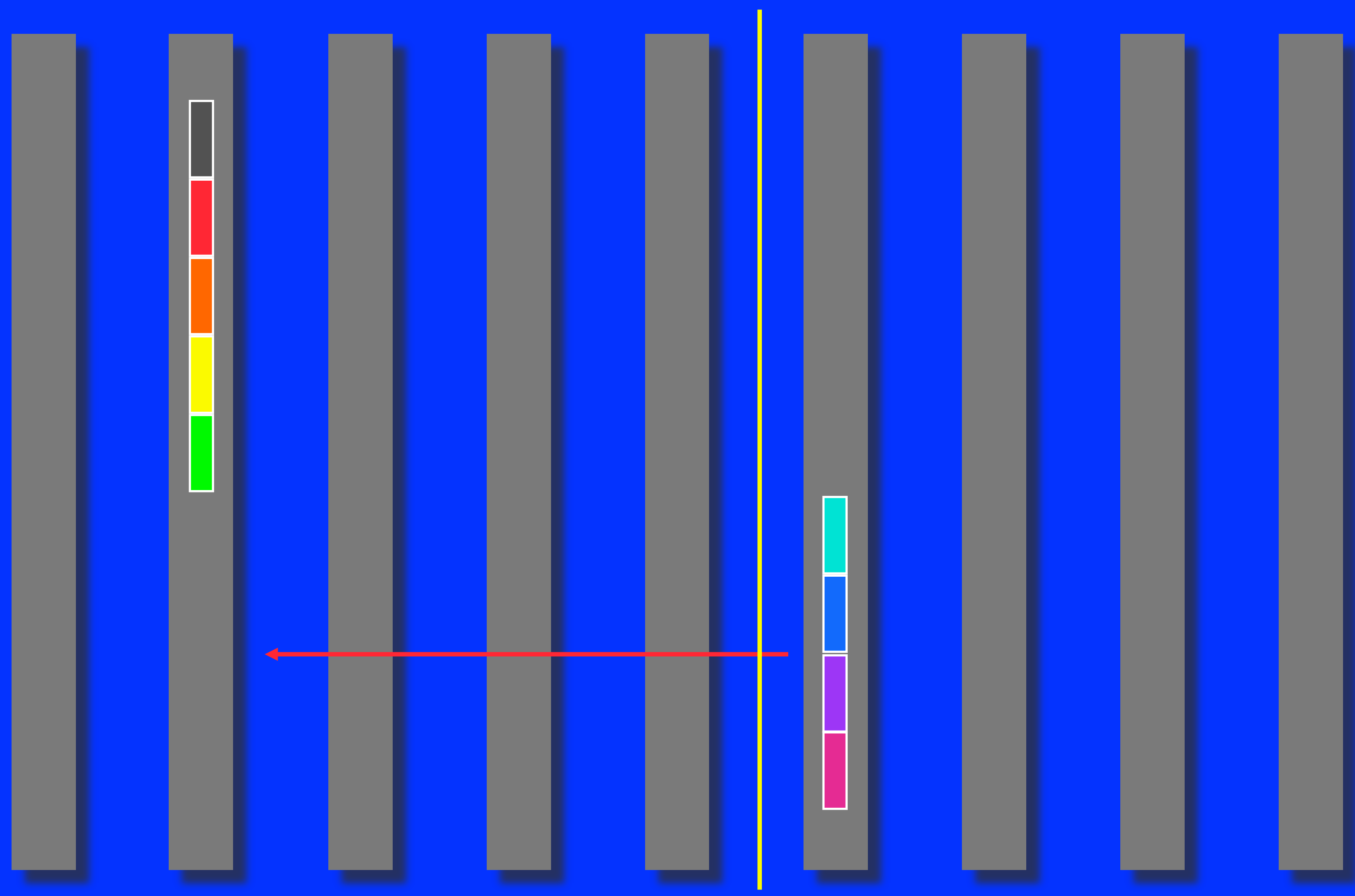


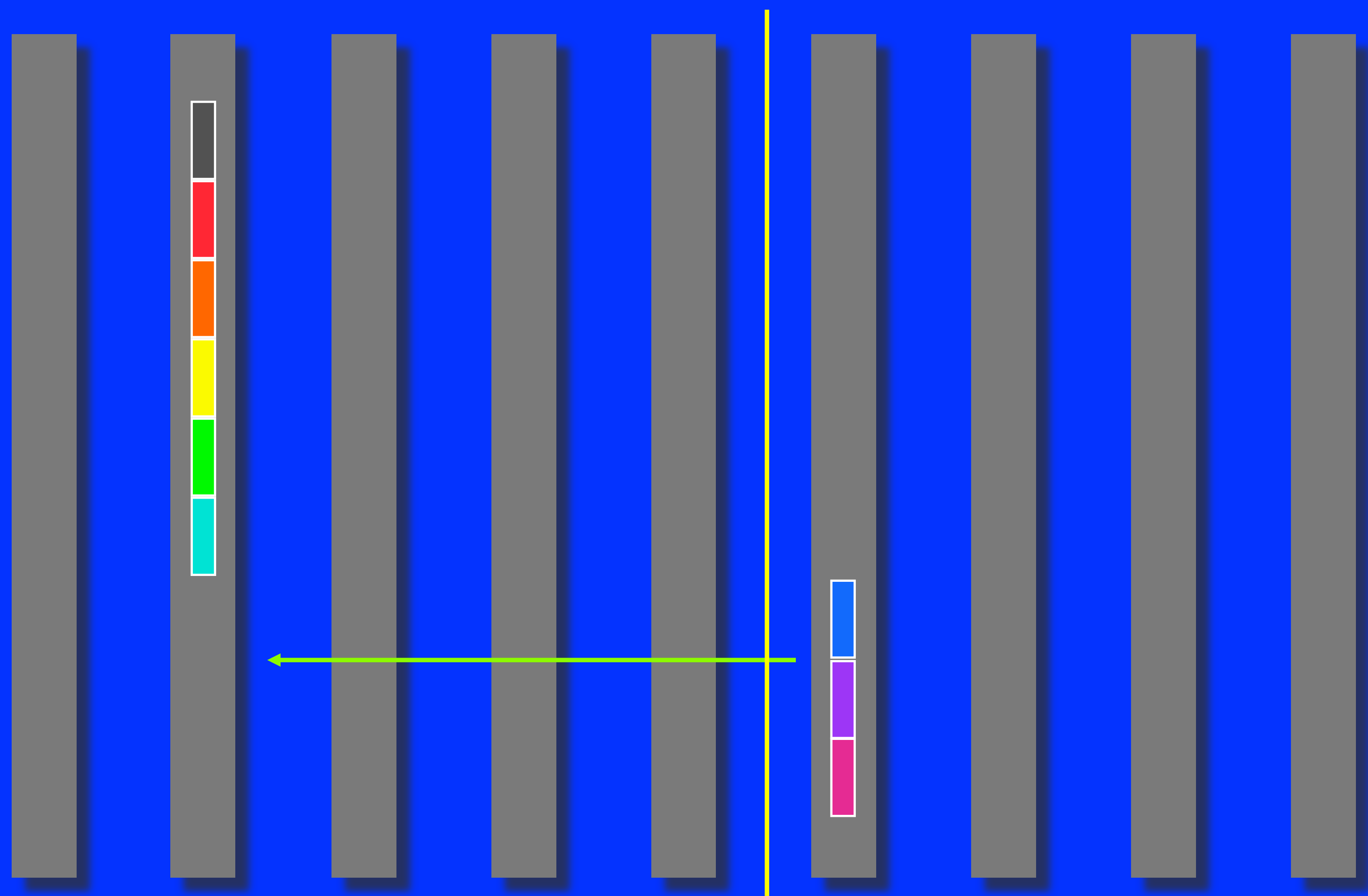


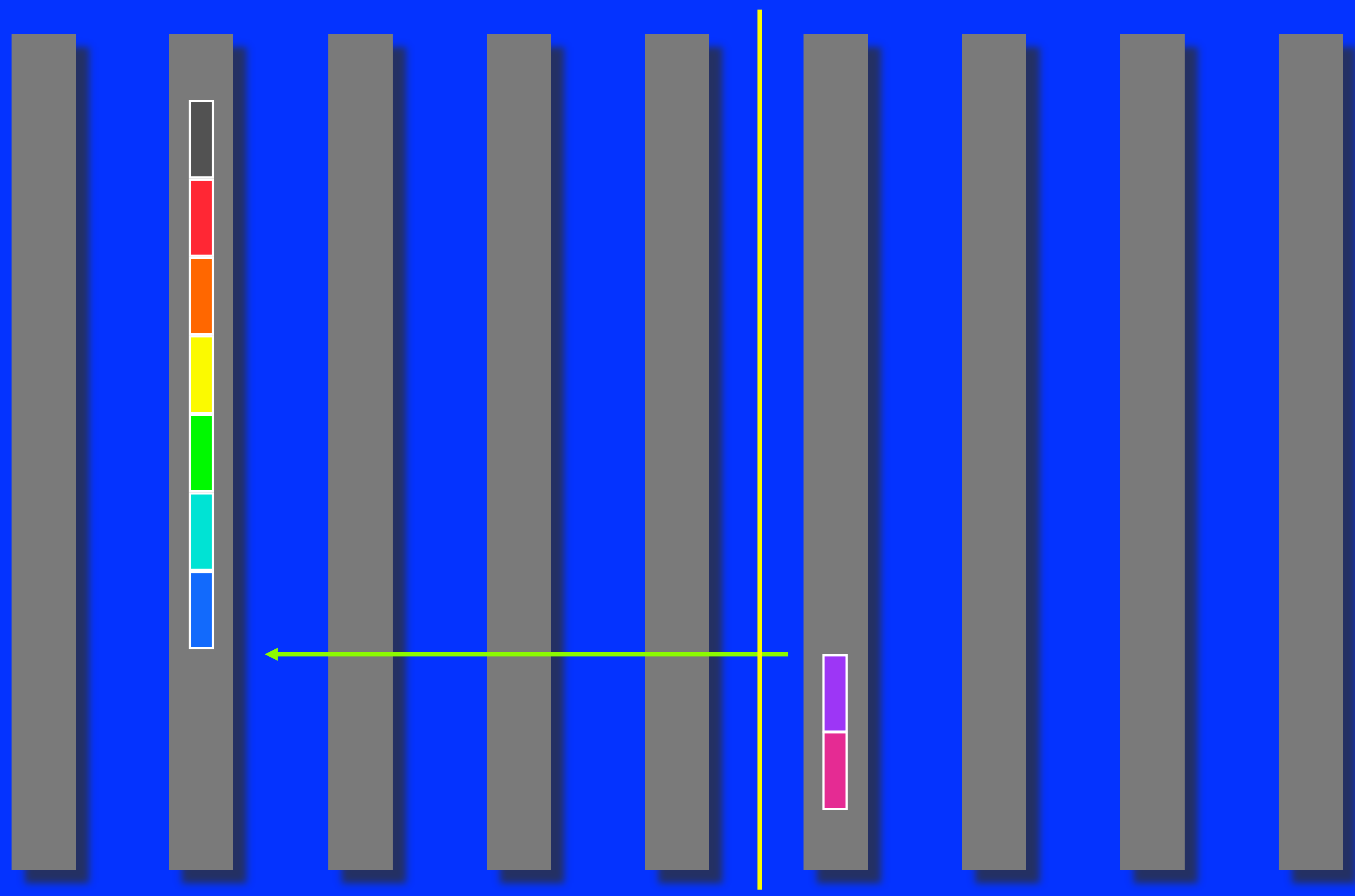


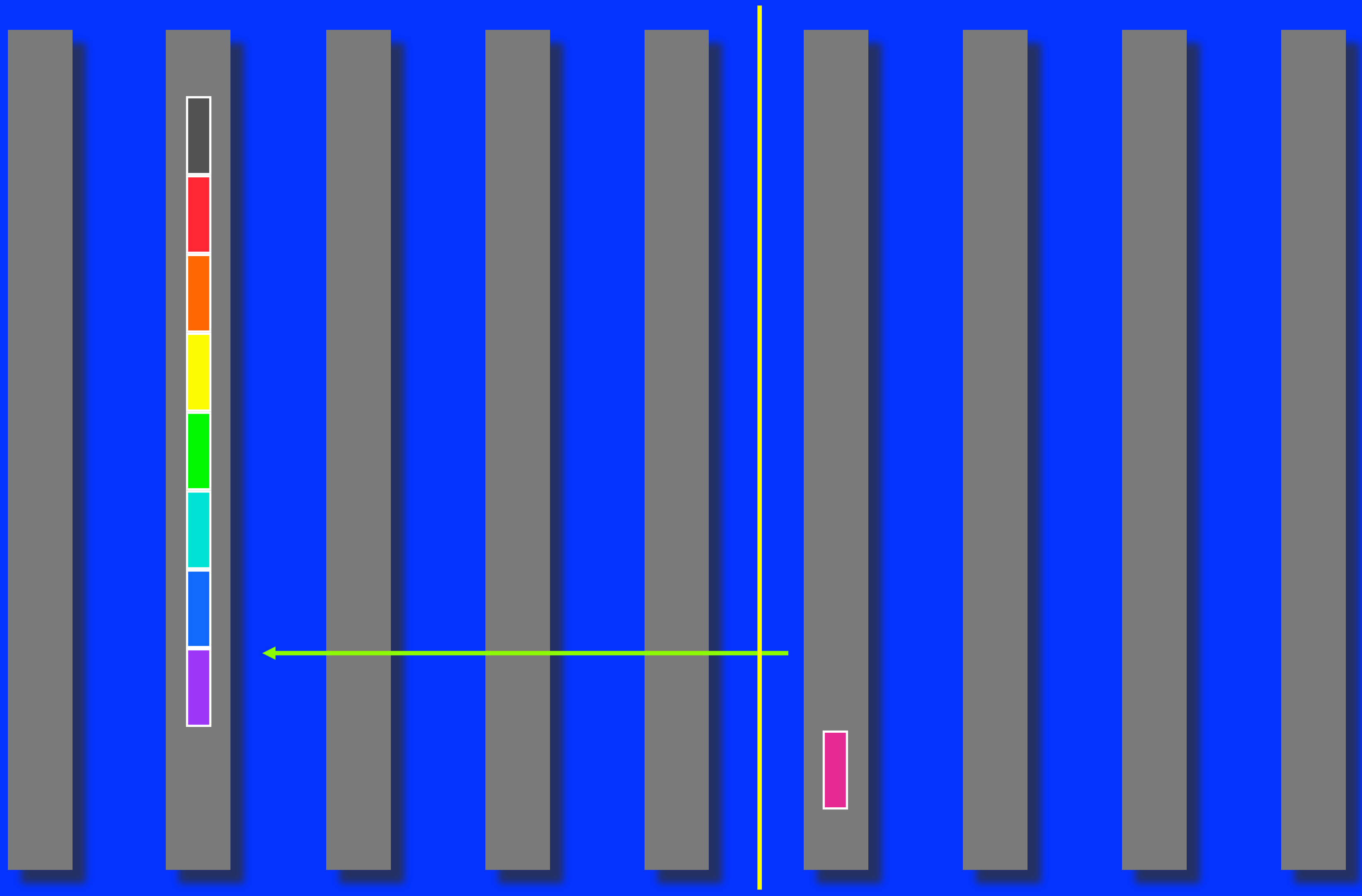


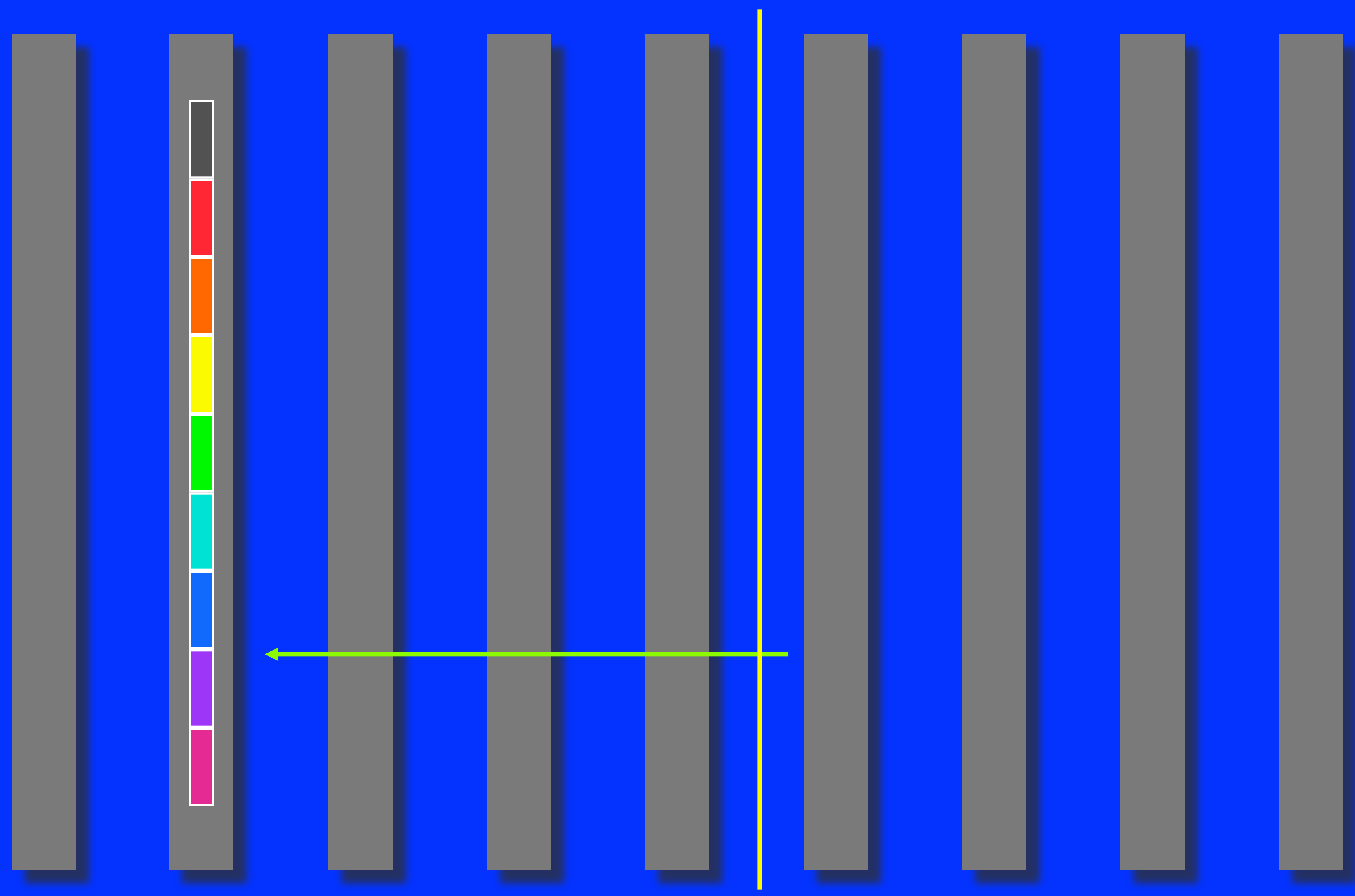


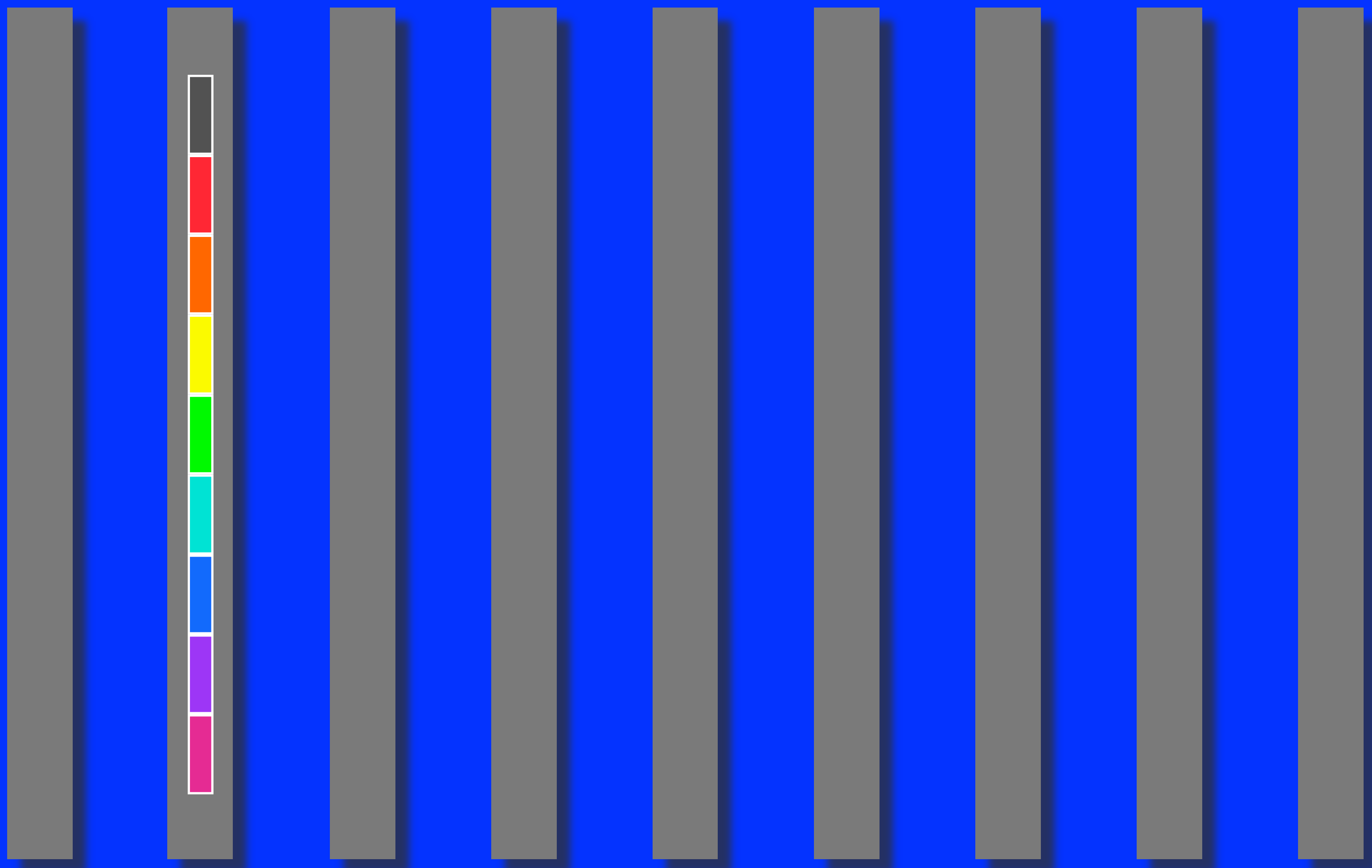












Cost of minimum spanning tree gather

- **Assumption: power of two number of nodes**

$$\sum_{k=1}^{\log(p)} \left(\alpha + \frac{n}{2^k} \beta \right)$$

$$=$$

$$\log(p) \; \alpha + \frac{p-1}{p} n\beta$$

Cost of minimum spanning tree gather

- **Assumption: power of two number of nodes**

$$\sum_{k=1}^{log(p)} \left(\alpha + \frac{n}{2^k} \beta \right)$$

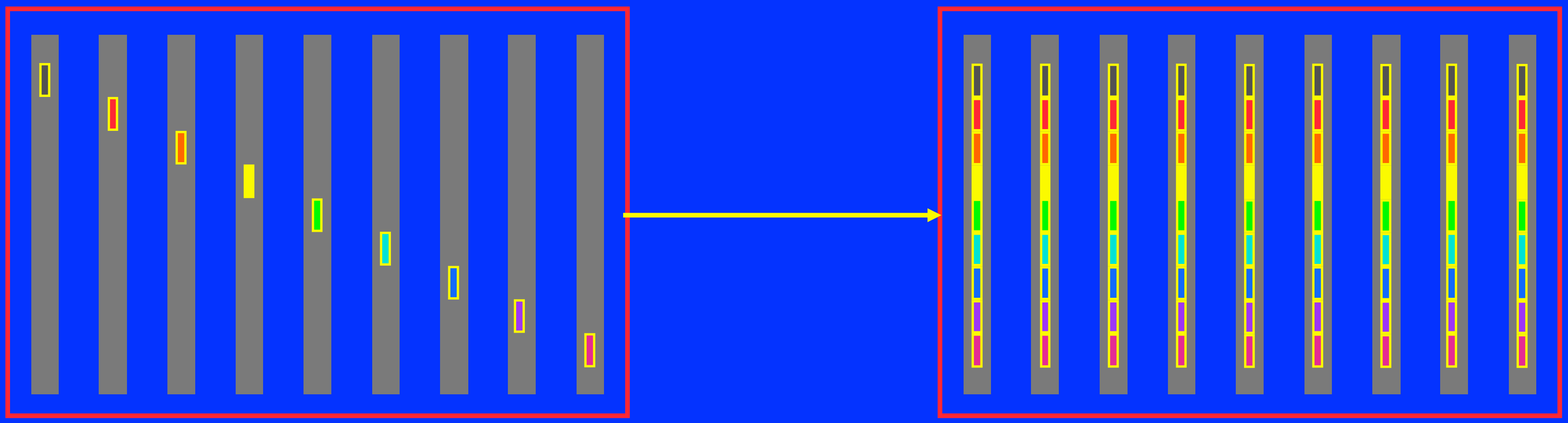
$$=$$

$$log(p) \quad \alpha + \frac{p-1}{p} n\beta$$

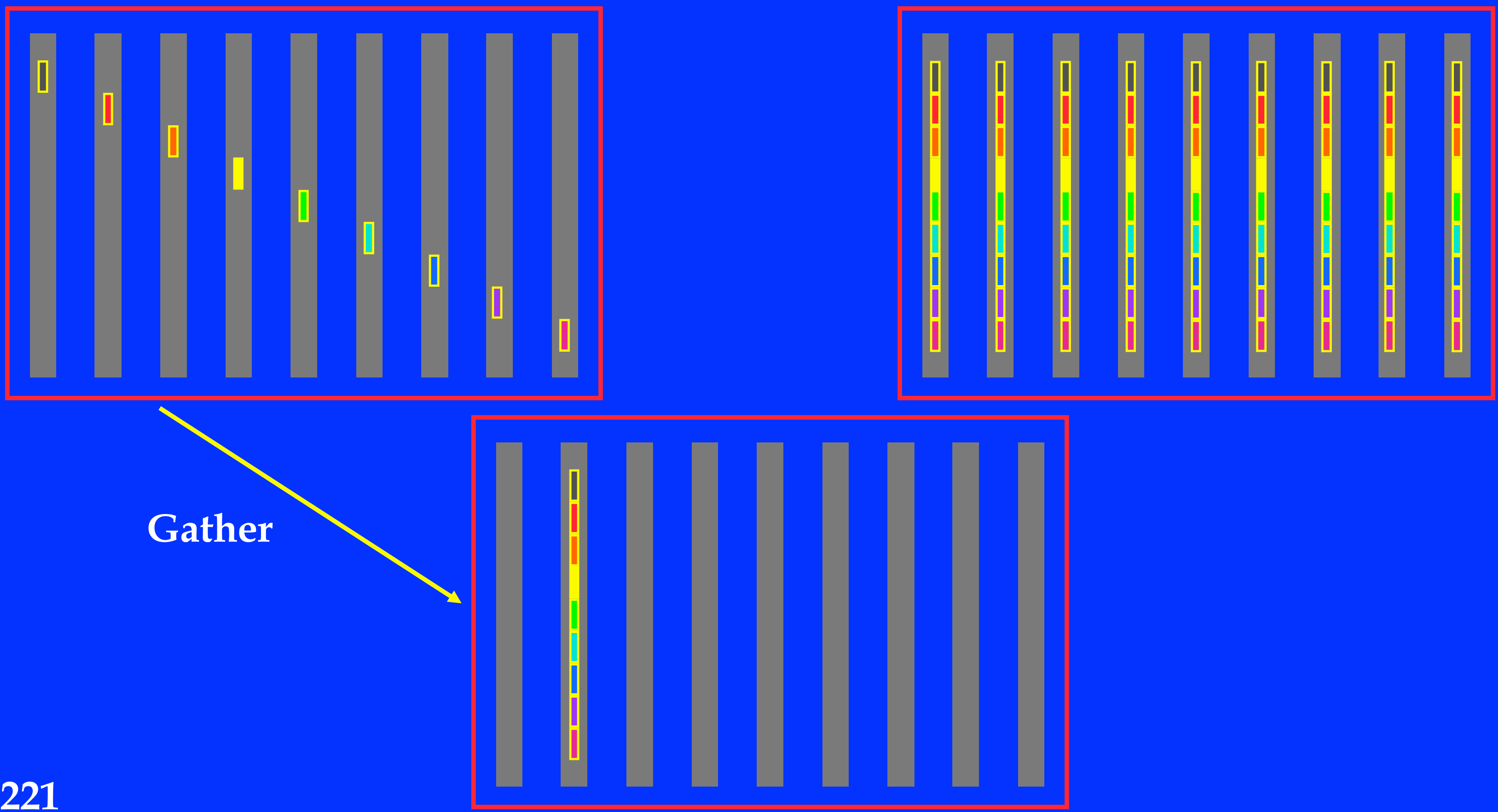
Notice: attains lower bound for latency and bandwidth components

Using the building blocks

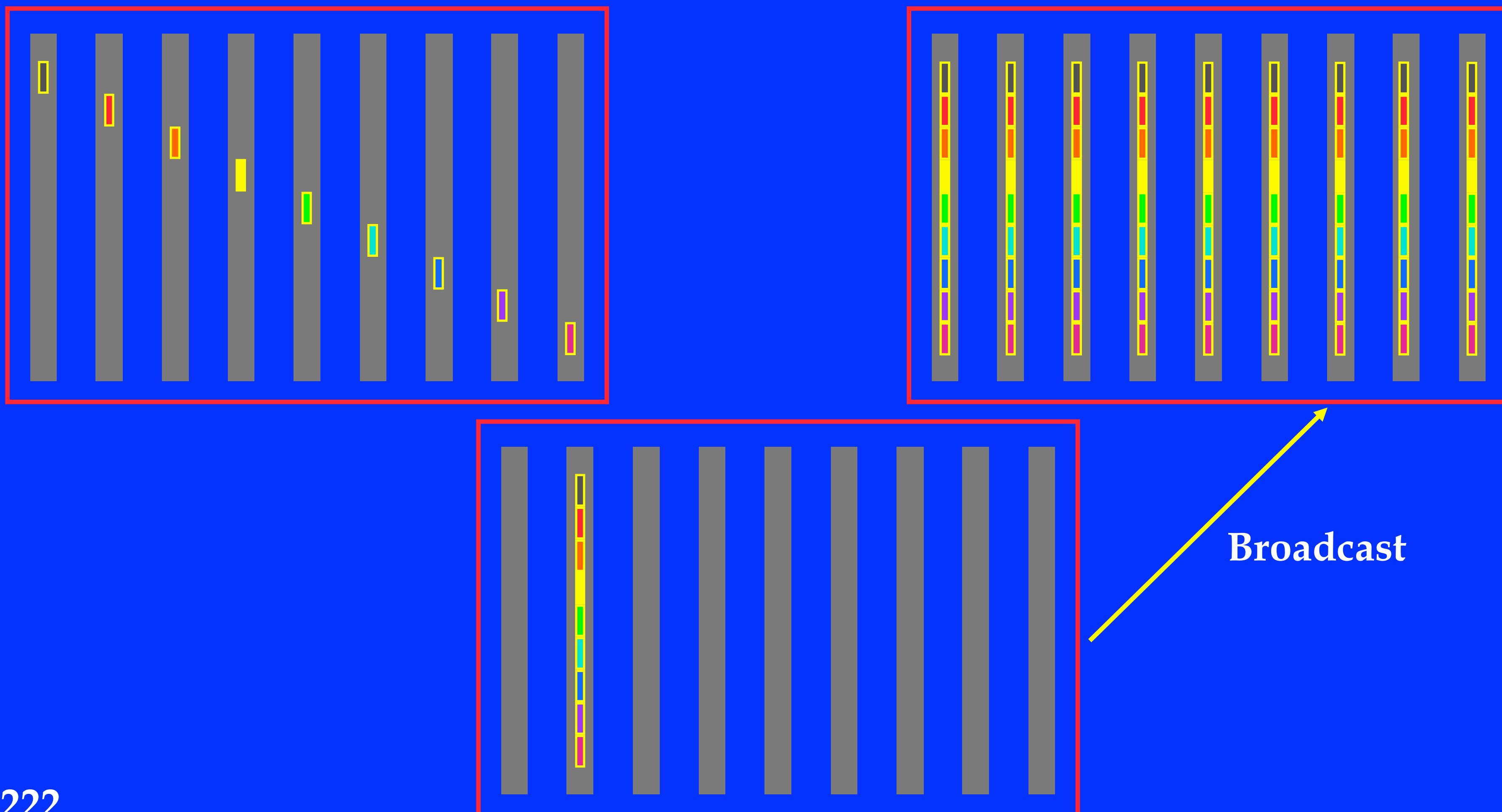
Allgather (short vector)



Allgather (short vector)



Allgather (short vector)



Cost of gather/broadcast allgather

- **Assumption: power of two number of nodes**

gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

broadcast

$$\log(p)(\alpha + n\beta)$$

$$2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p)\right)n\beta$$

Cost of gather/broadcast allgather

- **Assumption: power of two number of nodes**

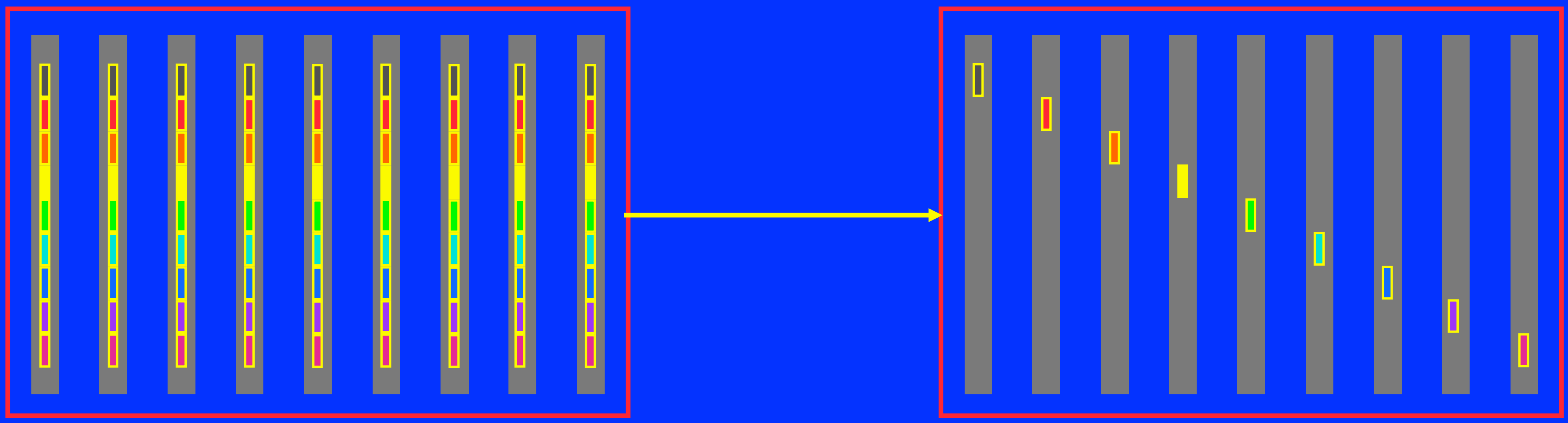
gather
$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

broadcast
$$\log(p)(\alpha + n\beta)$$

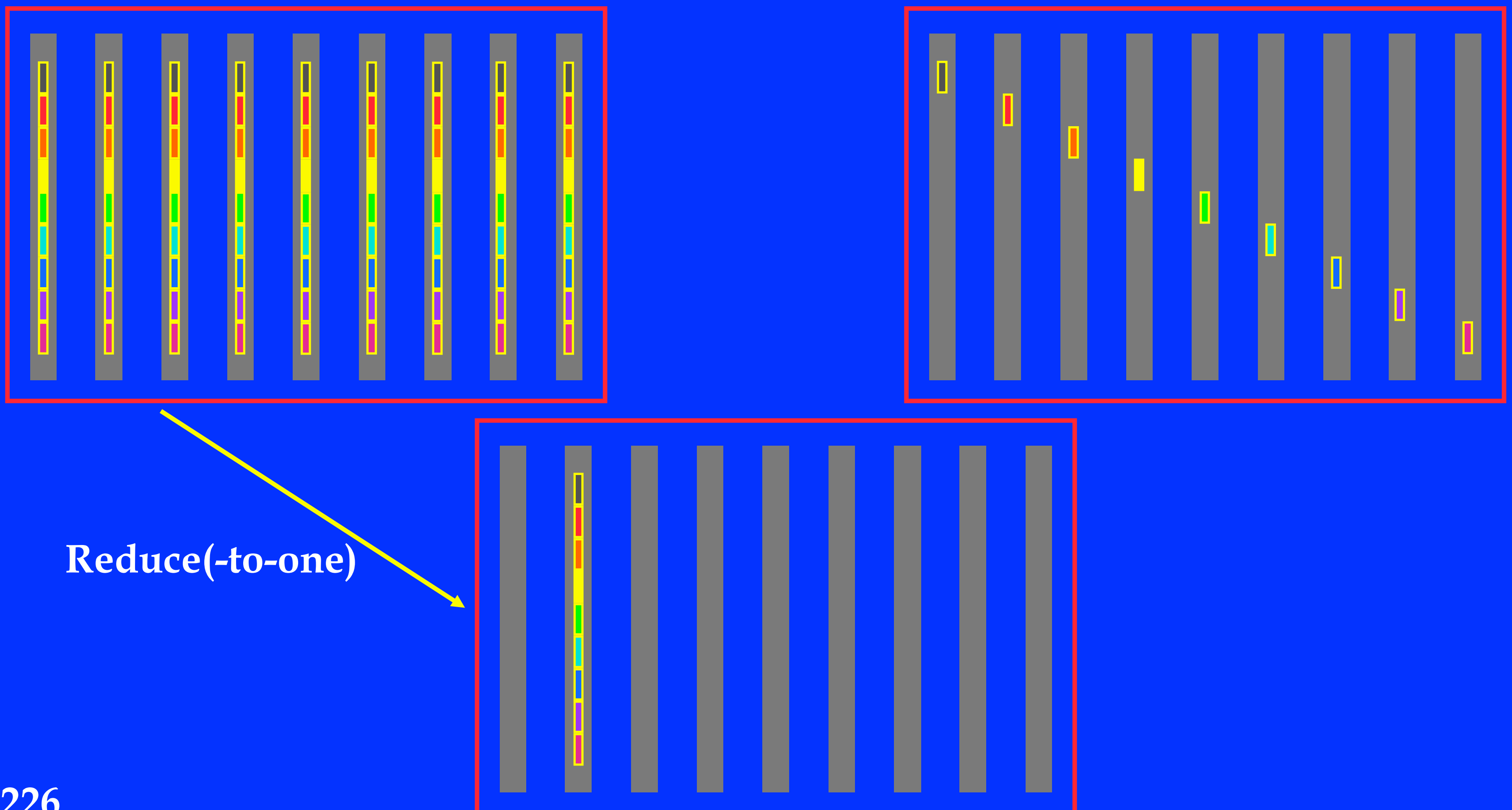
$$2\log(p)\alpha + \left(\frac{p-1}{p} + \log(p)\right) n\beta$$

Notice: does not attain lower bound for latency or bandwidth components

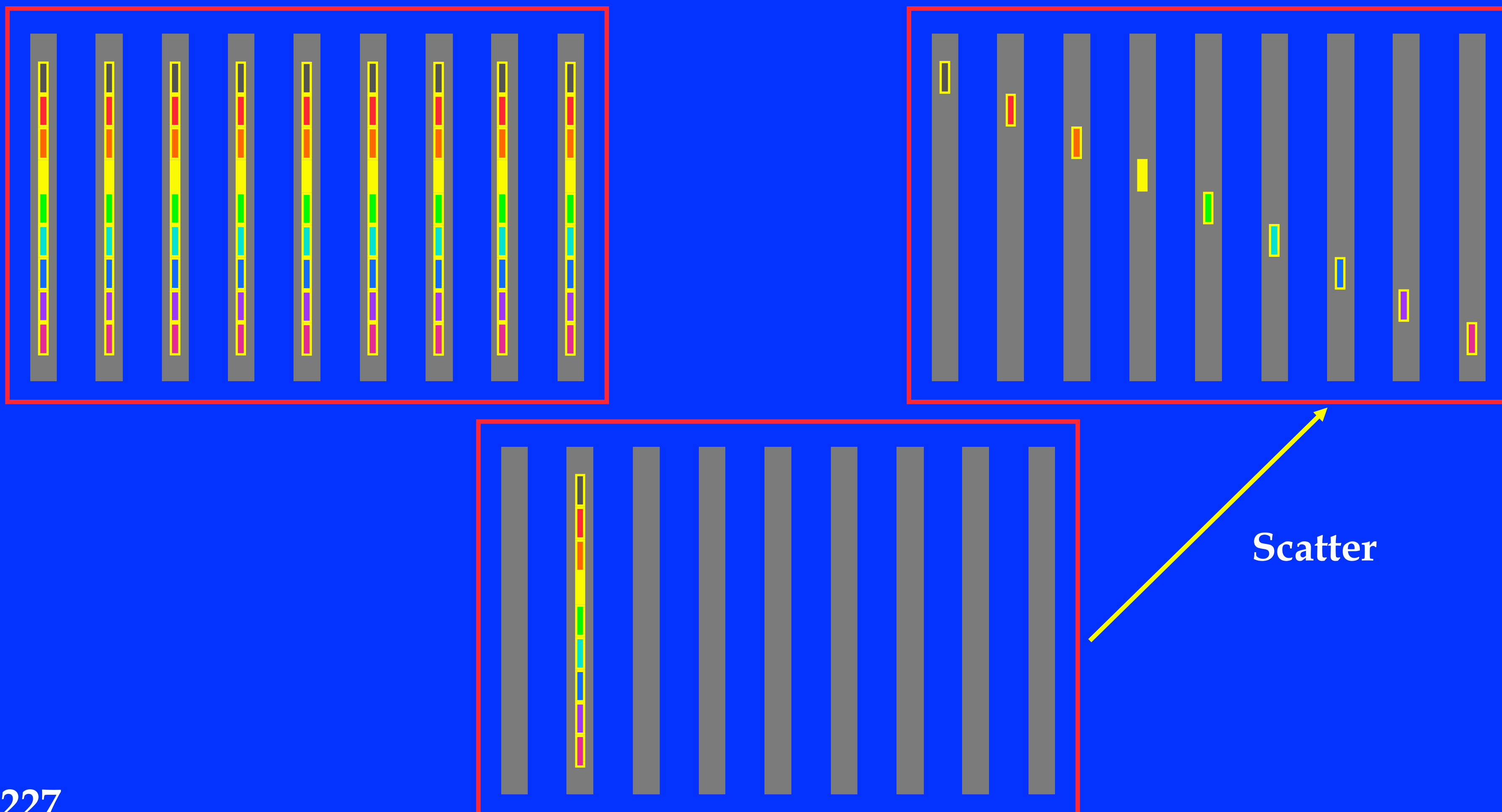
Reduce-scatter (short vector)



Reduce-scatter
(short vector)



Reduce-scatter (short vector)



Cost of Reduce(-to-one)/scatter Reduce-scatter

- **Assumption: power of two number of nodes**

Reduce(-to-one) $log(p)(\alpha + n\beta + n\gamma)$

scatter $log(p)\alpha + \frac{p-1}{p} n\beta$

$$2log(p)\alpha + \left(\frac{p-1}{p} + log(p)\right) n\beta + log(p)n\gamma$$

Cost of Reduce(-to-one)/scatter reduce-scatter

- **Assumption: power of two number of nodes**

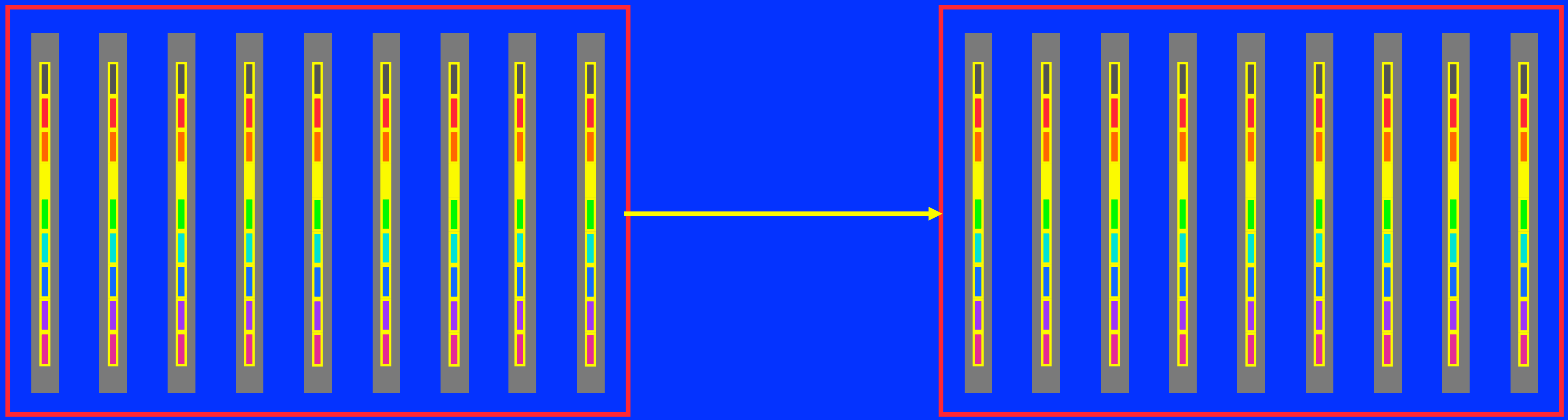
Reduce(-to-one) $log(p)(\alpha + n\beta + n\gamma)$

scatter $log(p)\alpha + \frac{p-1}{p} n\beta$

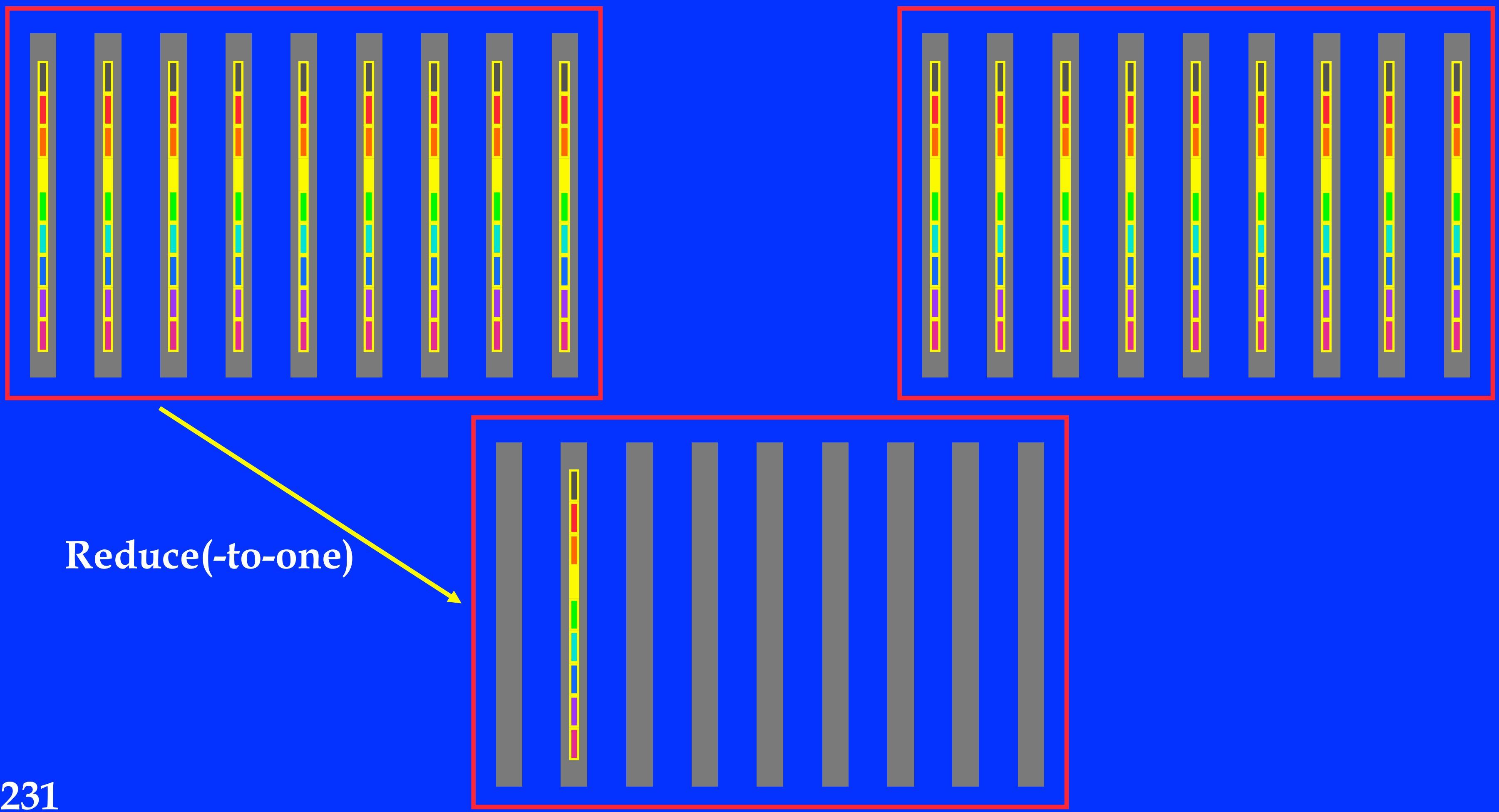
$$2log(p)\alpha + \left(\frac{p-1}{p} + log(p) \right) n\beta + log(p) n\gamma$$

Notice: does not attain lower bound for latency or bandwidth components

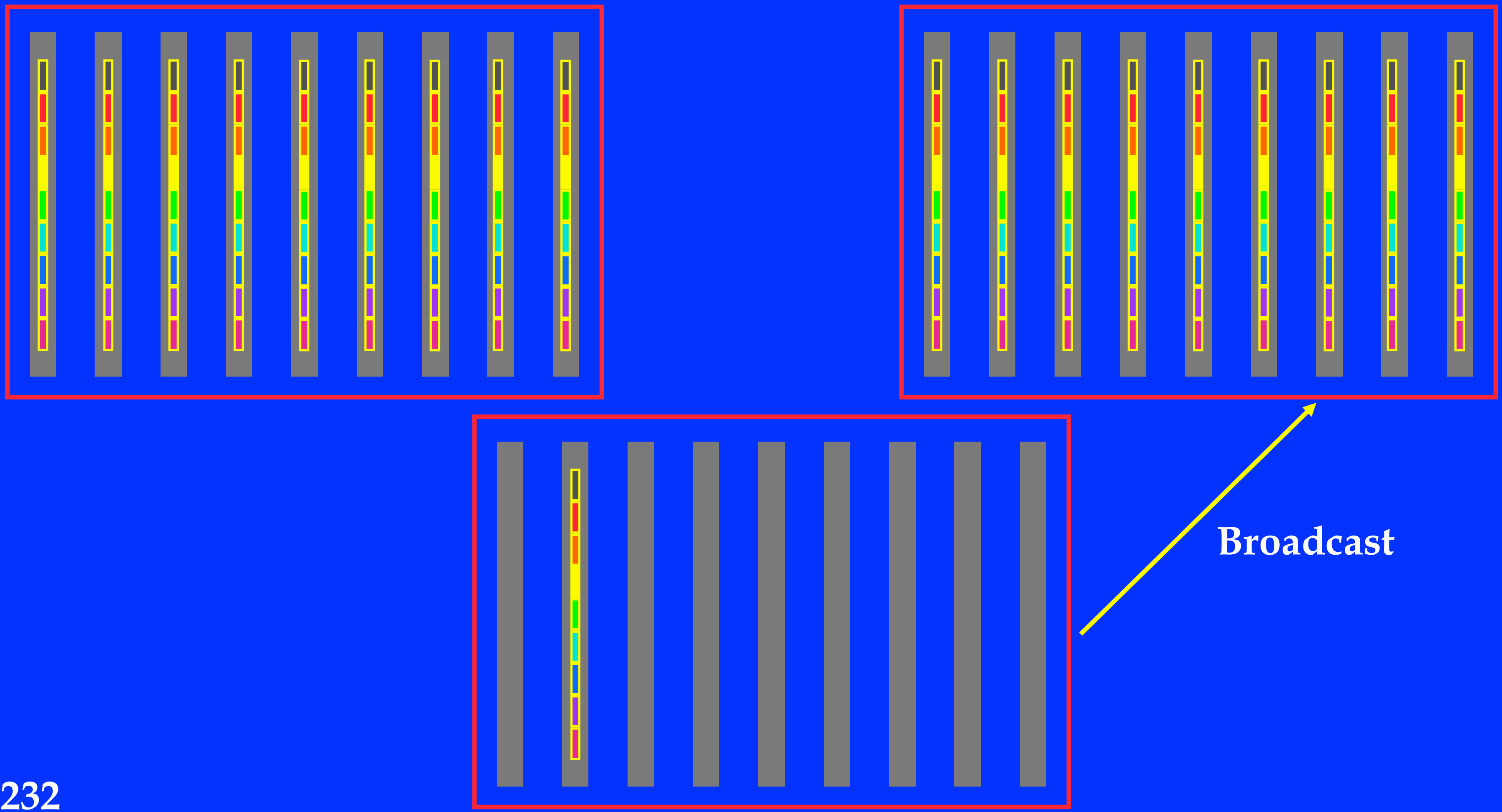
Allreduce (short vector)



Allreduce
(short vector)



Allreduce
(short vector)



Cost of reduce(-to-one)/broadcast Allreduce

- Assumption: power of two number of nodes

Reduce(-to-one) $log(p)(\alpha + n\beta + n\gamma)$

broadcast $log(p)(\alpha + n\beta)$

$$2log(p)\alpha + 2log(p)n\beta + log(p)n\gamma$$

Cost of reduce(-to-one)/broadcast Allreduce

- Assumption: power of two number of nodes

Reduce(-to-one) $log(p)(\alpha + n\beta + n\gamma)$

broadcast $log(p)(\alpha + n\beta)$

$$2log(p)\alpha + 2log(p)n\beta + log(p)n\gamma$$

Notice: does not attain lower bound for latency or bandwidth components

Recap

Reduce(-to-one)

$log(p)(\alpha + n\beta + n\gamma)$

Scatter

$log(p)\alpha + \frac{p-1}{p} n\beta$

Gather

$log(p)\alpha + \frac{p-1}{p} n\beta$

Broadcast

$log(p)(\alpha + n\beta)$

Reduce-scatter

Allreduce

Allgather

Recap

Reduce(-to-one)

$log(p)(\alpha + n\beta + n\gamma)$

Scatter

$log(p)\alpha + \frac{p-1}{p} n\beta$

Gather

$log(p)\alpha + \frac{p-1}{p} n\beta$

Broadcast

$log(p)(\alpha + n\beta)$

Reduce-scatter

$2log(p)\alpha + log(p)n(\beta + \gamma) + \frac{p-1}{p} n\beta$

Allreduce

Allgather

Recap

Reduce(-to-one)

$log(p)(\alpha + n\beta + n\gamma)$

Scatter

$log(p)\alpha + \frac{p-1}{p} n\beta$

Gather

$log(p)\alpha + \frac{p-1}{p} n\beta$

Broadcast

$log(p)(\alpha + n\beta)$

Reduce-scatter

$2log(p)\alpha + log(p)n(\beta + \gamma) + \frac{p-1}{p} n\beta$

Allreduce

$2log(p)\alpha + log(p)n(2\beta + \gamma)$

Allgather

$2log(p)\alpha + log(p)n\beta + \frac{p-1}{p} n\beta$

Recap

Reduce(-to-one)

$\log(p)(\alpha + n\beta + n\gamma)$

Scatter

$\log(p)\alpha + \frac{p-1}{p} n\beta$

Gather

$\log(p)\alpha + \frac{p-1}{p} n\beta$

Broadcast

$\log(p)(\alpha + n\beta)$

Reduce-scatter

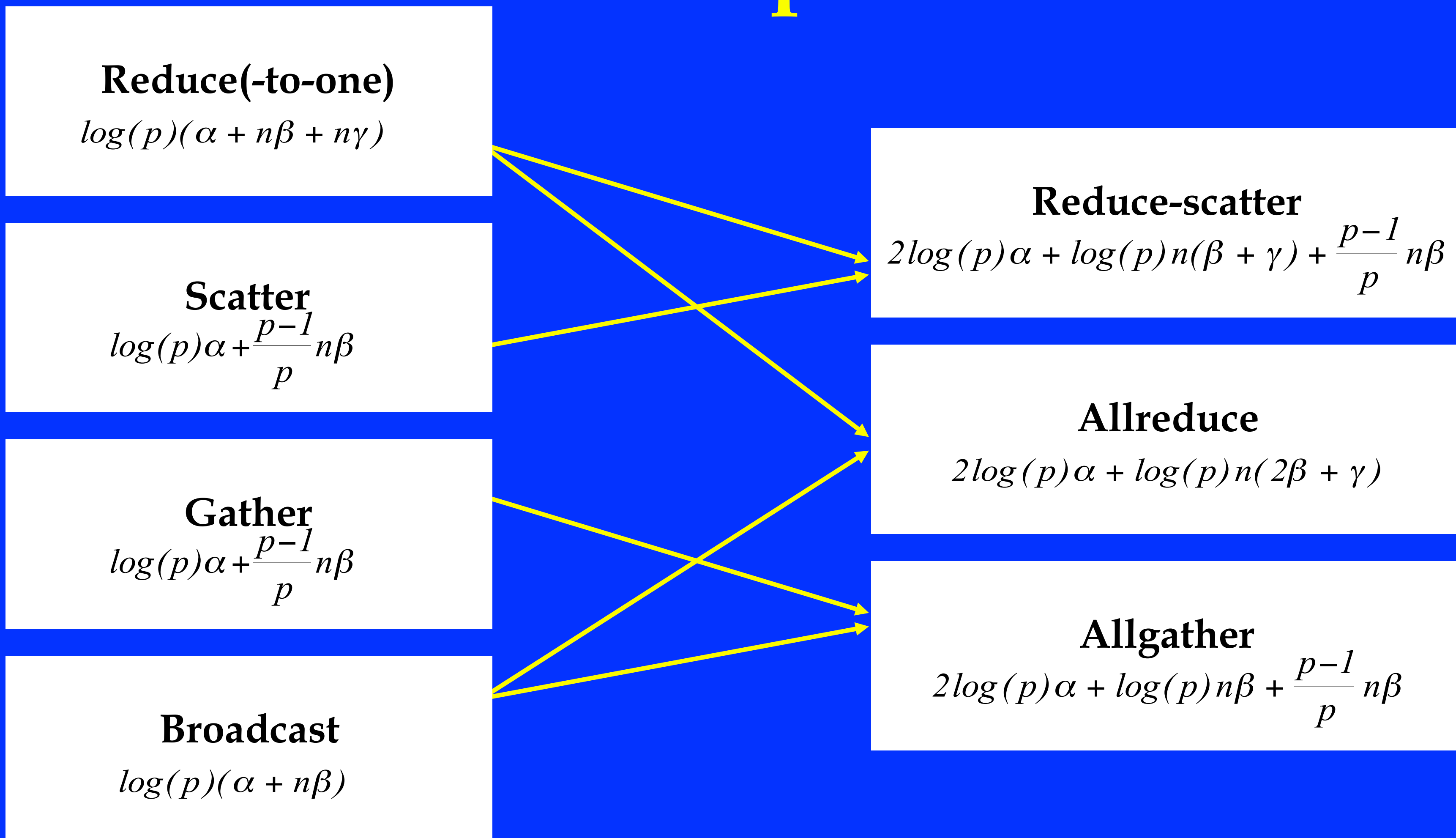
$2\log(p)\alpha + \log(p)n(\beta + \gamma) + \frac{p-1}{p} n\beta$

Allreduce

$2\log(p)\alpha + \log(p)n(2\beta + \gamma)$

Allgather

Recap


Reduce(-to-one)
$log(p)(\alpha + n\beta + n\gamma)$
Scatter
$log(p)\alpha + \frac{p-1}{p} n\beta$
Gather
$log(p)\alpha + \frac{p-1}{p} n\beta$
Broadcast
$log(p)(\alpha + n\beta)$
Reduce-scatter
$2log(p)\alpha + log(p)n(\beta + \gamma) + \frac{p-1}{p} n\beta$
Allreduce
$2log(p)\alpha + log(p)n(2\beta + \gamma)$
Allgather
$2log(p)\alpha + log(p)n\beta + \frac{p-1}{p} n\beta$

A building block approach to library implementation

- Short vector case
- **Long vector case**
- Hybrid algorithms

Long vector case

- Primary concern:
  - algorithms must have low cost due to vector length
  - algorithms must avoid network conflicts

- Secondary concerns:
  - algorithms must work for arbitrary number of nodes
    - in particular, not just for power-of-two numbers of nodes

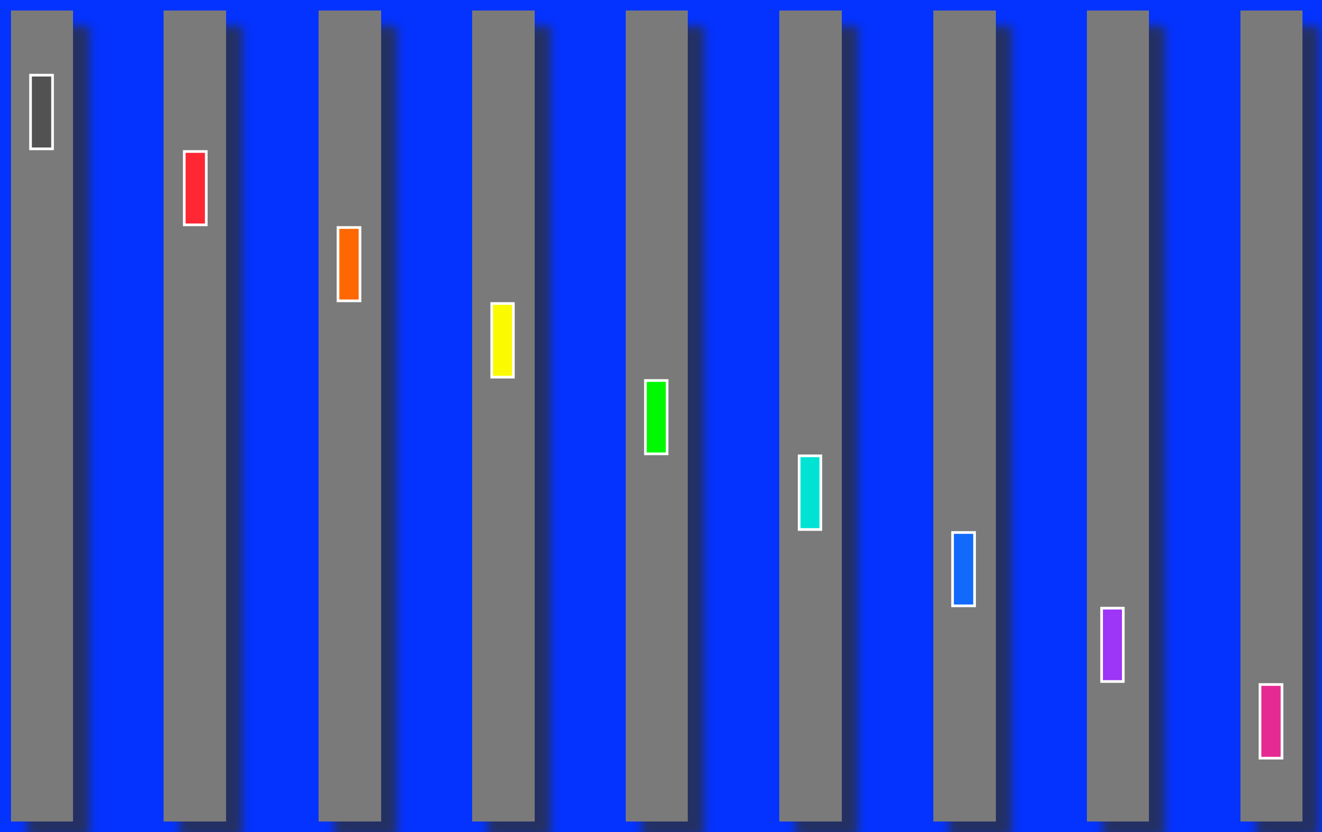
Long vector building blocks

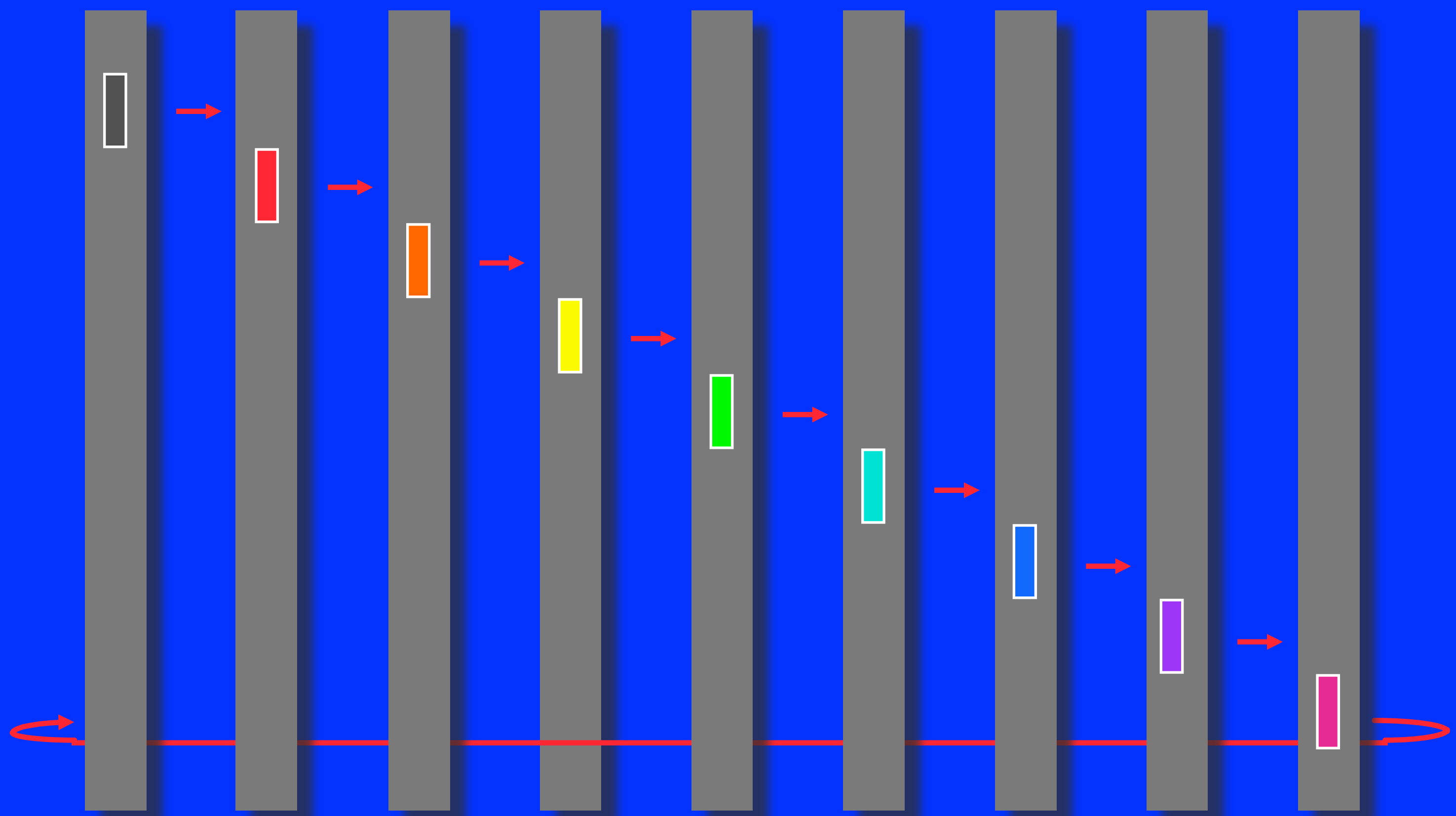
- We will show how the following building blocks:
  - collect/distributed combine
  - scatter/gather

  can be implemented using “bucket” algorithms while attaining
  - minimal cost due to length of vectors
  - implementation for arbitrary numbers of nodes
  - no network conflicts
- NOTICE: scatter and gather already satisfy these conditions

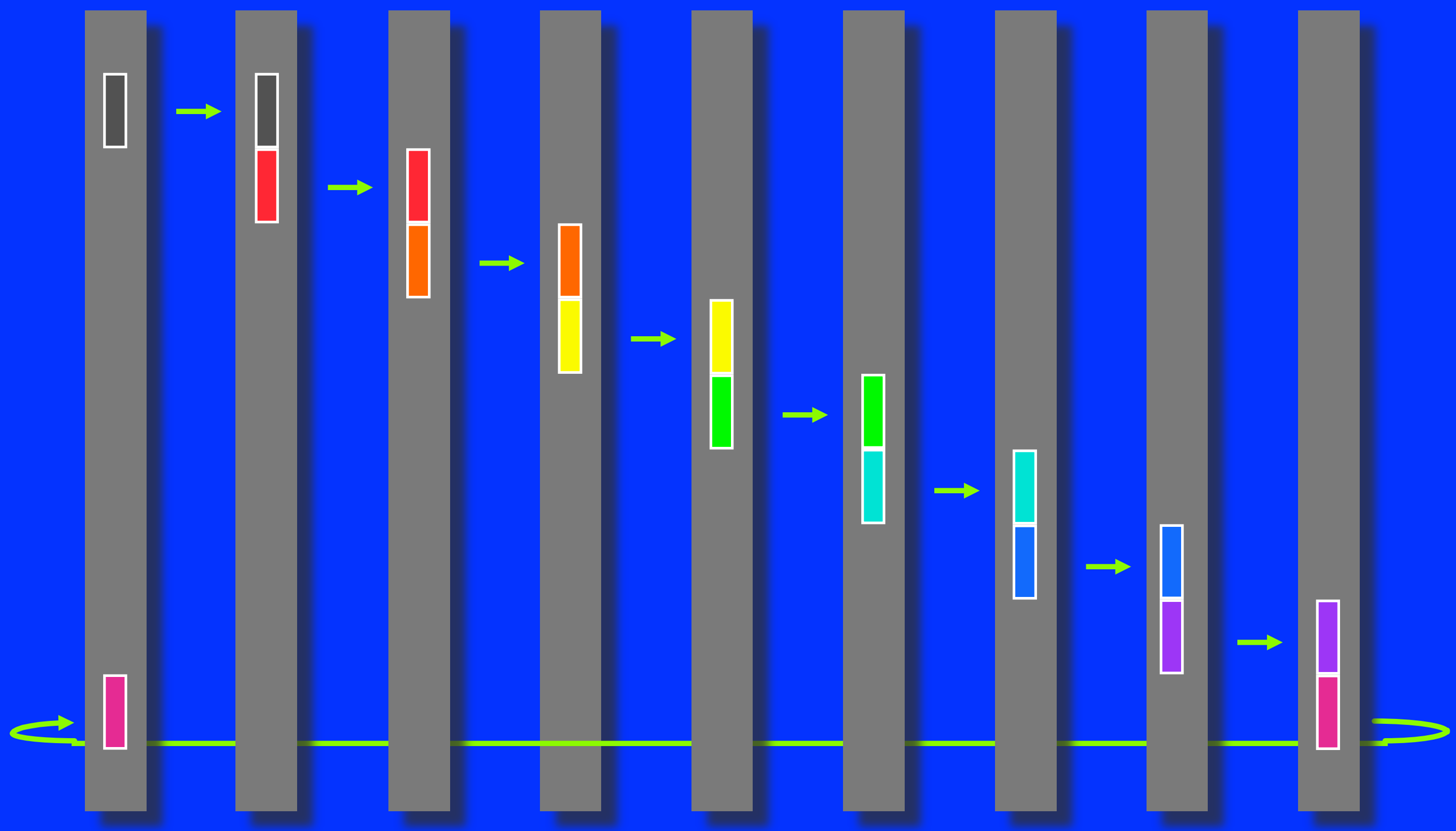
General principles

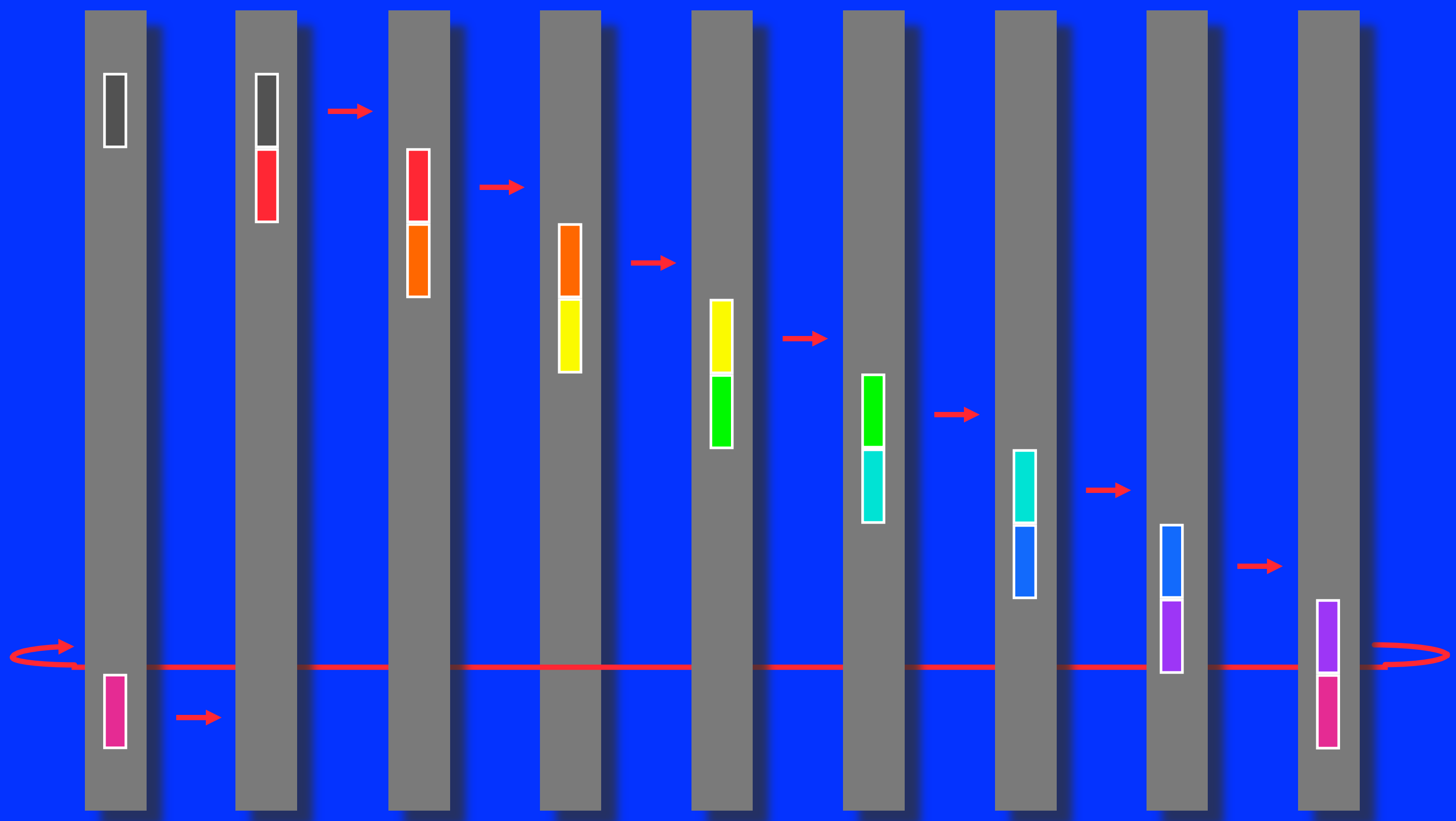
- A logical ring can be embedded in a physical linear array with worm-hole routing, since the "wrap-around" message doesn't conflict
  - This is used to "drop off" messages or to "pick up" contributions

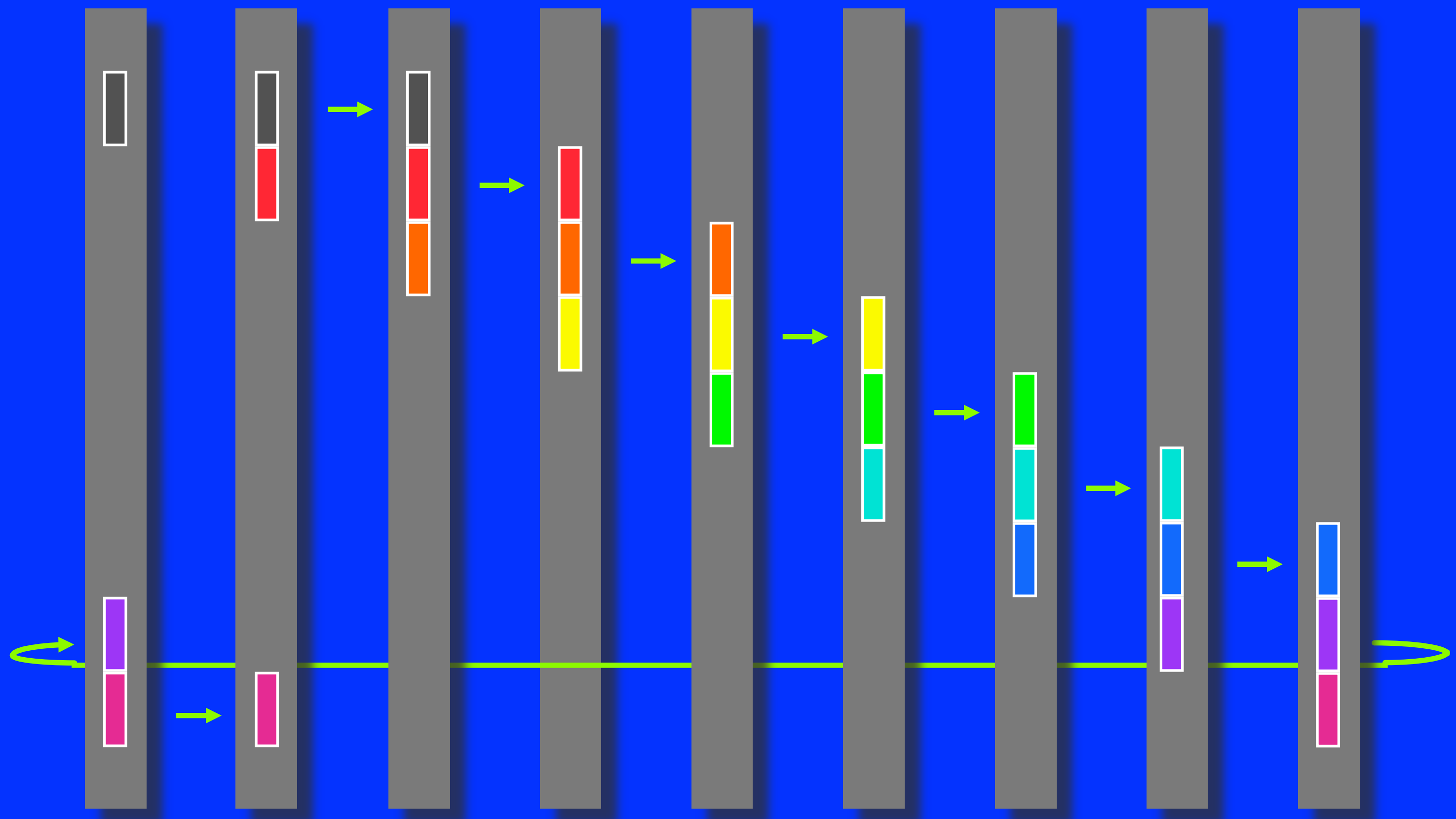




- A logical ring can be embedded in a physical linear array with worm-hole routing, since the “wrap-around” message doesn’t conflict





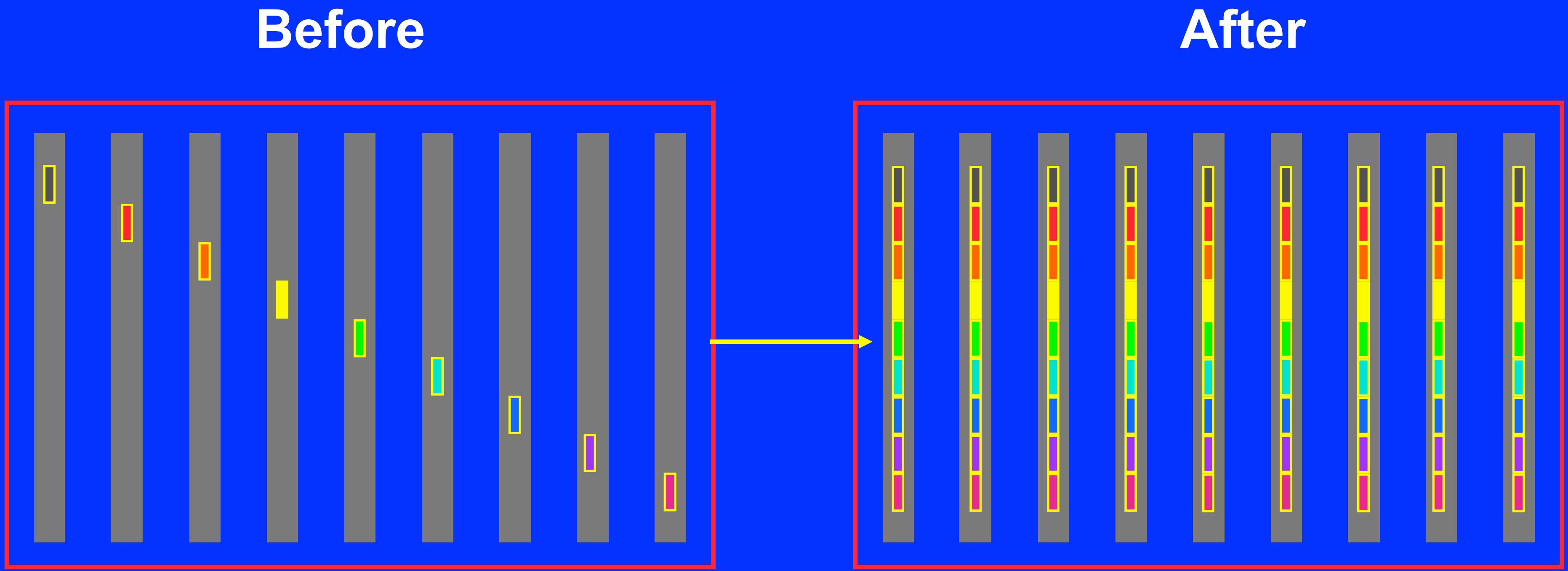


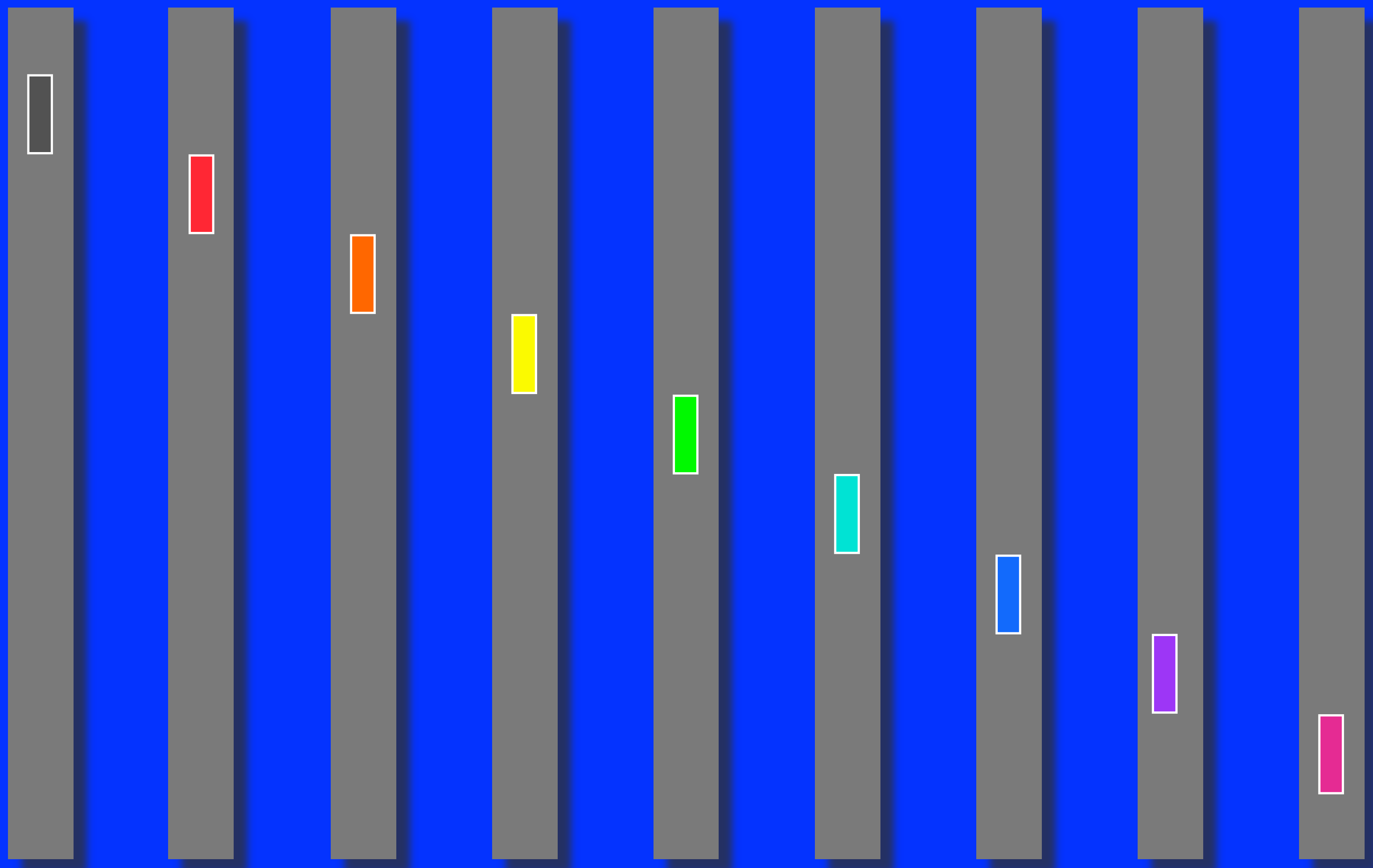
General principles

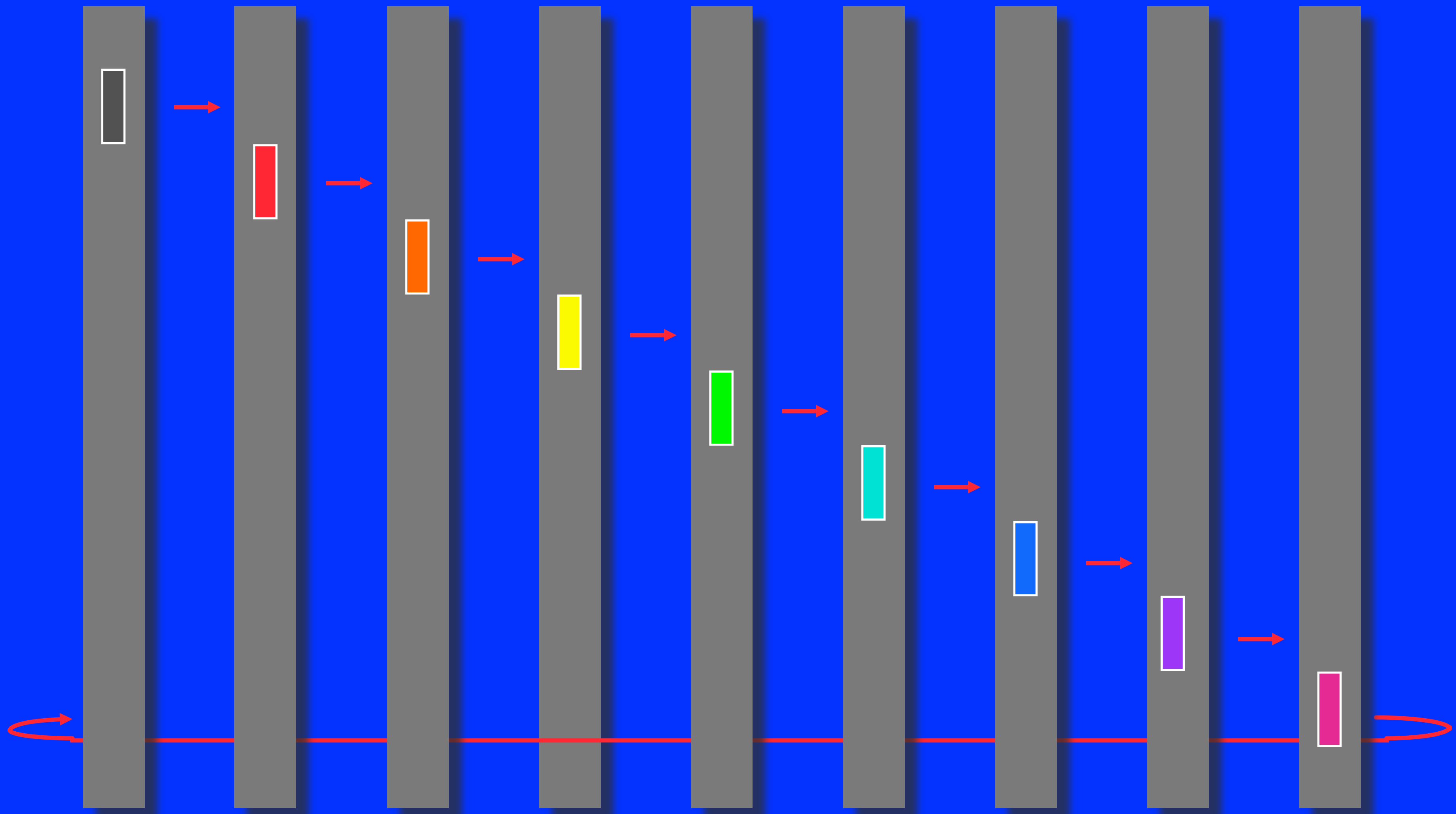
- Can be used to implement the following building blocks:
  - collect
  - distributed combine

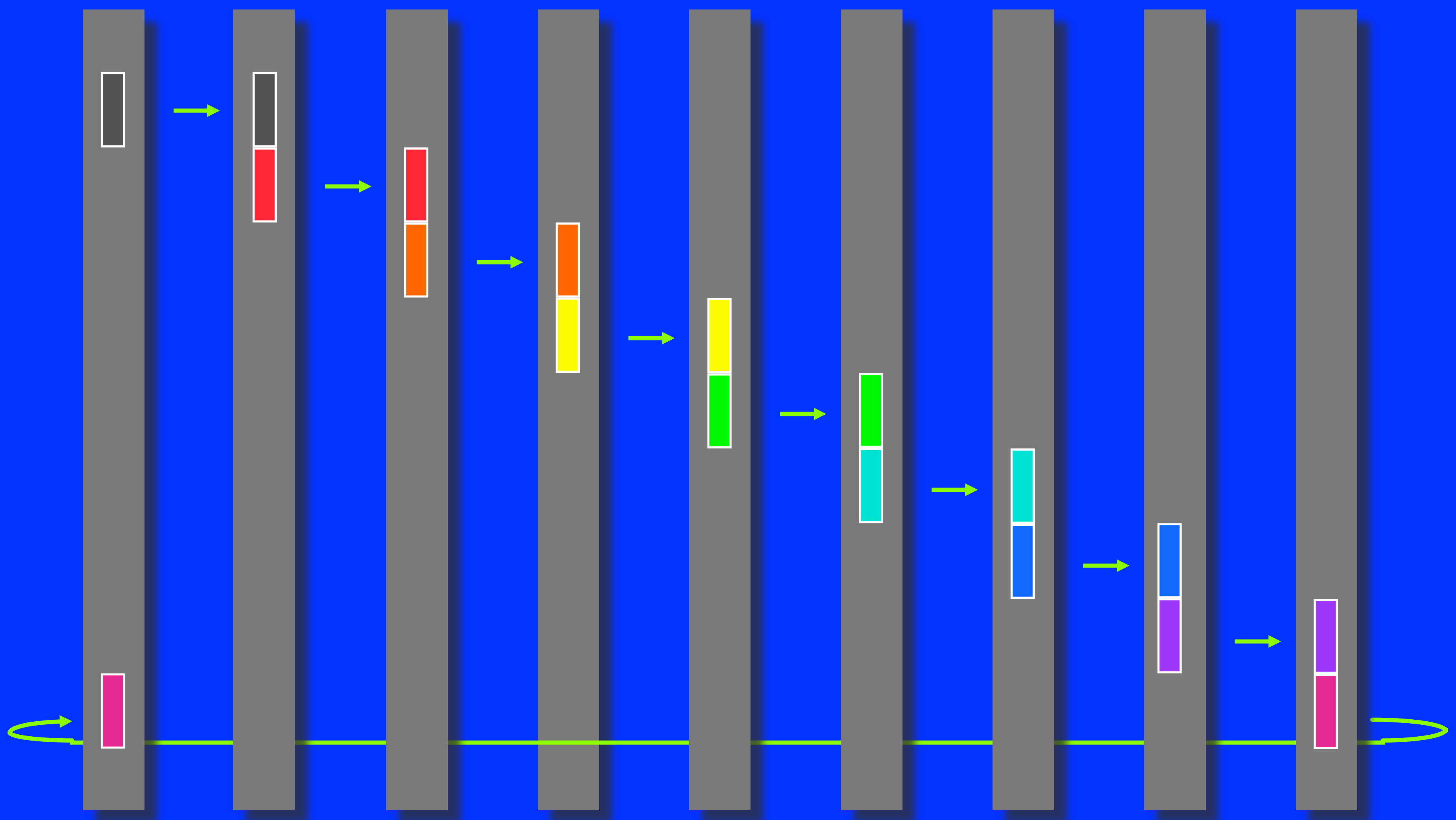
  using a bucket algorithm embedded in the physical linear array while attaining
  - minimal cost due to vector length
  - implementation for arbitrary numbers of nodes
  - no network conflicts

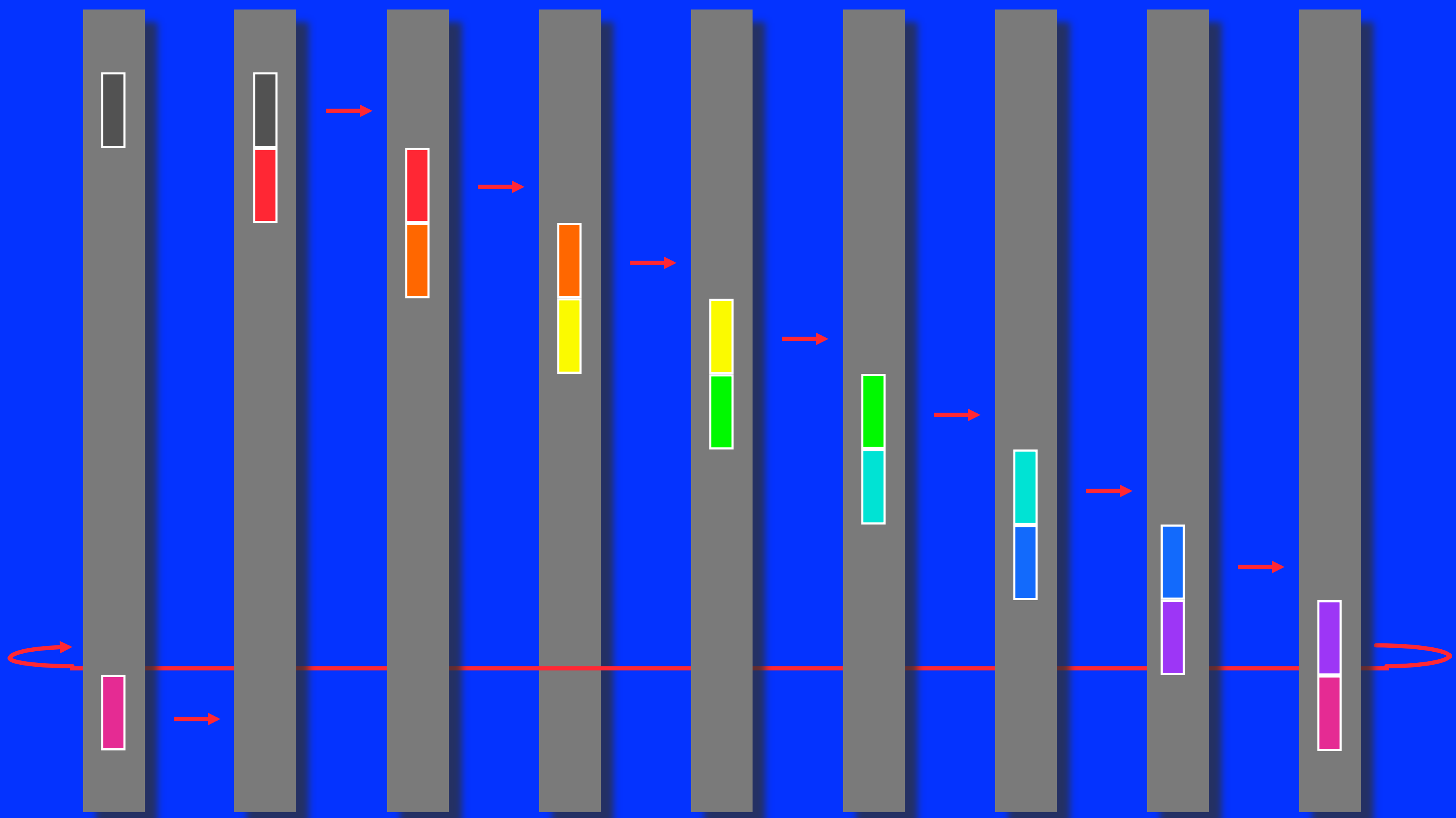
Allgather



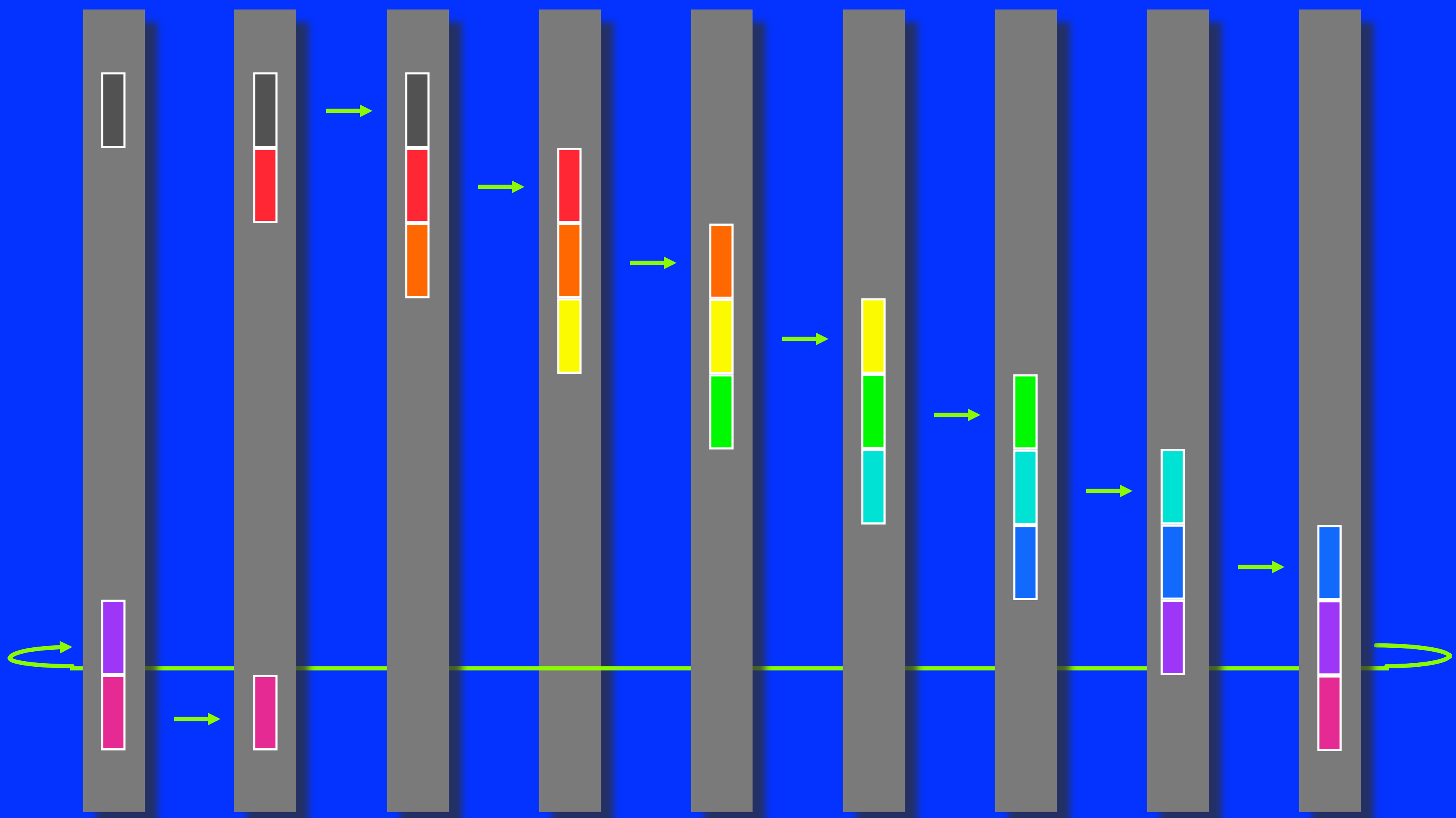


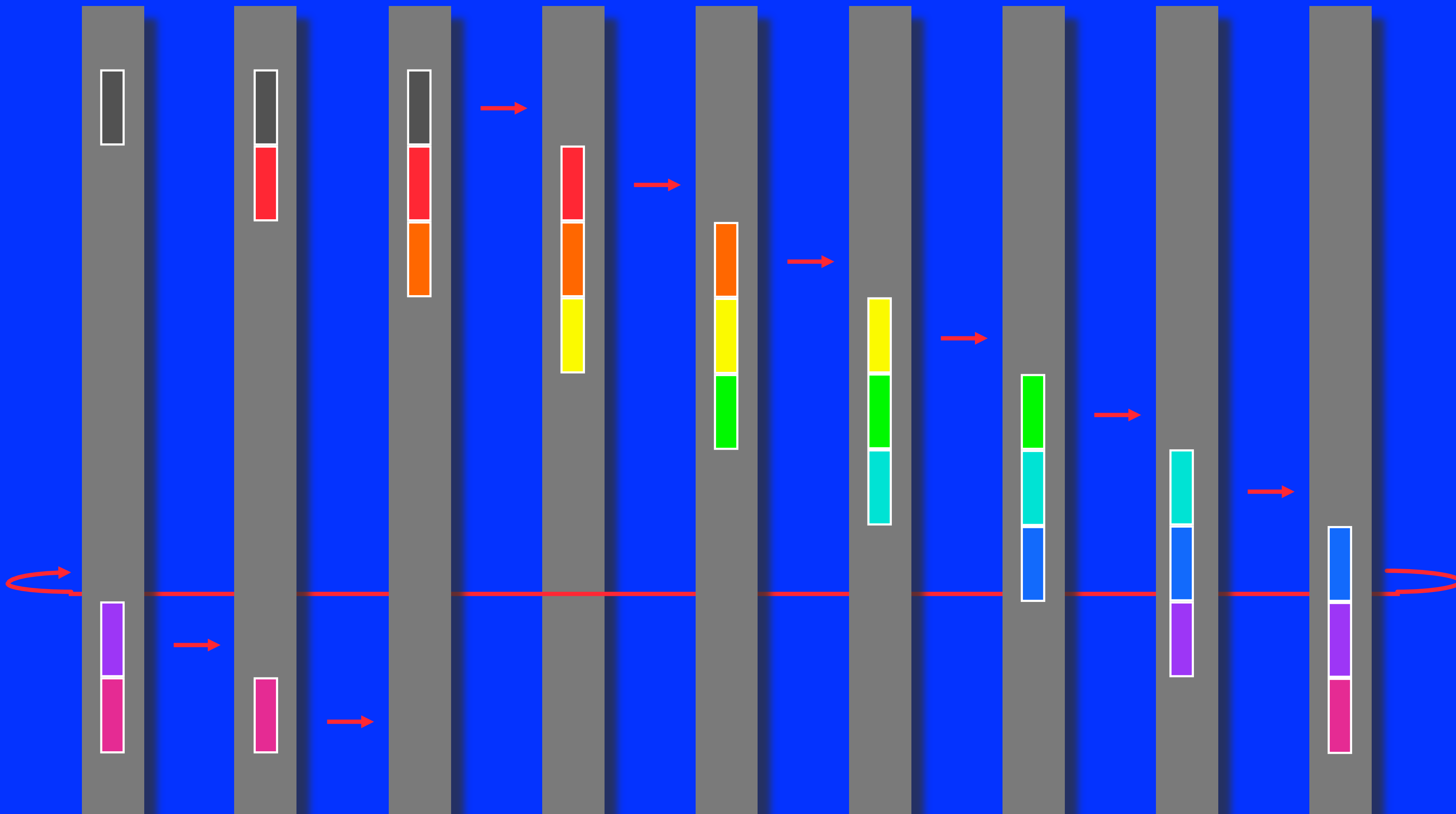


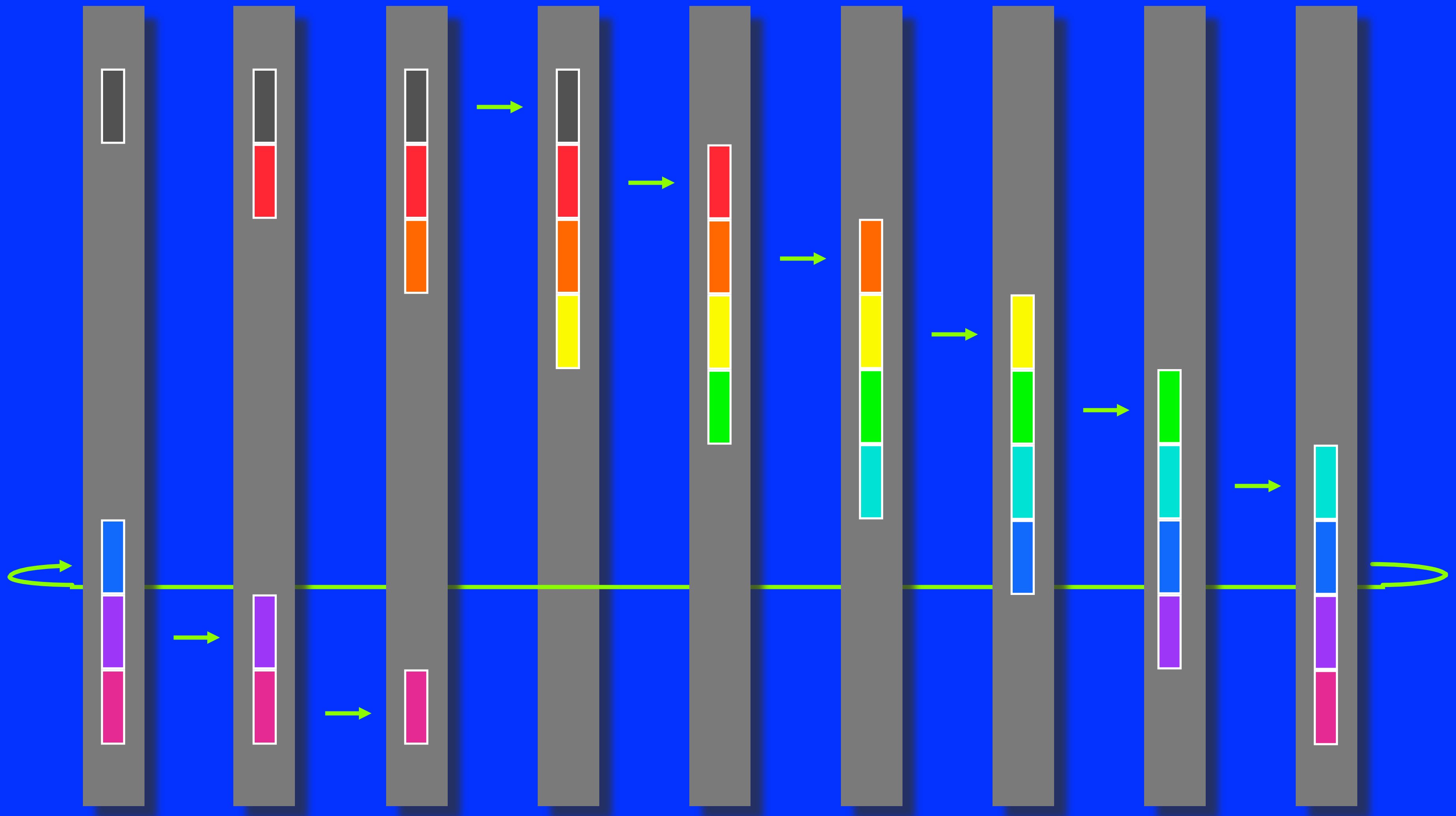


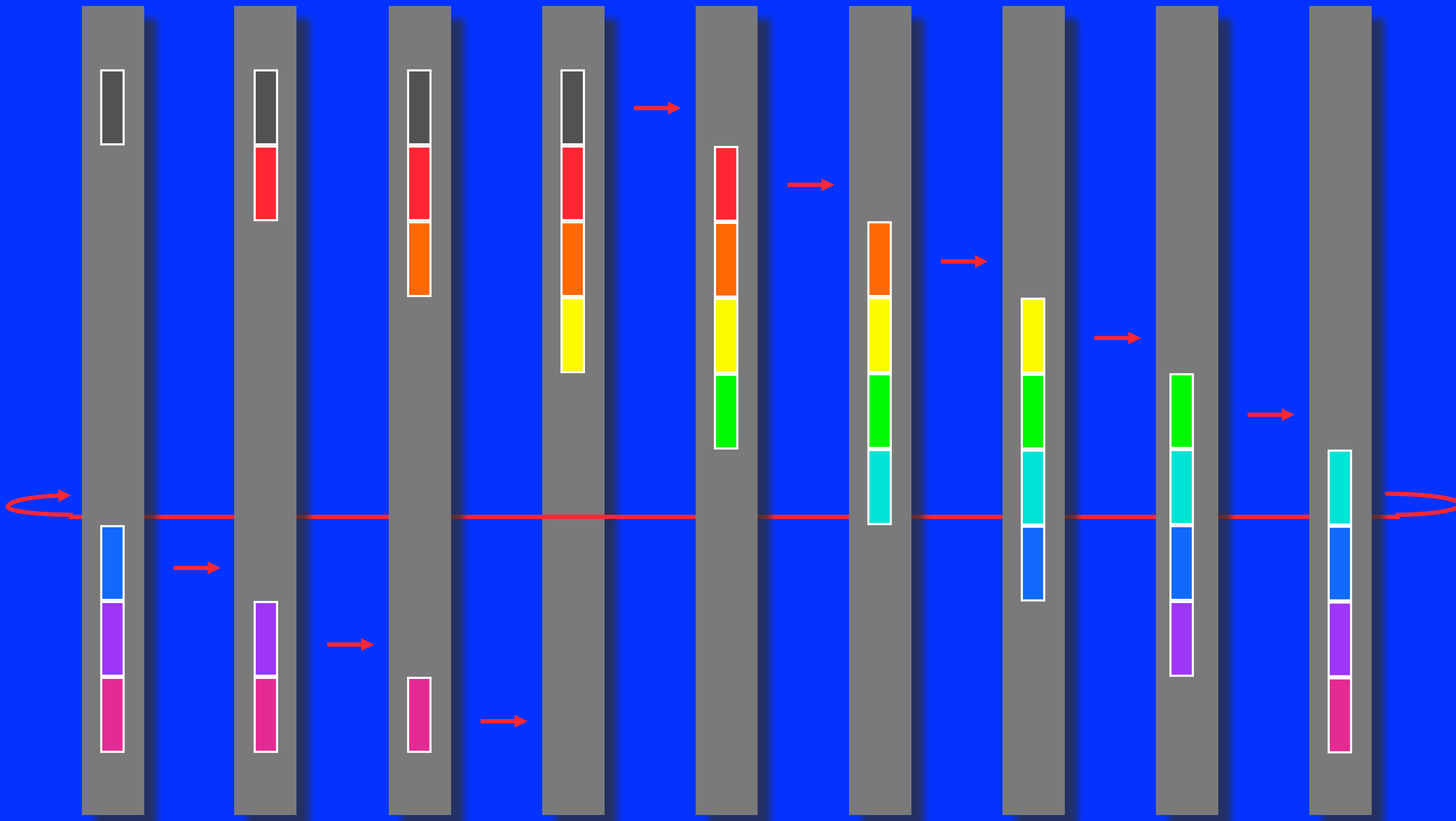


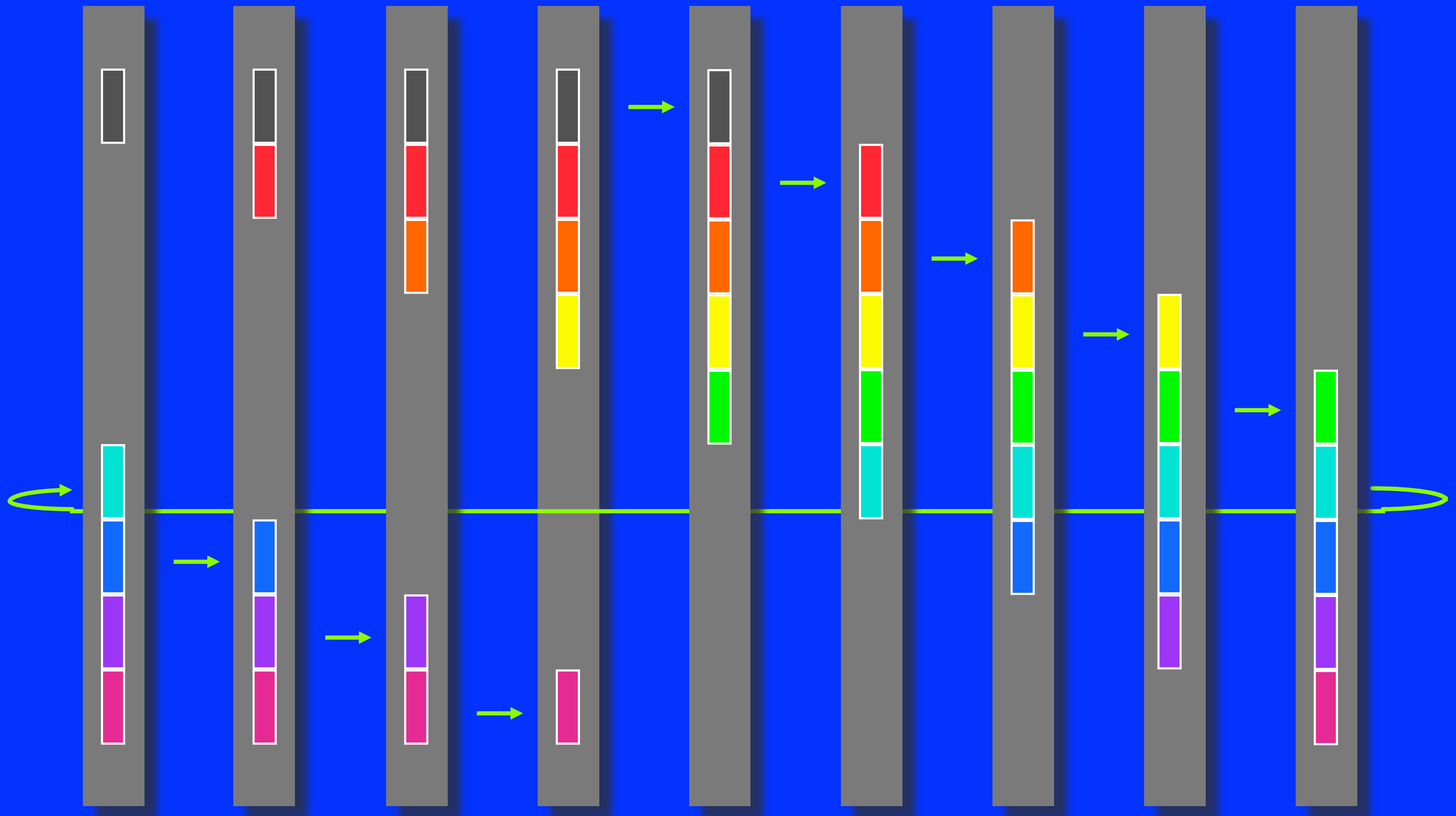
254

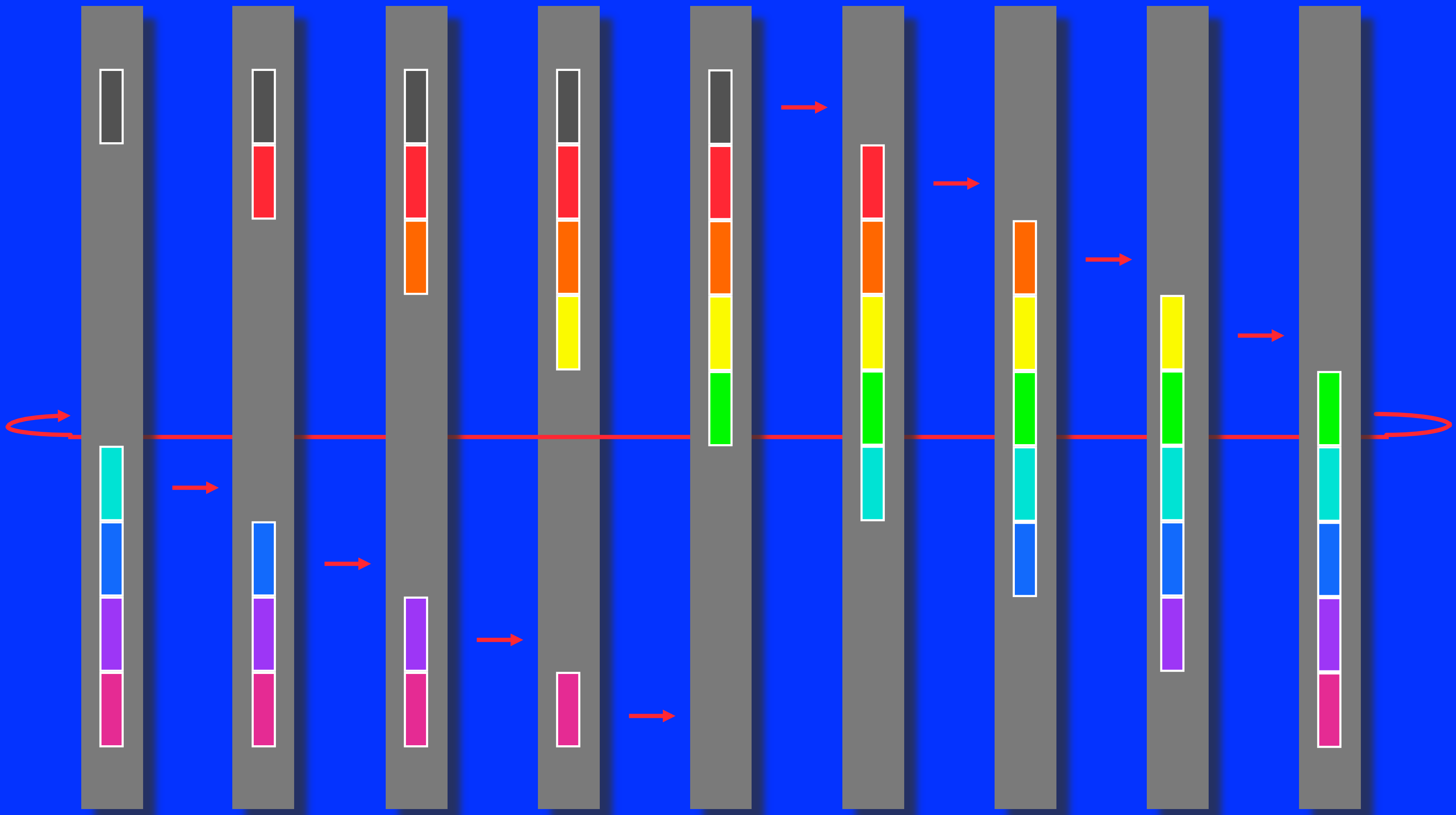


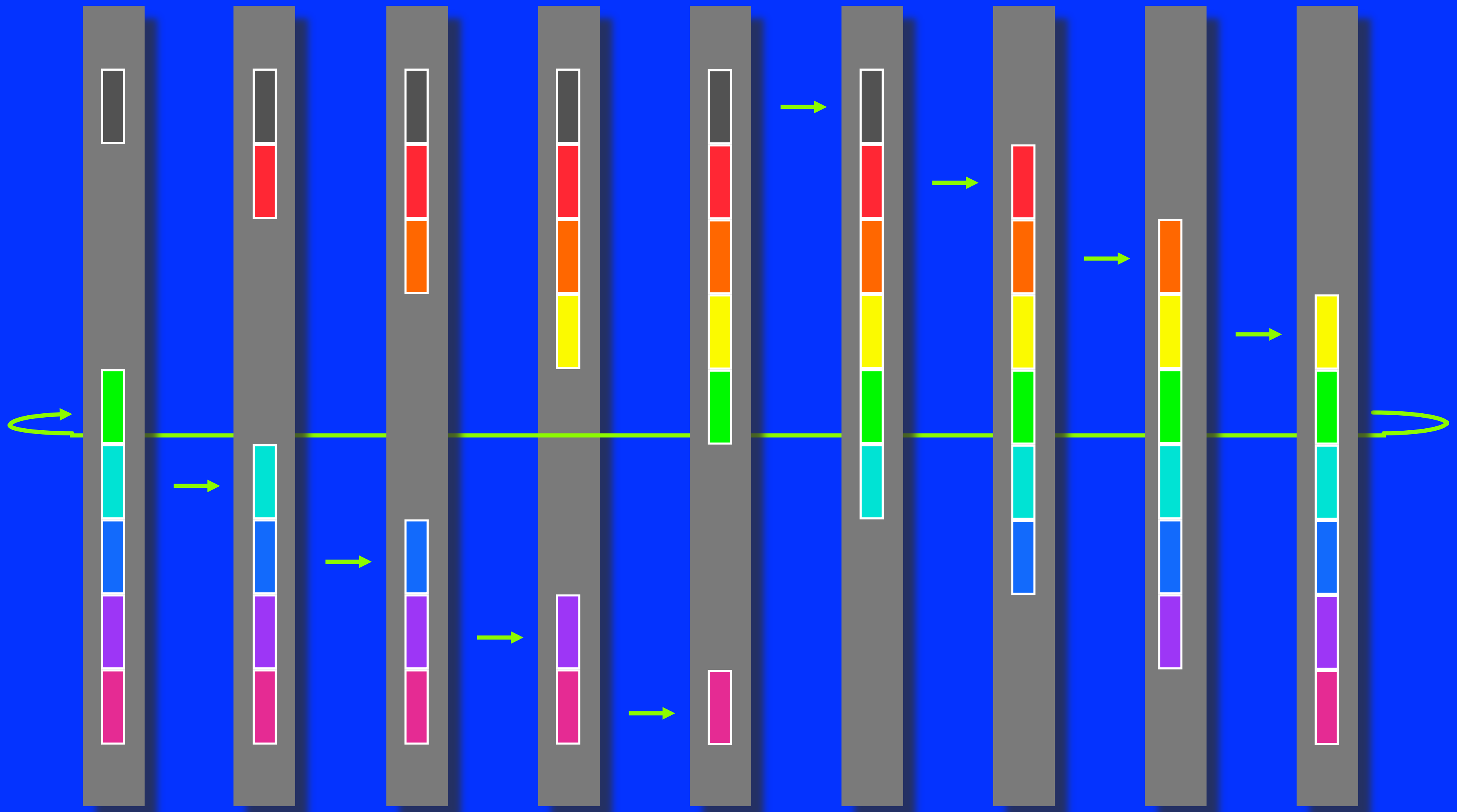


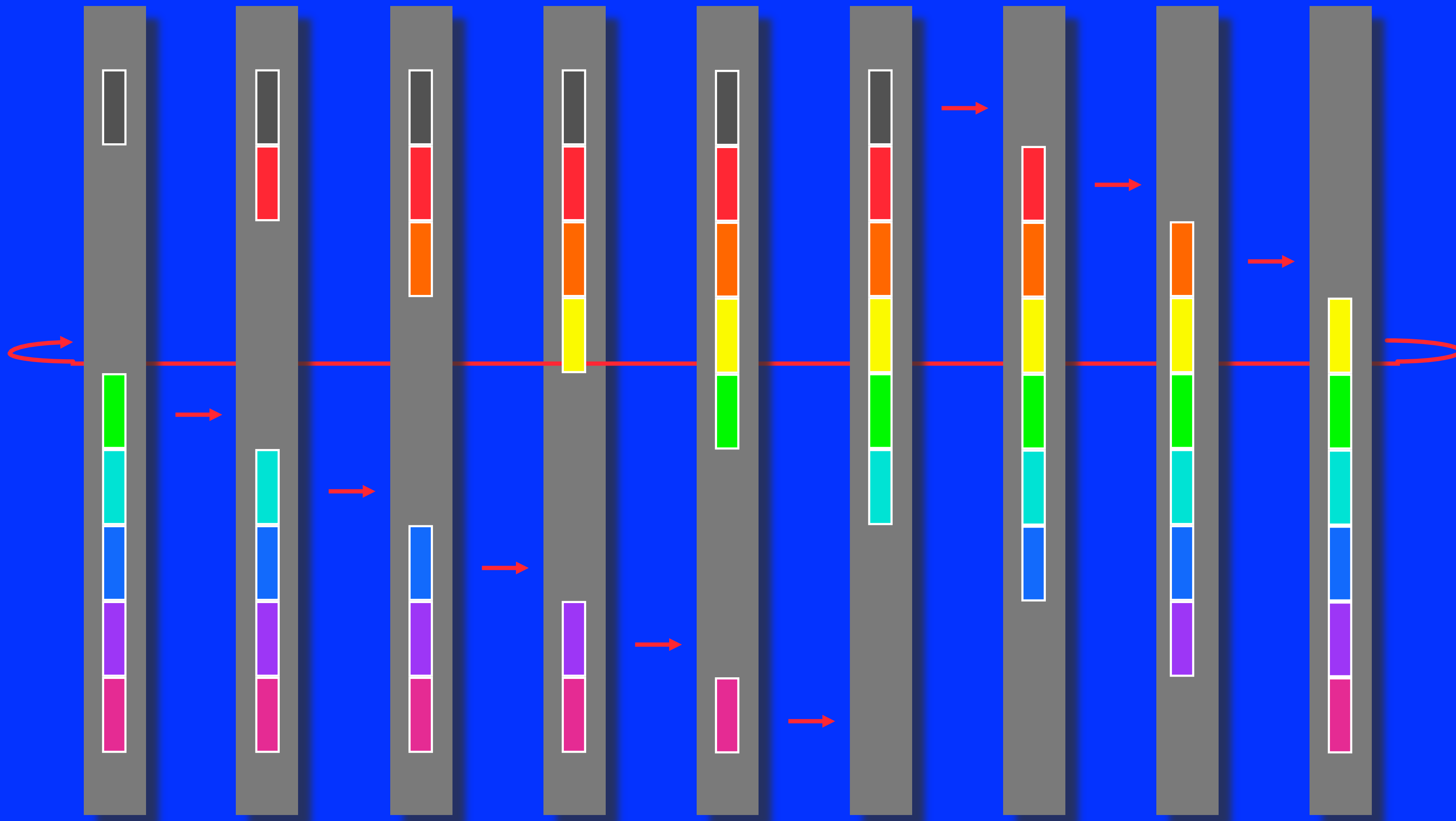


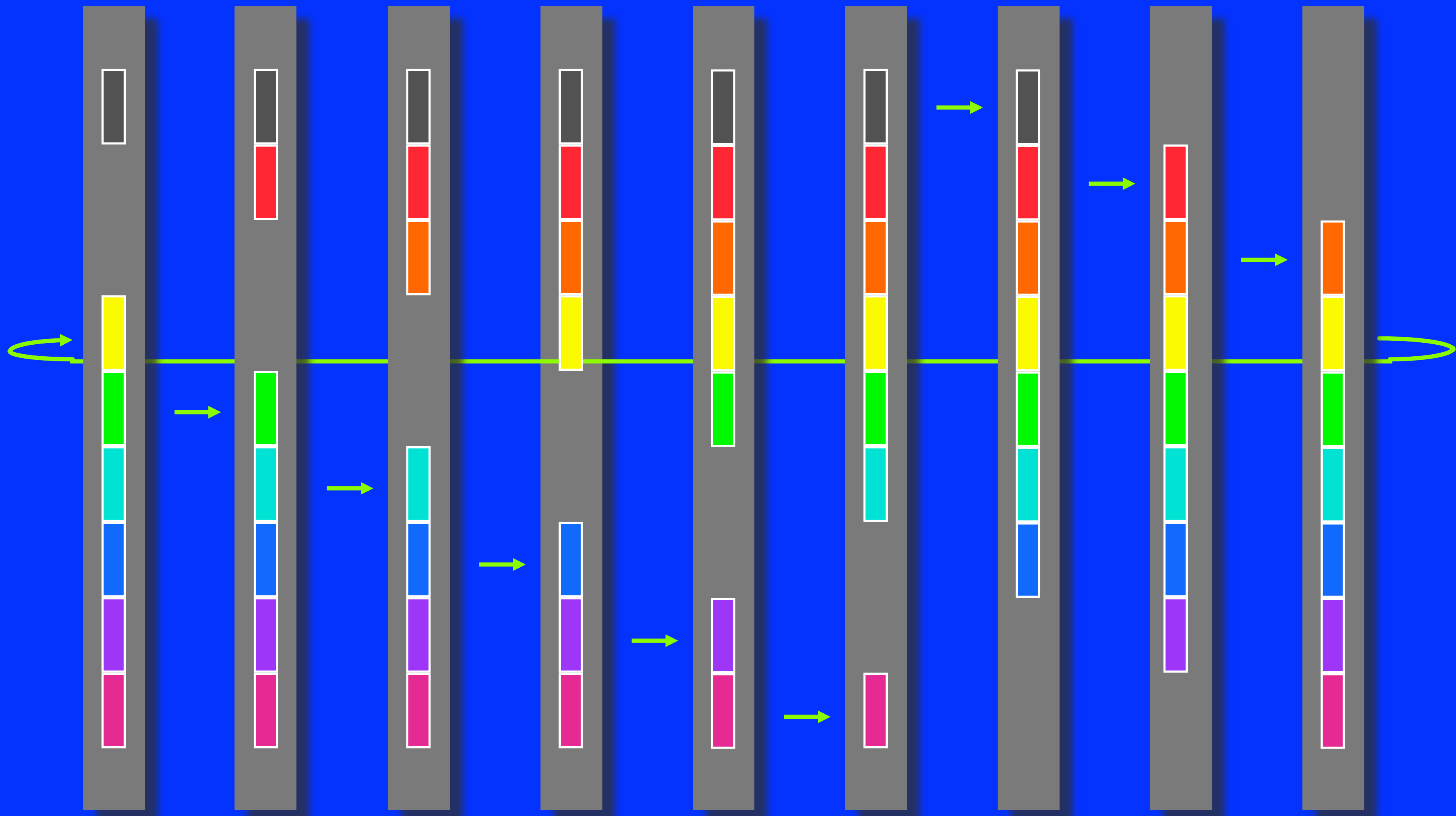


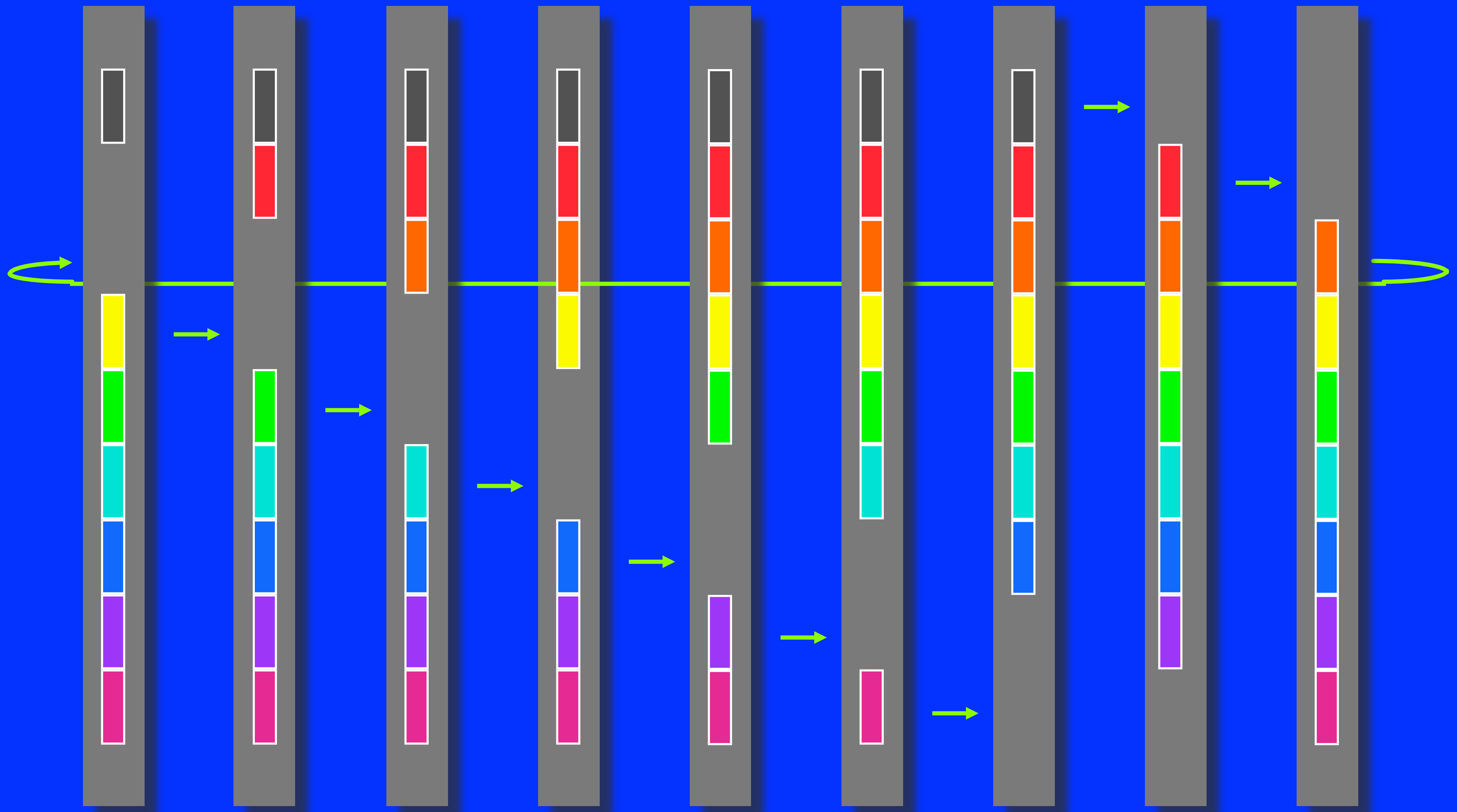




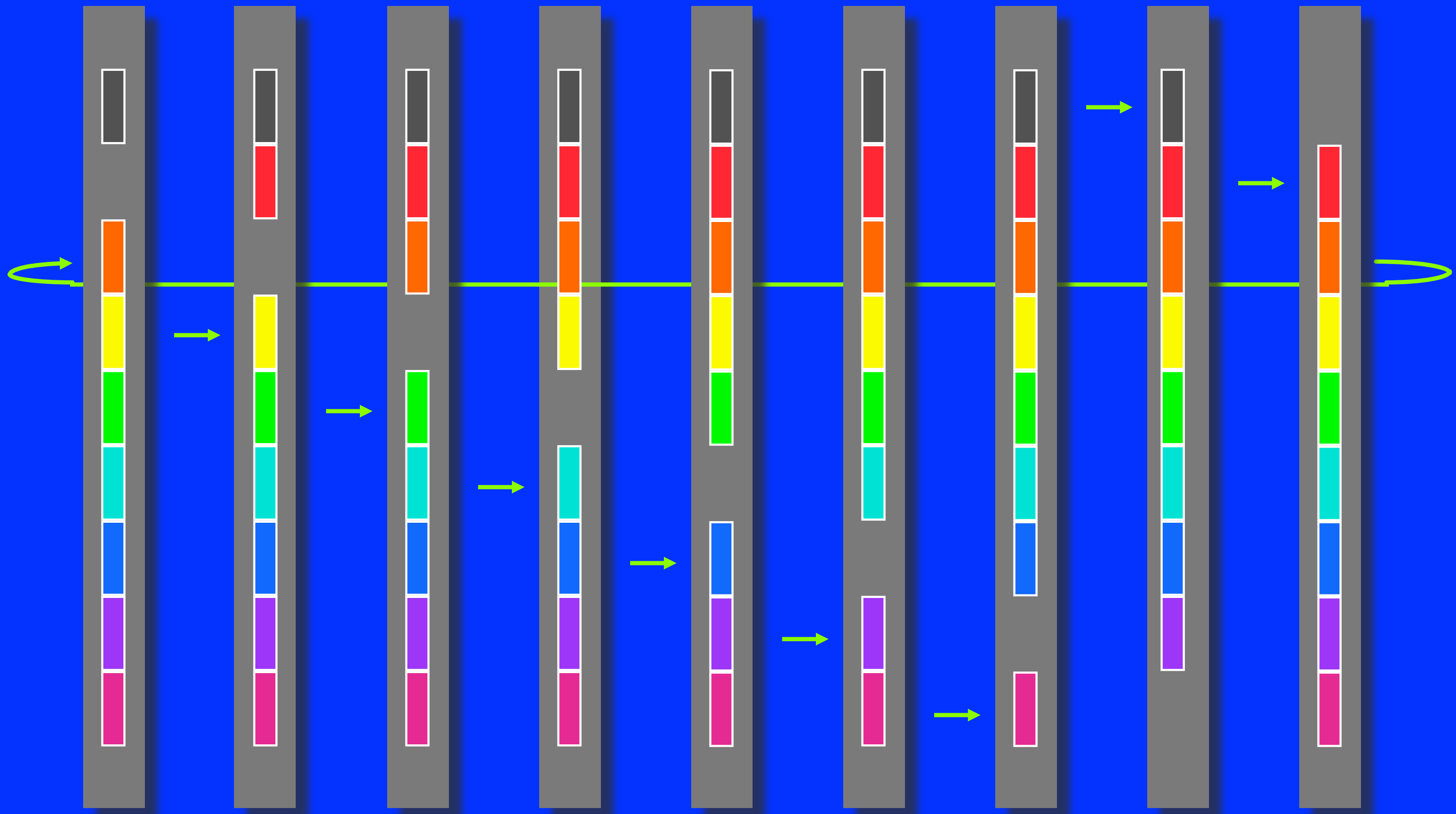


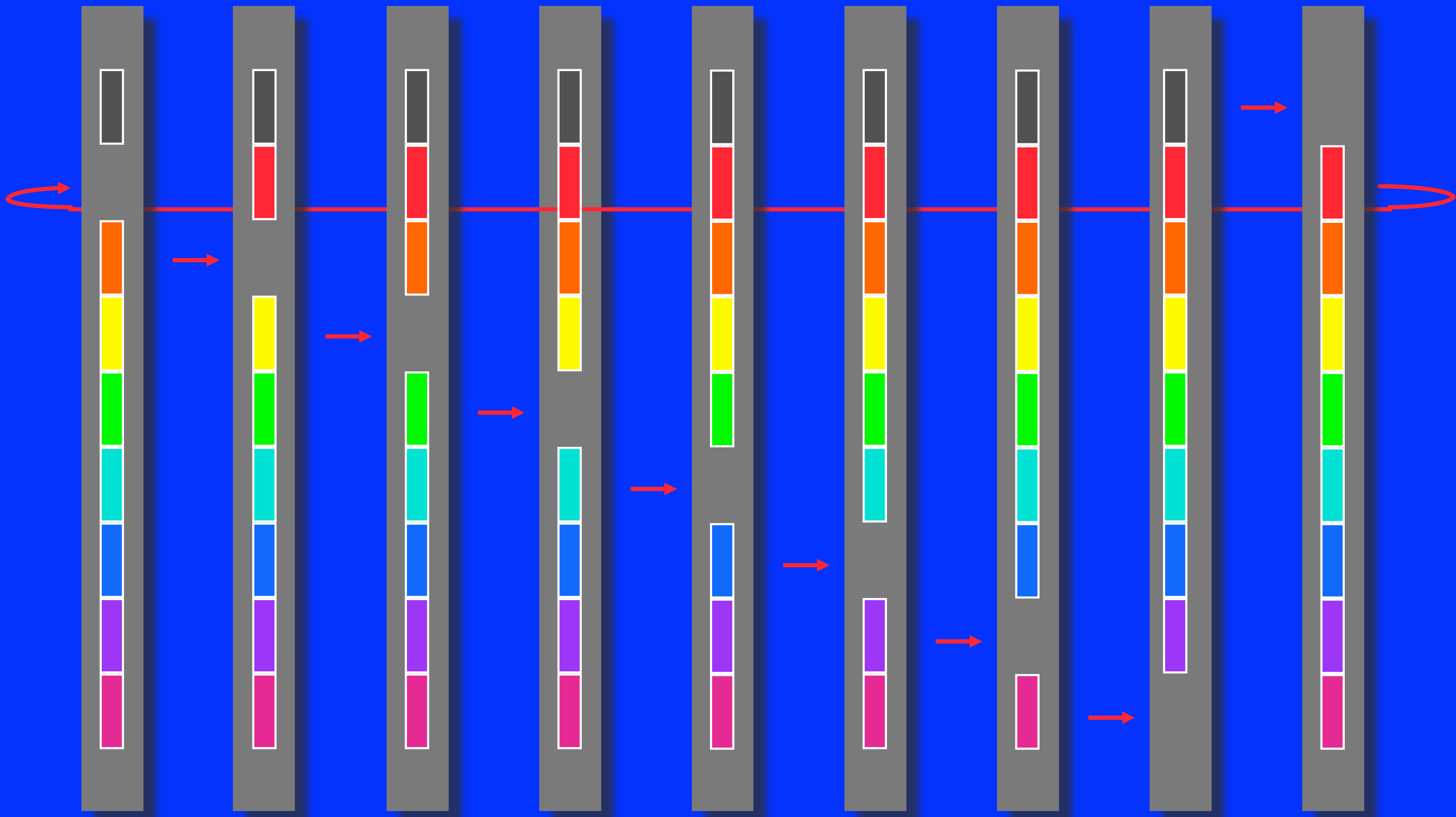


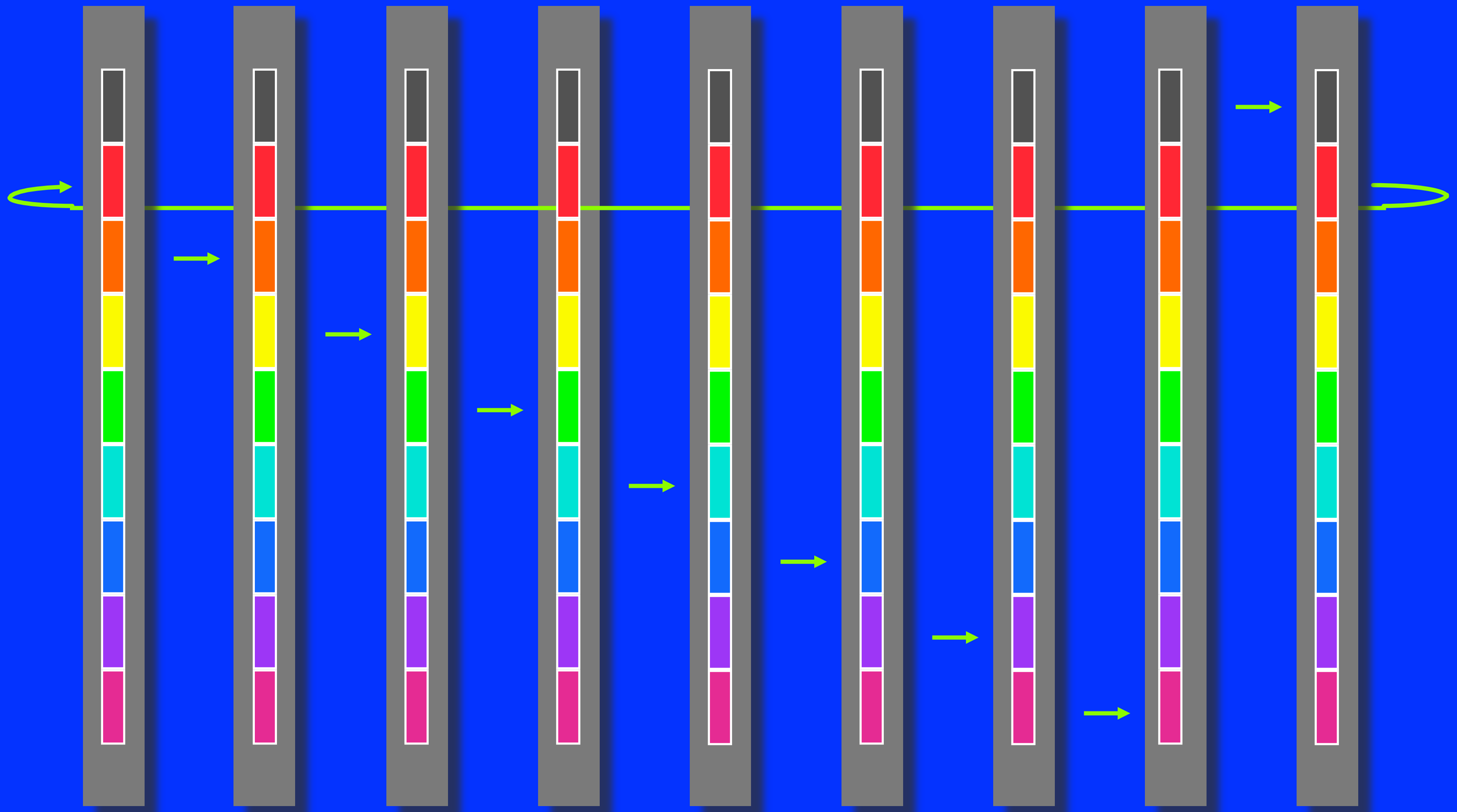


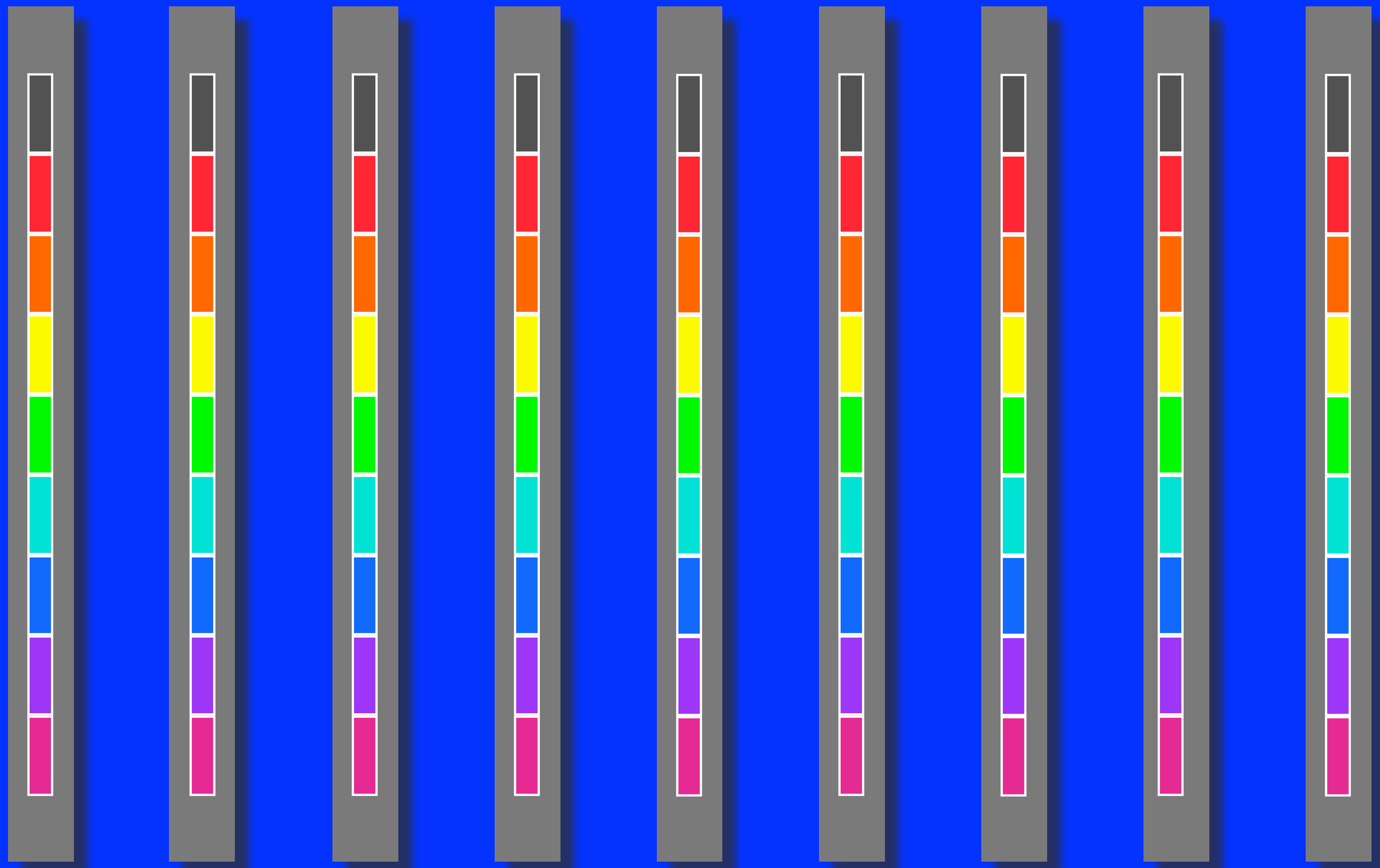


264









Cost of bucket Allgather

$$\boxed{(p-1)}\ \boxed{\left(\alpha + \frac{n}{p}\beta\right)}$$

number of steps

cost per steps

$$= (p-1)\alpha + \frac{p-1}{p} n\beta$$

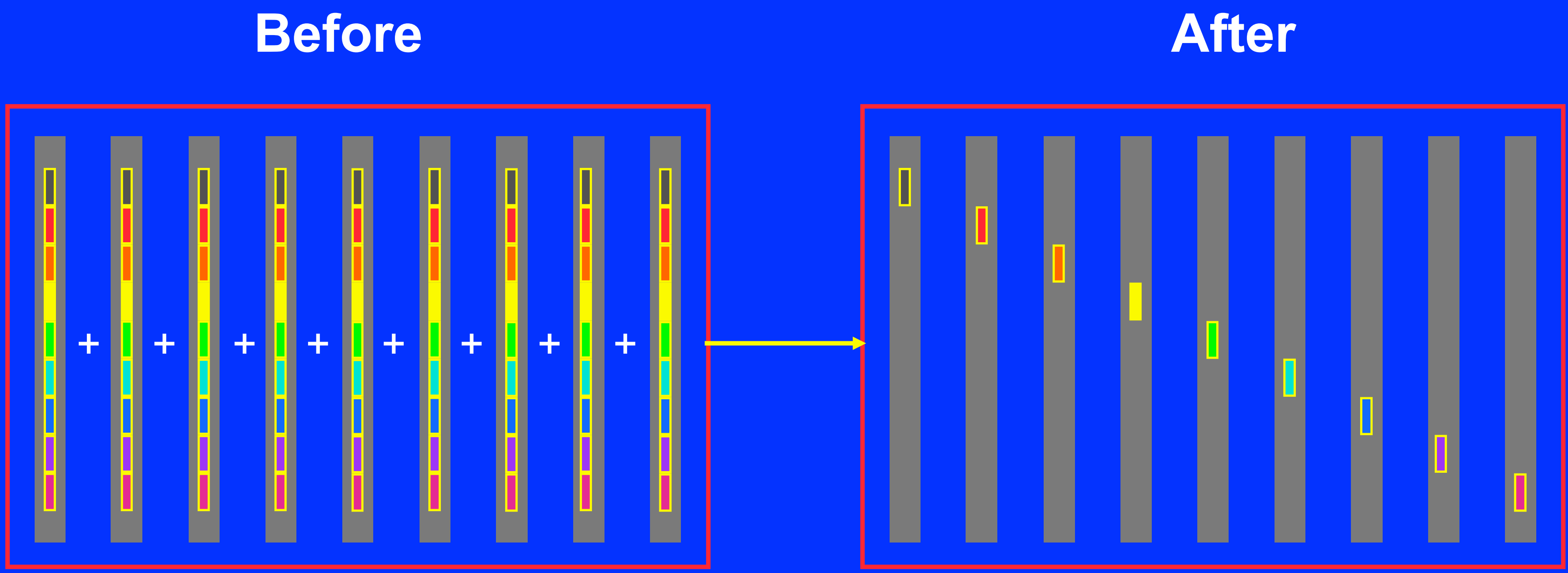
Cost of bucket Allgather

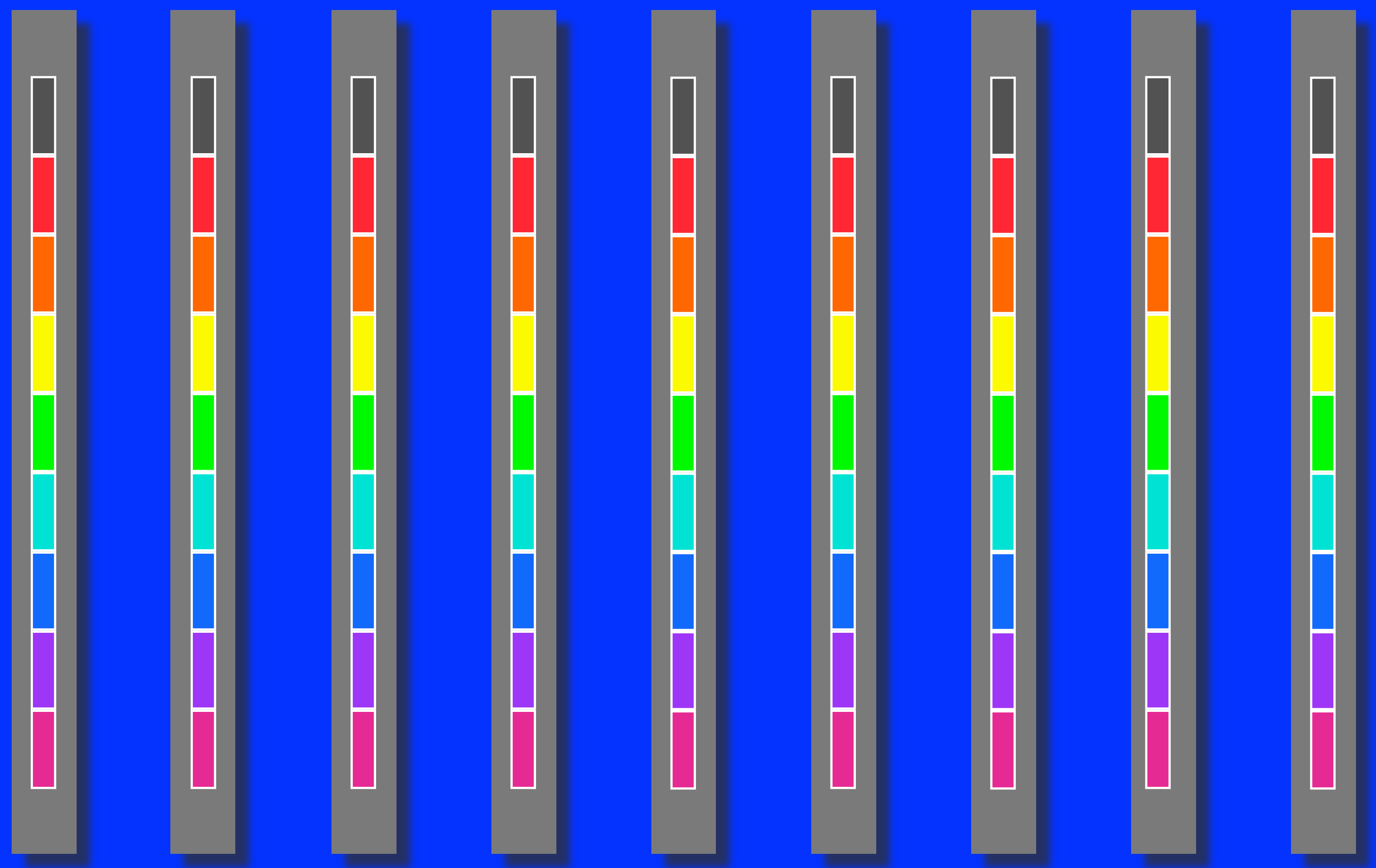
$$\underbrace{(p-1)}_{\text{number of steps}}\underbrace{\left(\alpha+\frac{n}{p}\beta\right)}_{\text{cost per steps}}$$

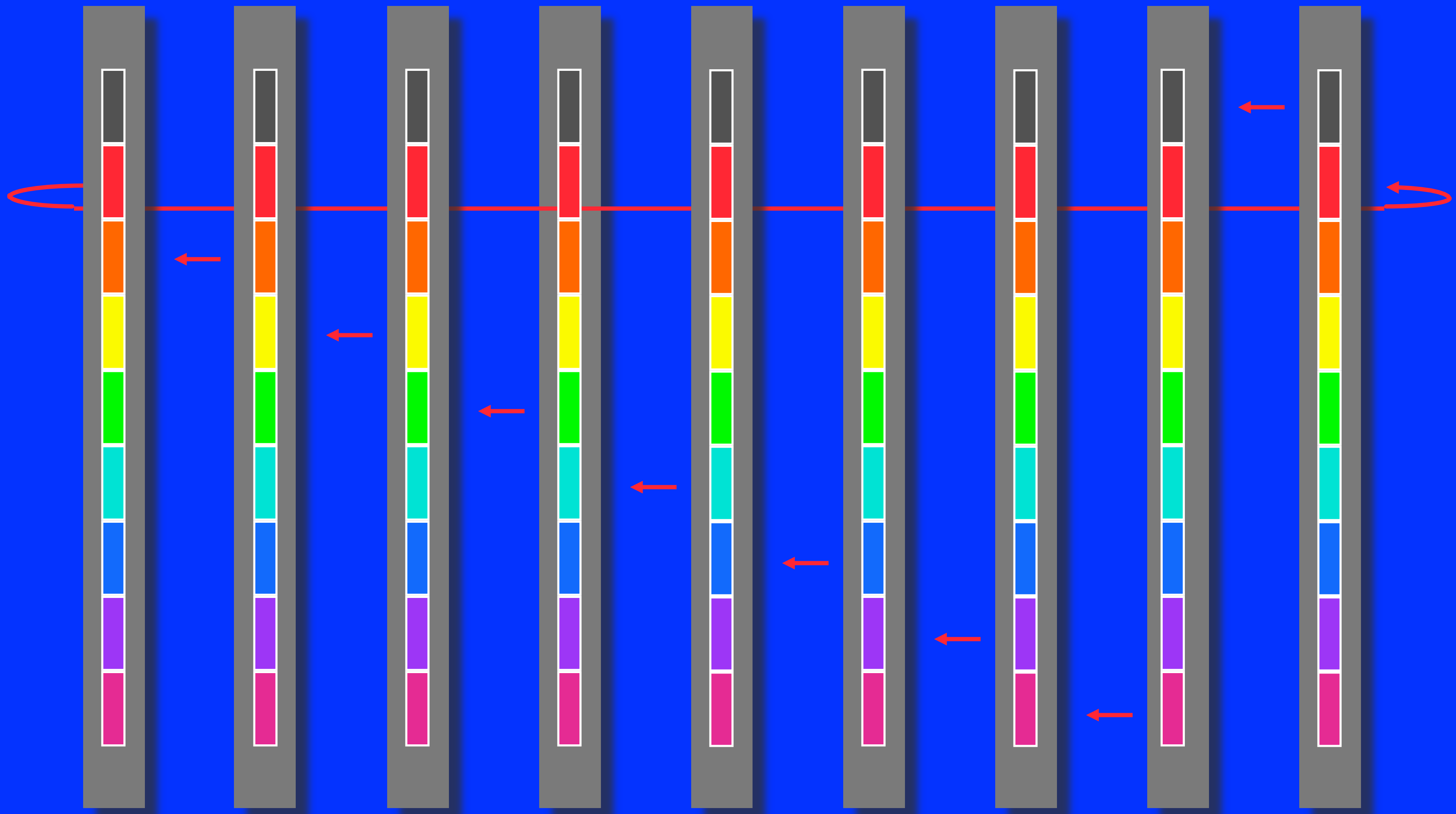
$$= (p-1)\alpha + \frac{p-1}{p} n\beta$$

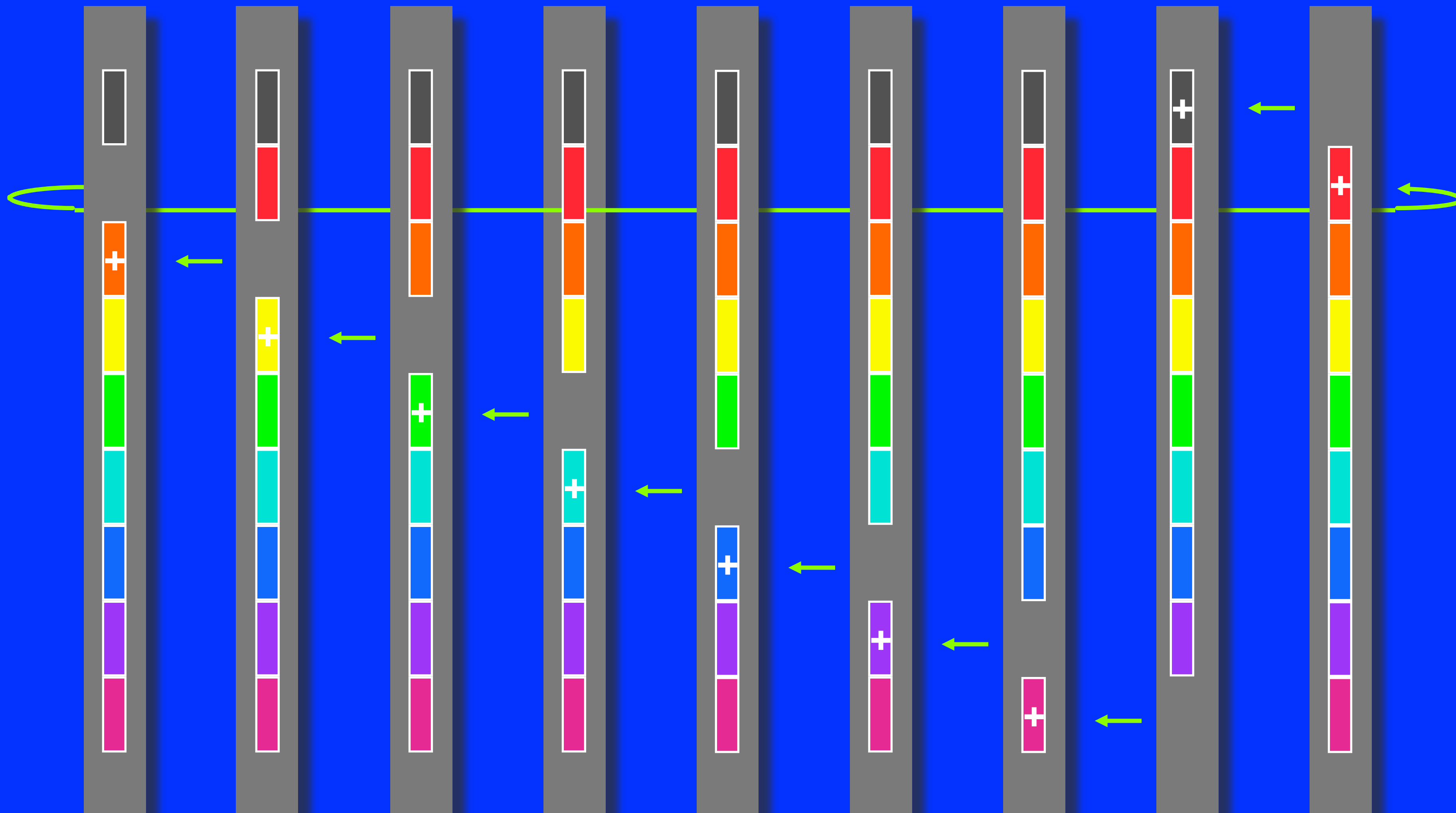
Notice: attains lower bound for bandwidth component

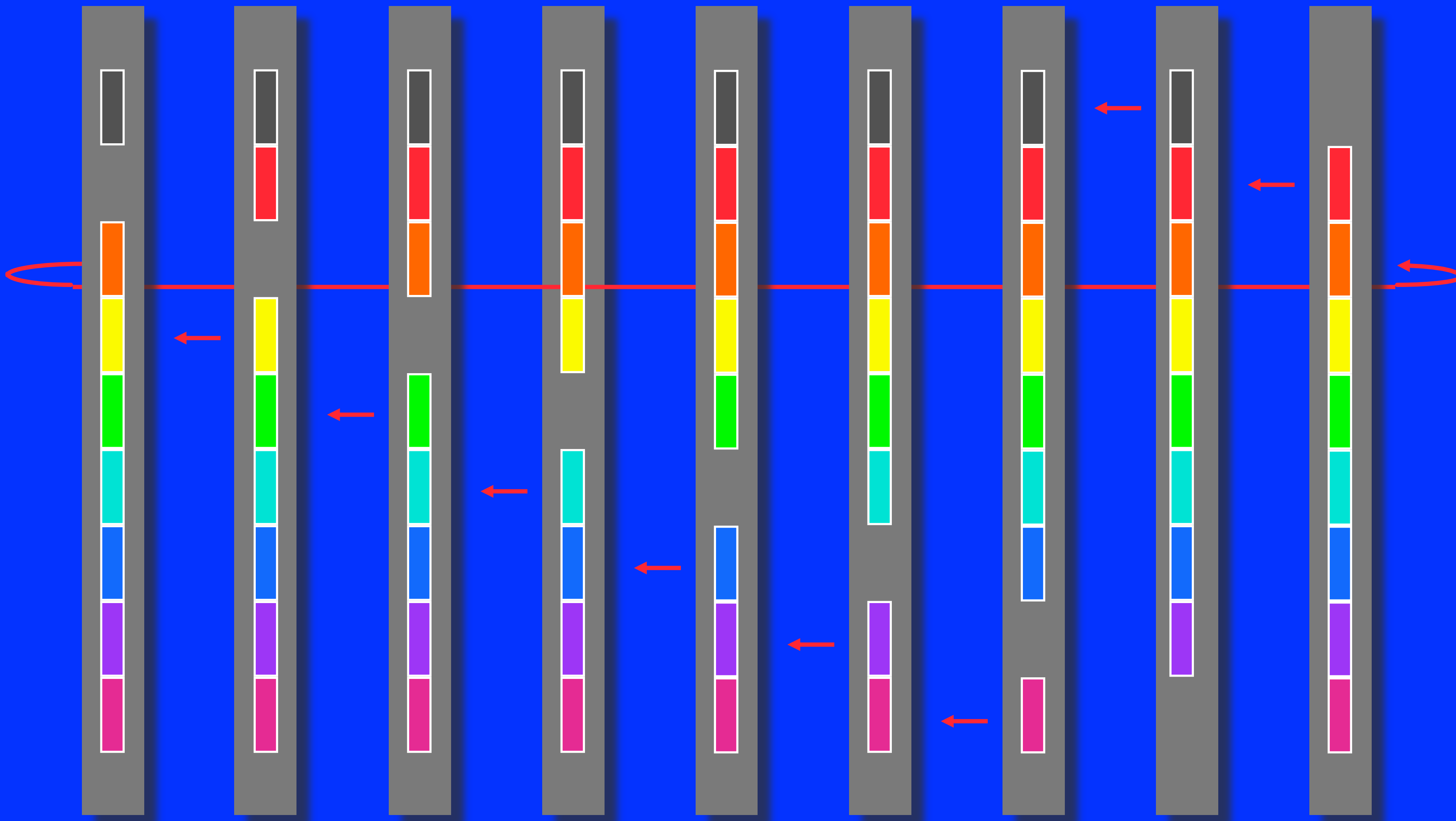
Reduce-scatter

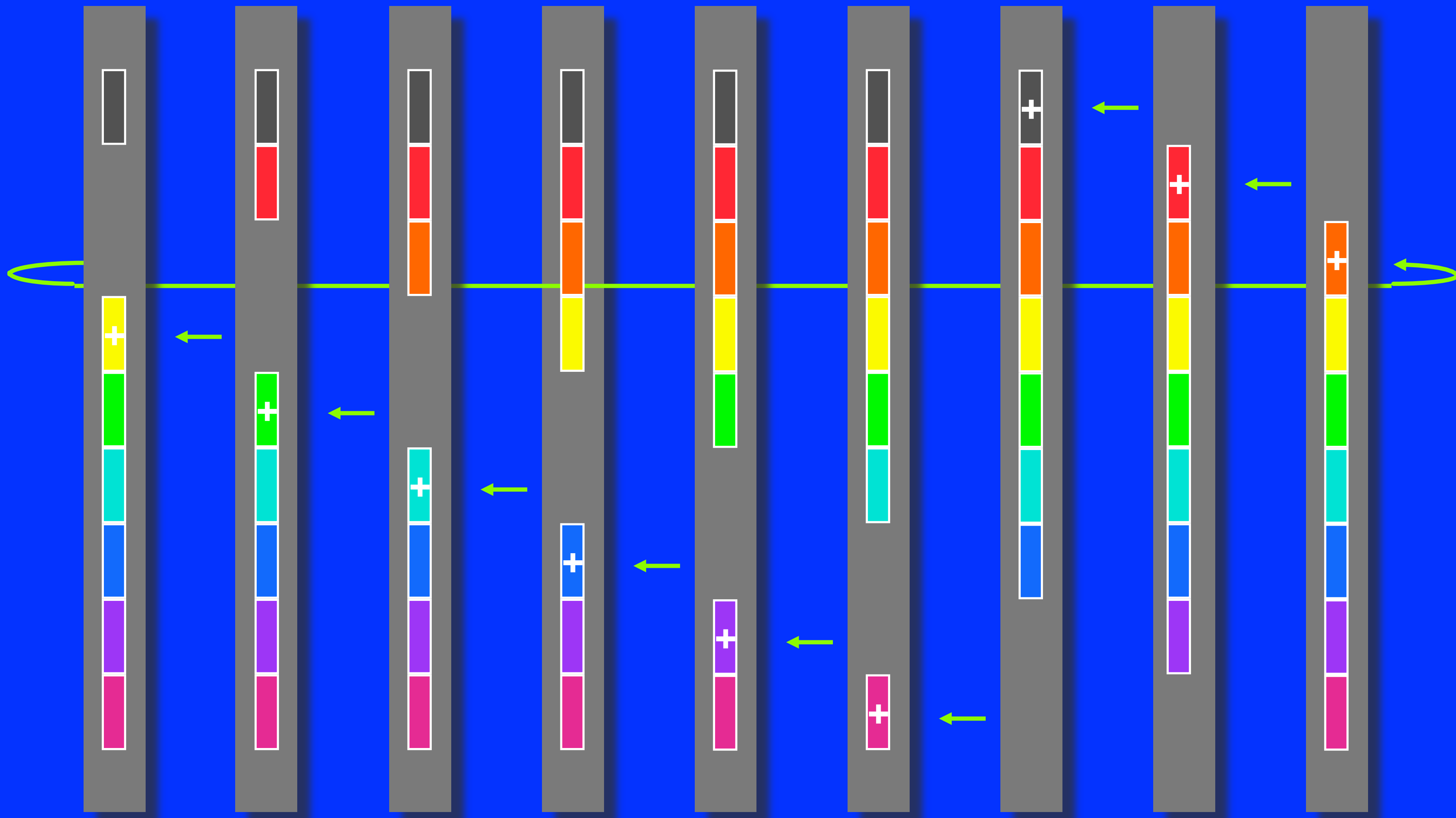


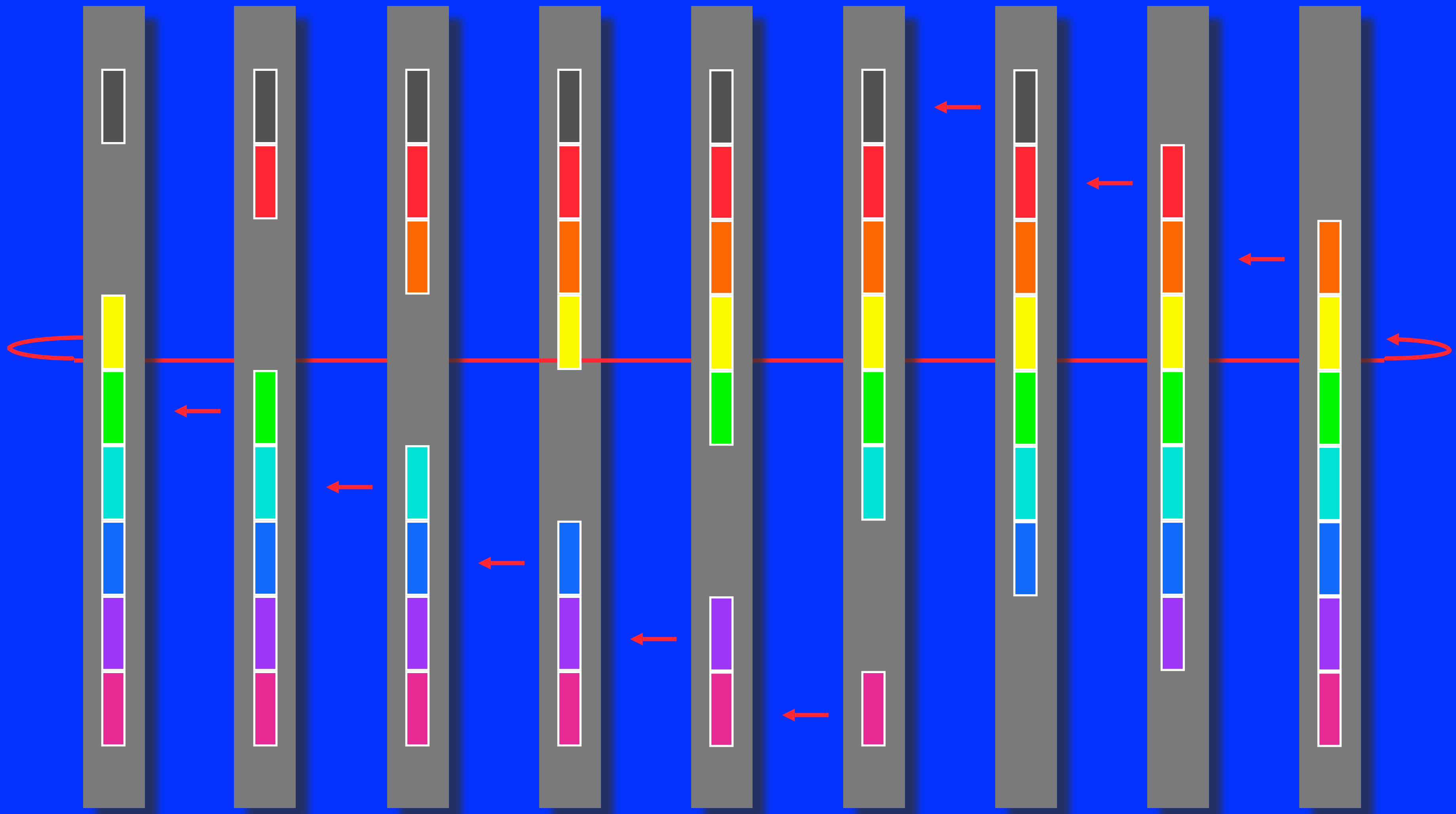


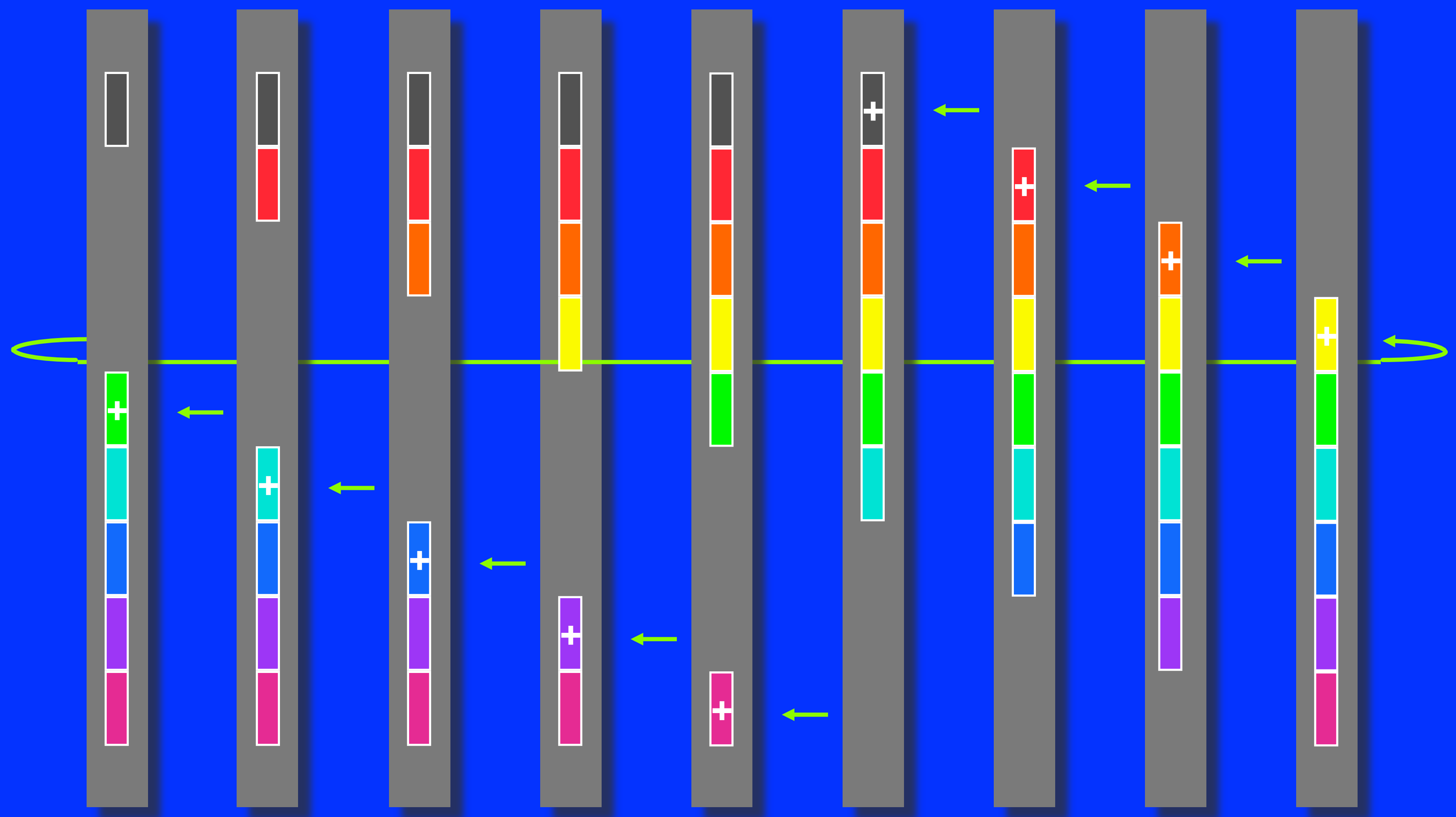


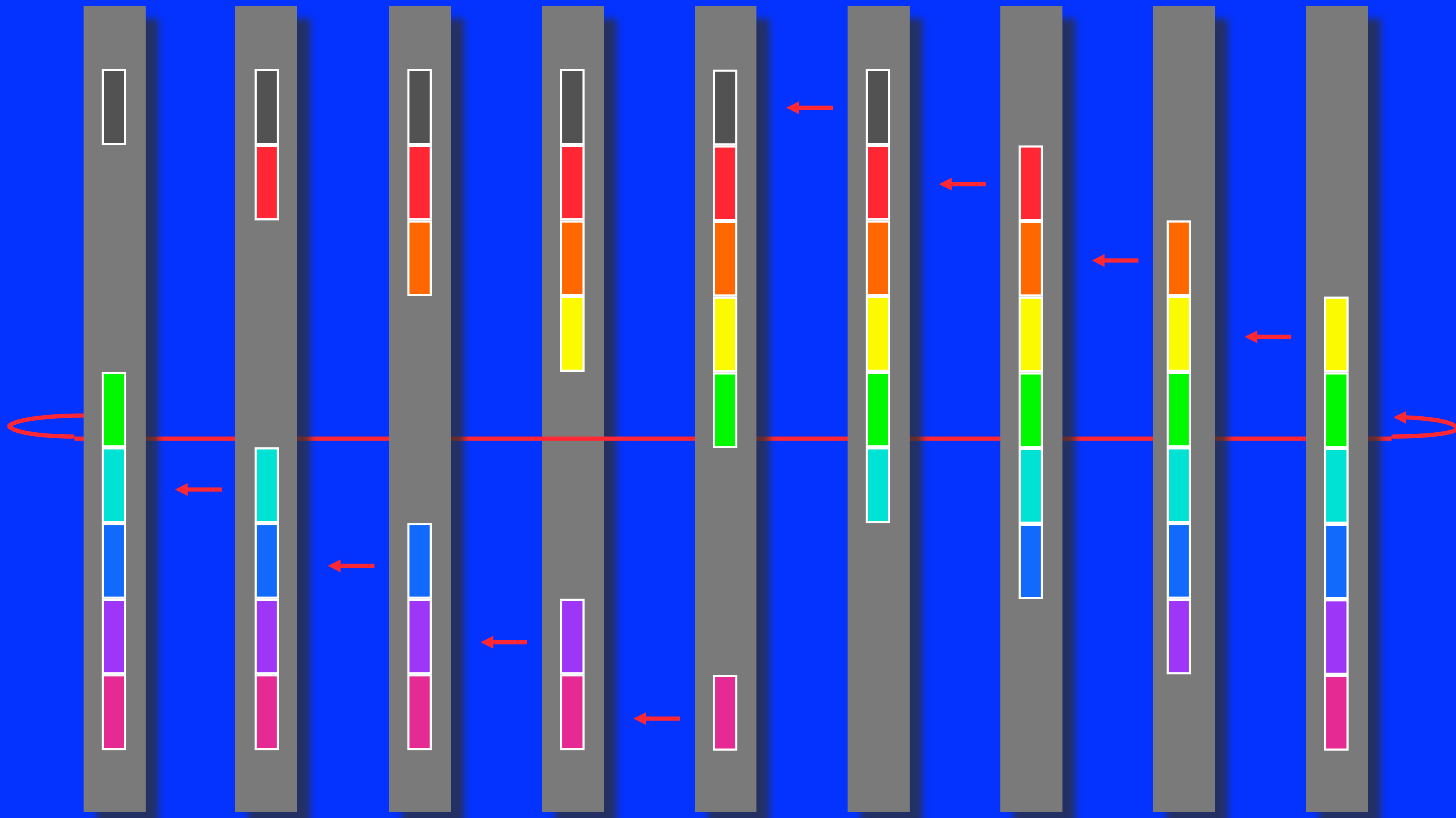


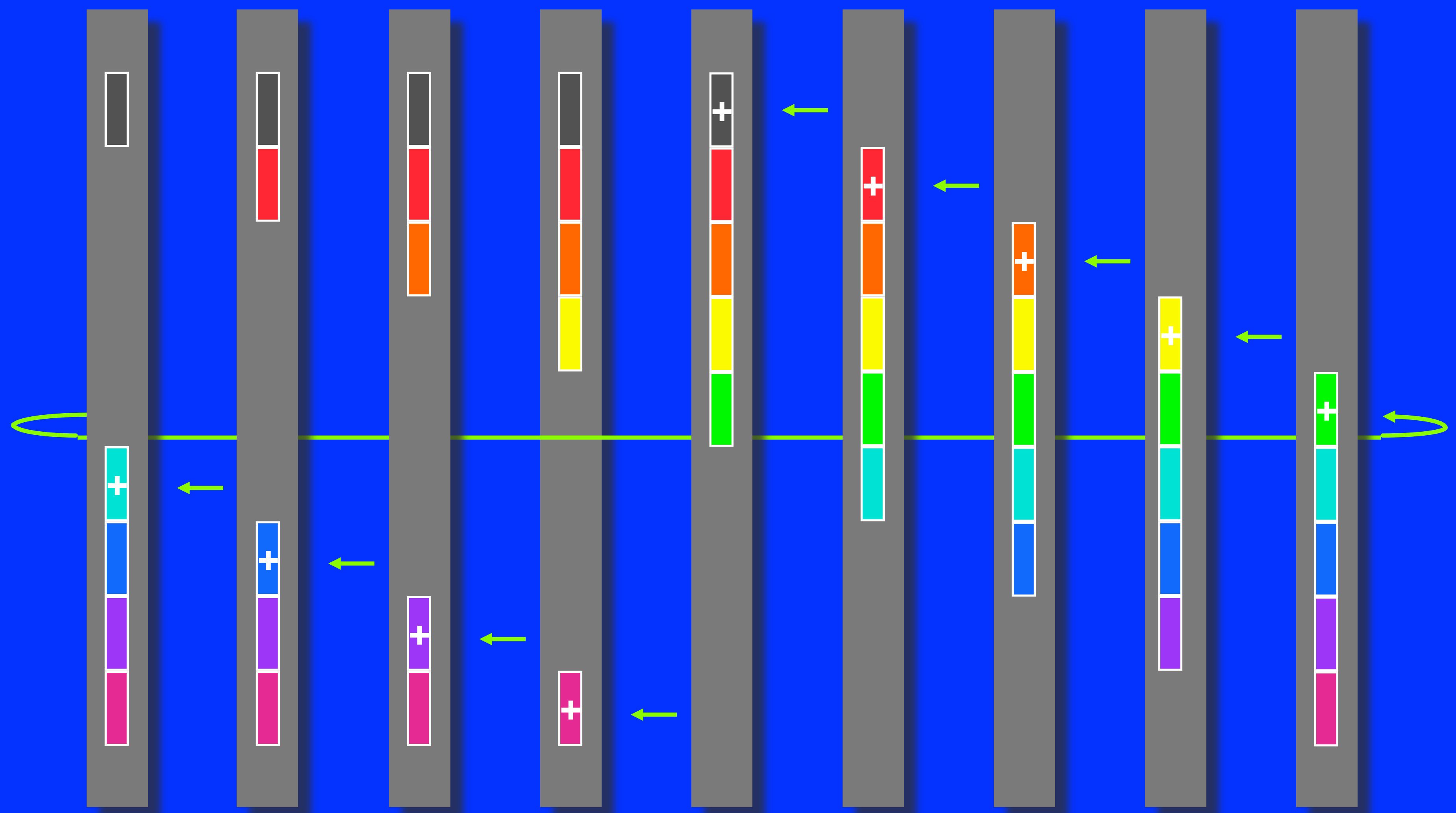


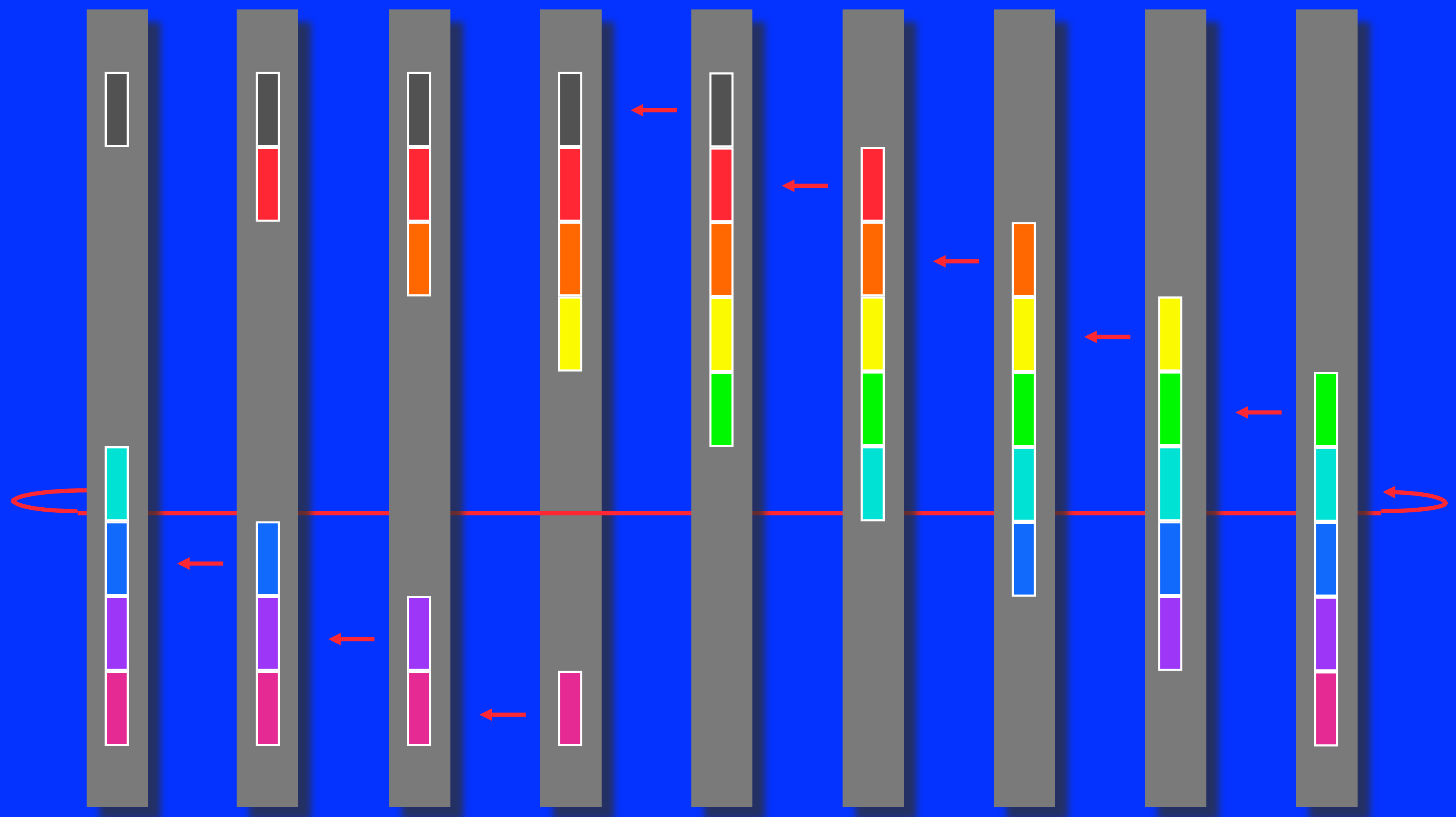


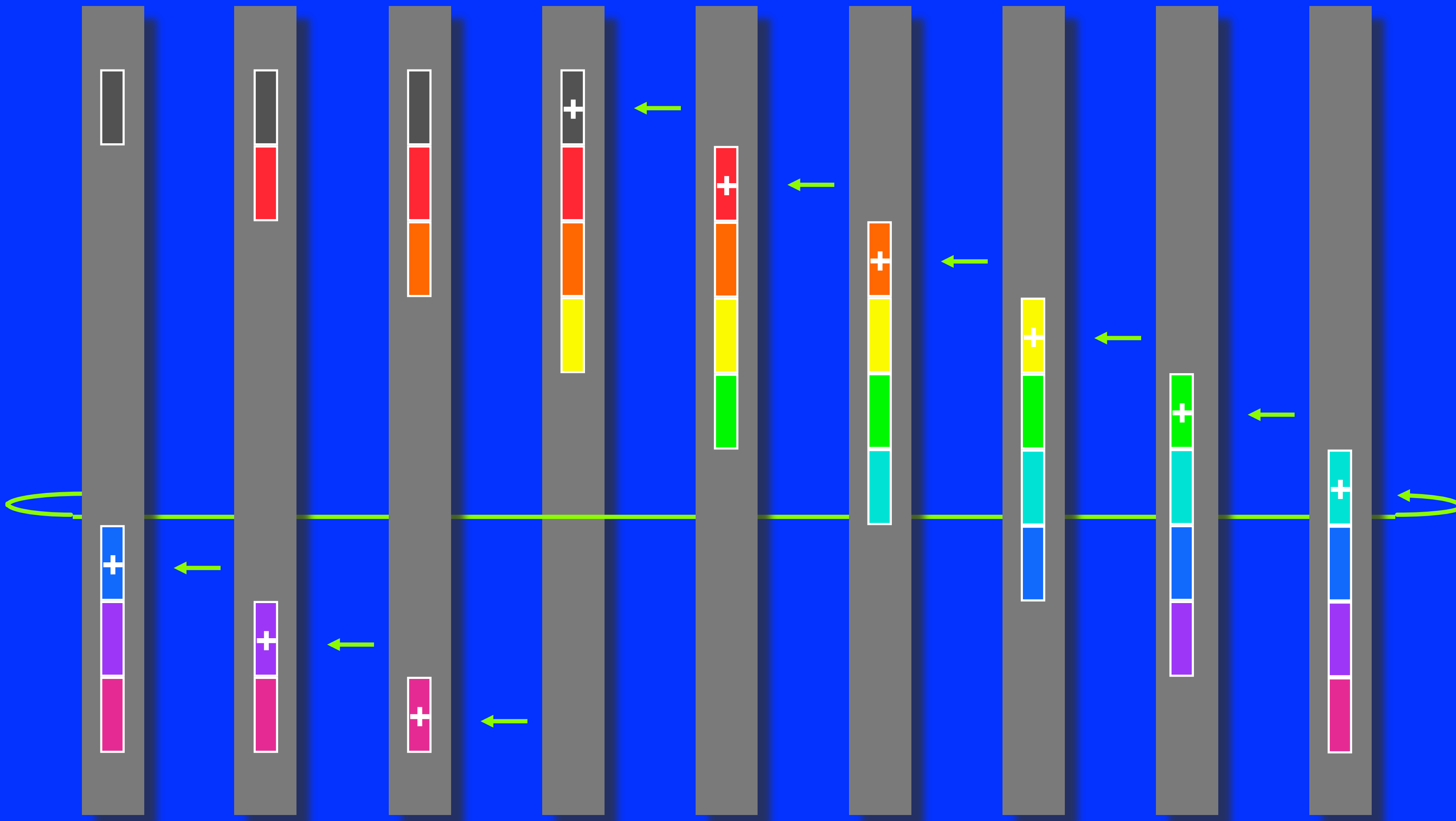


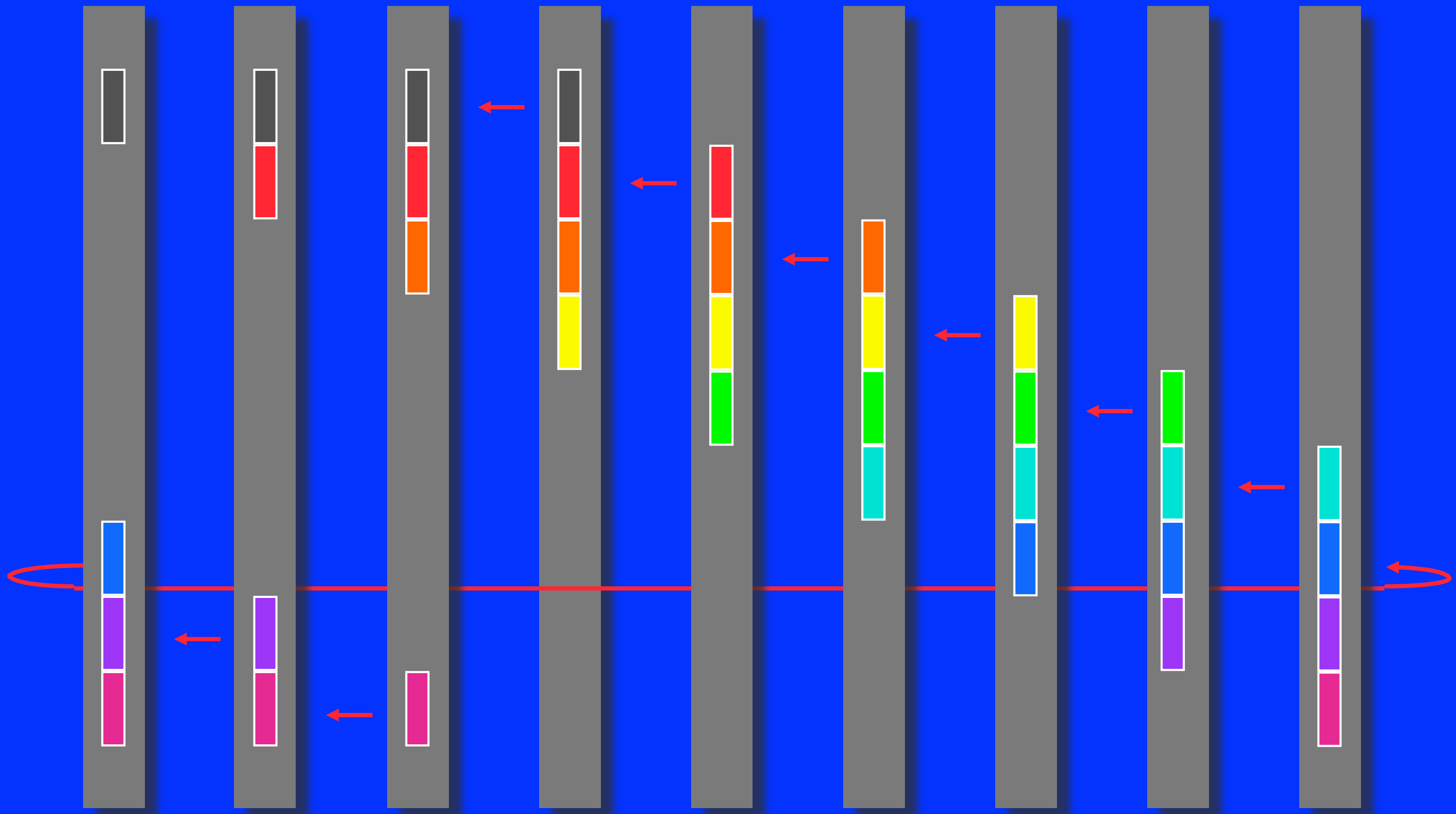


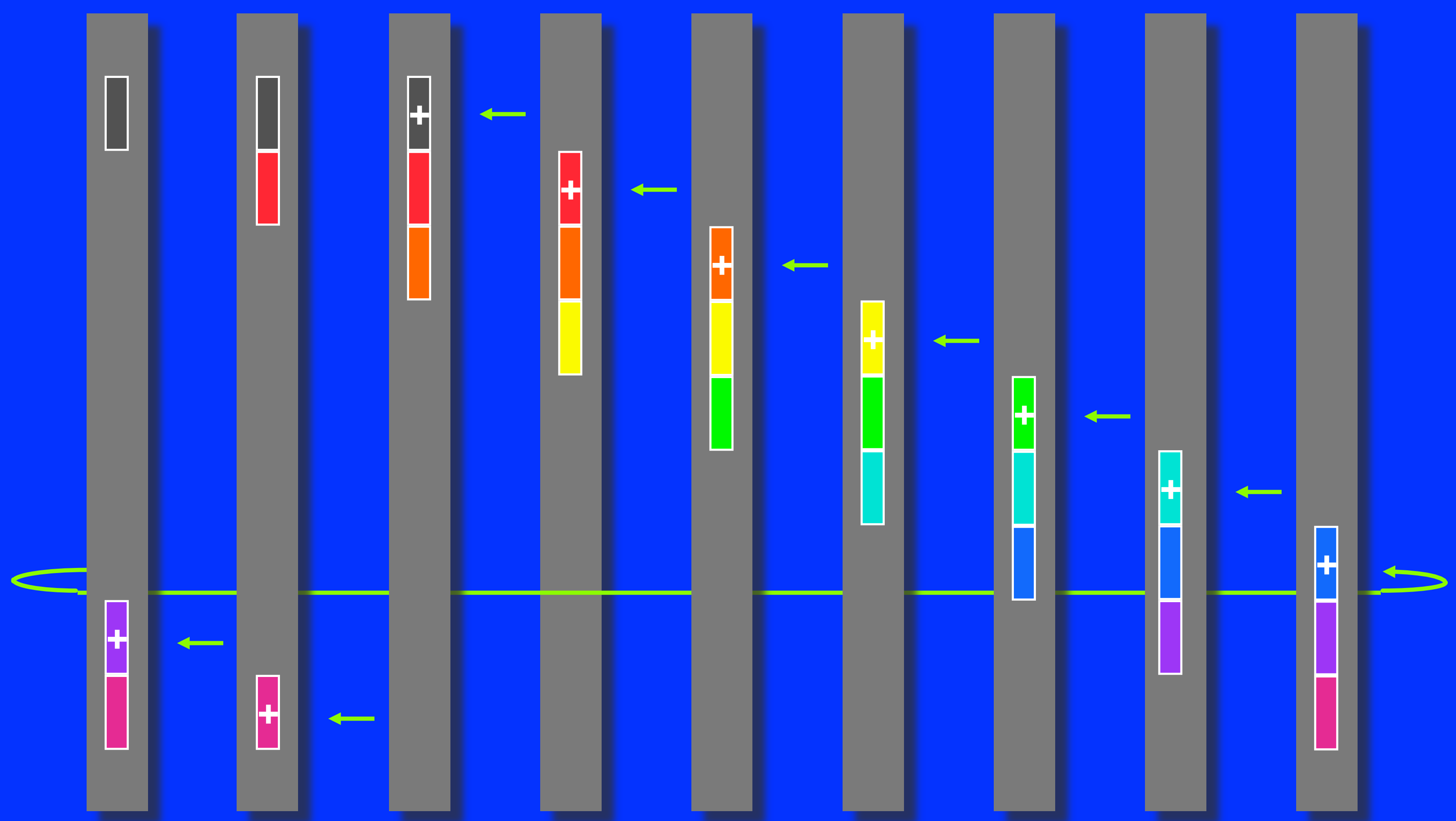


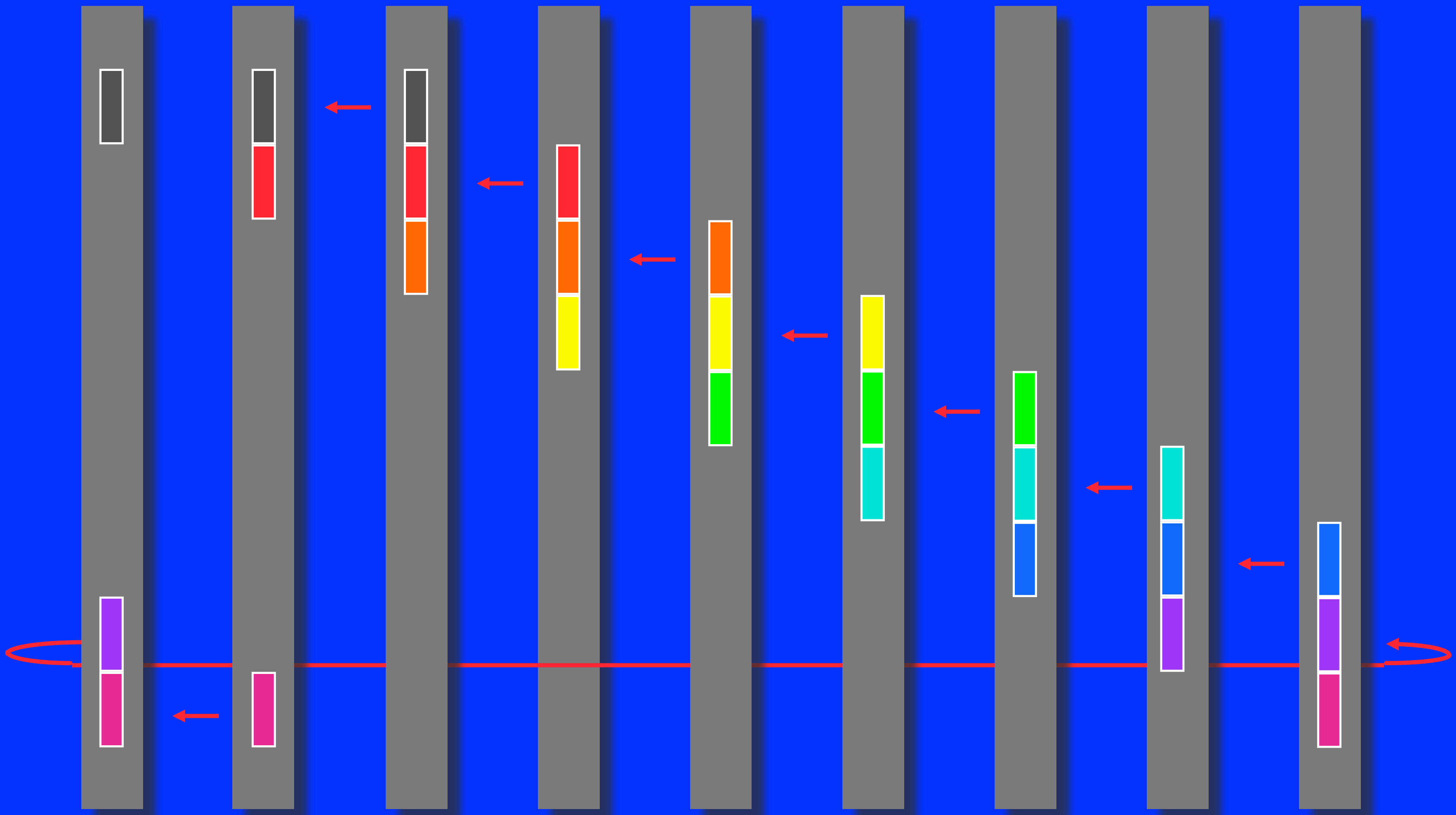


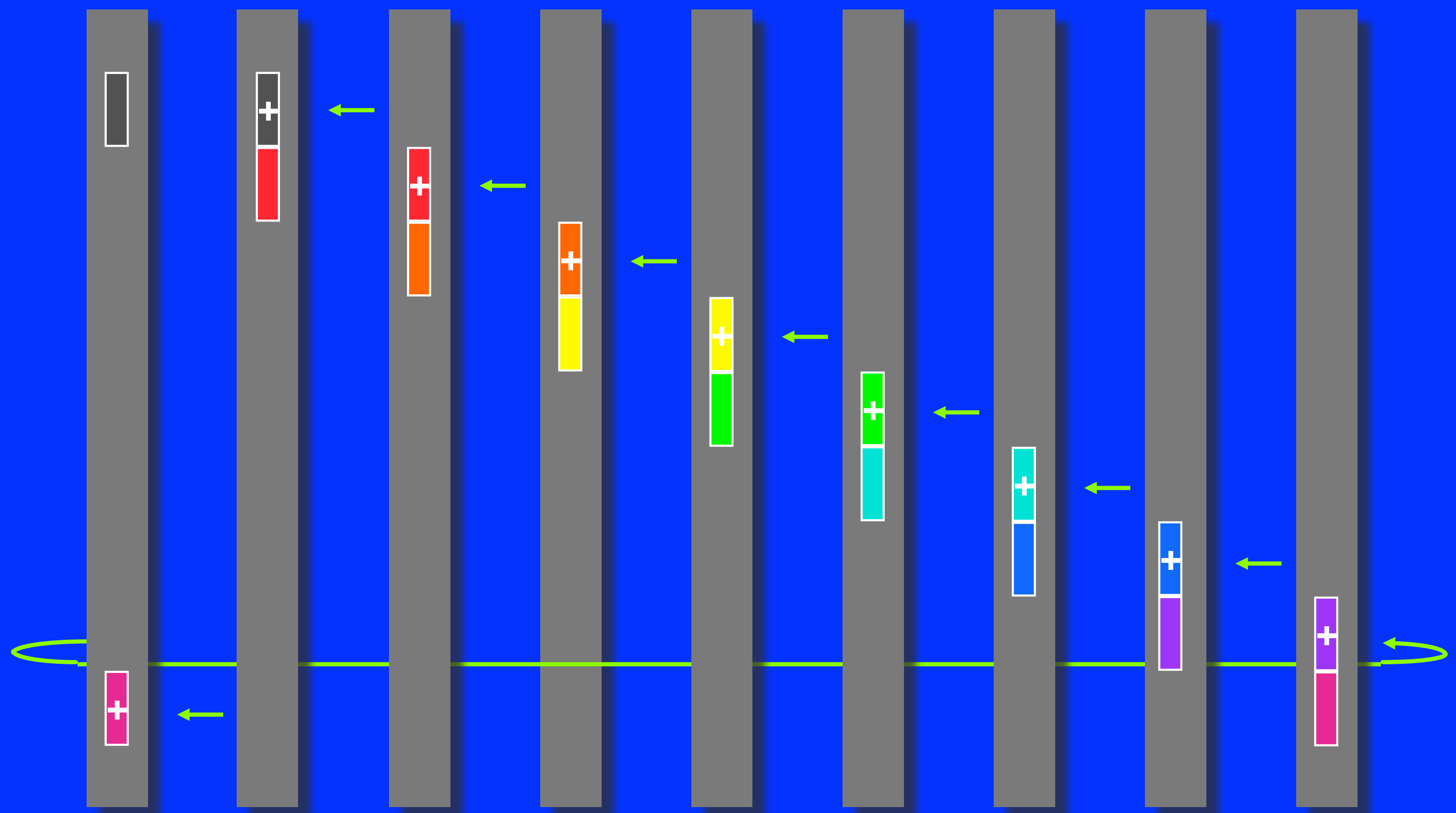


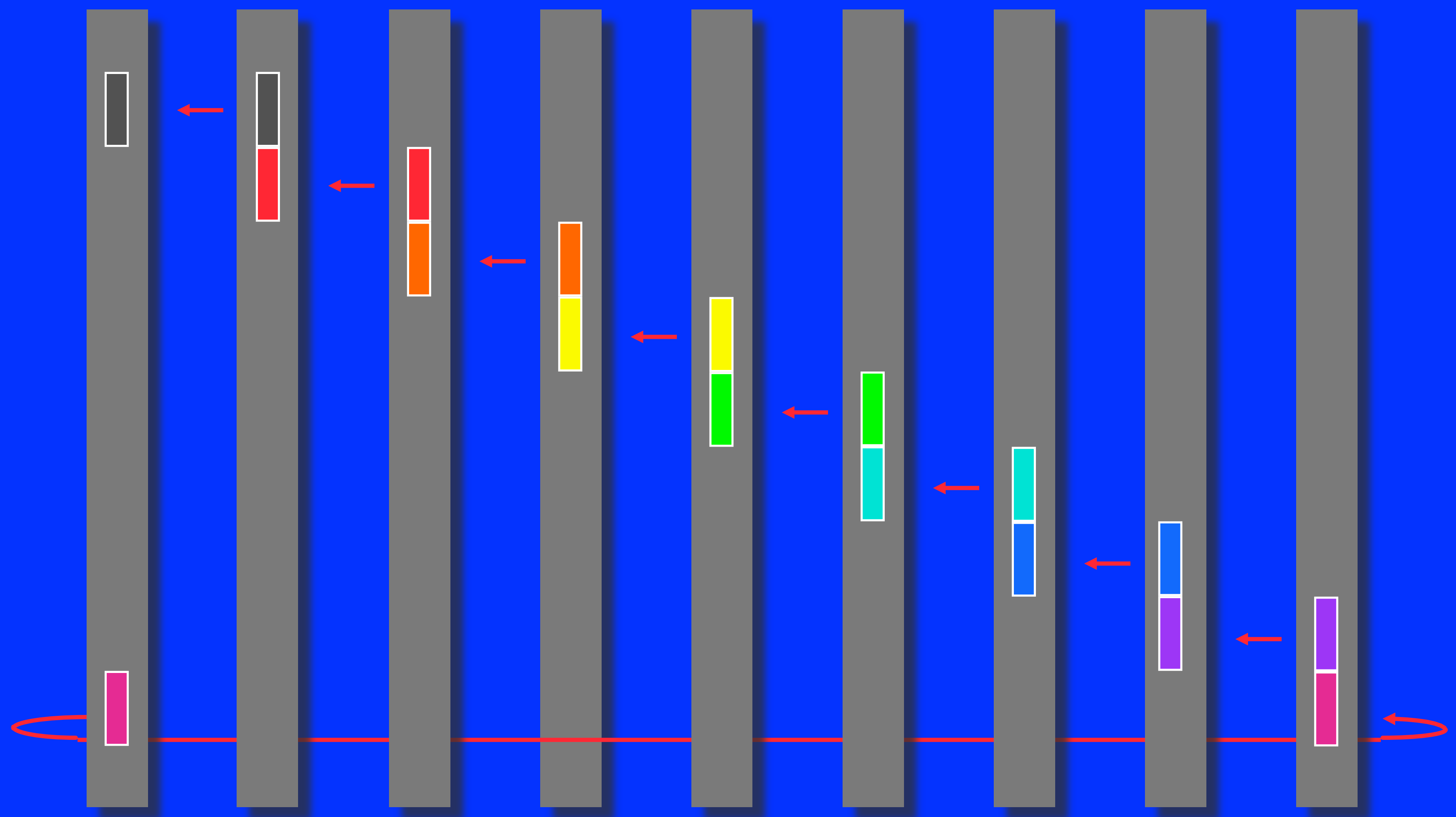


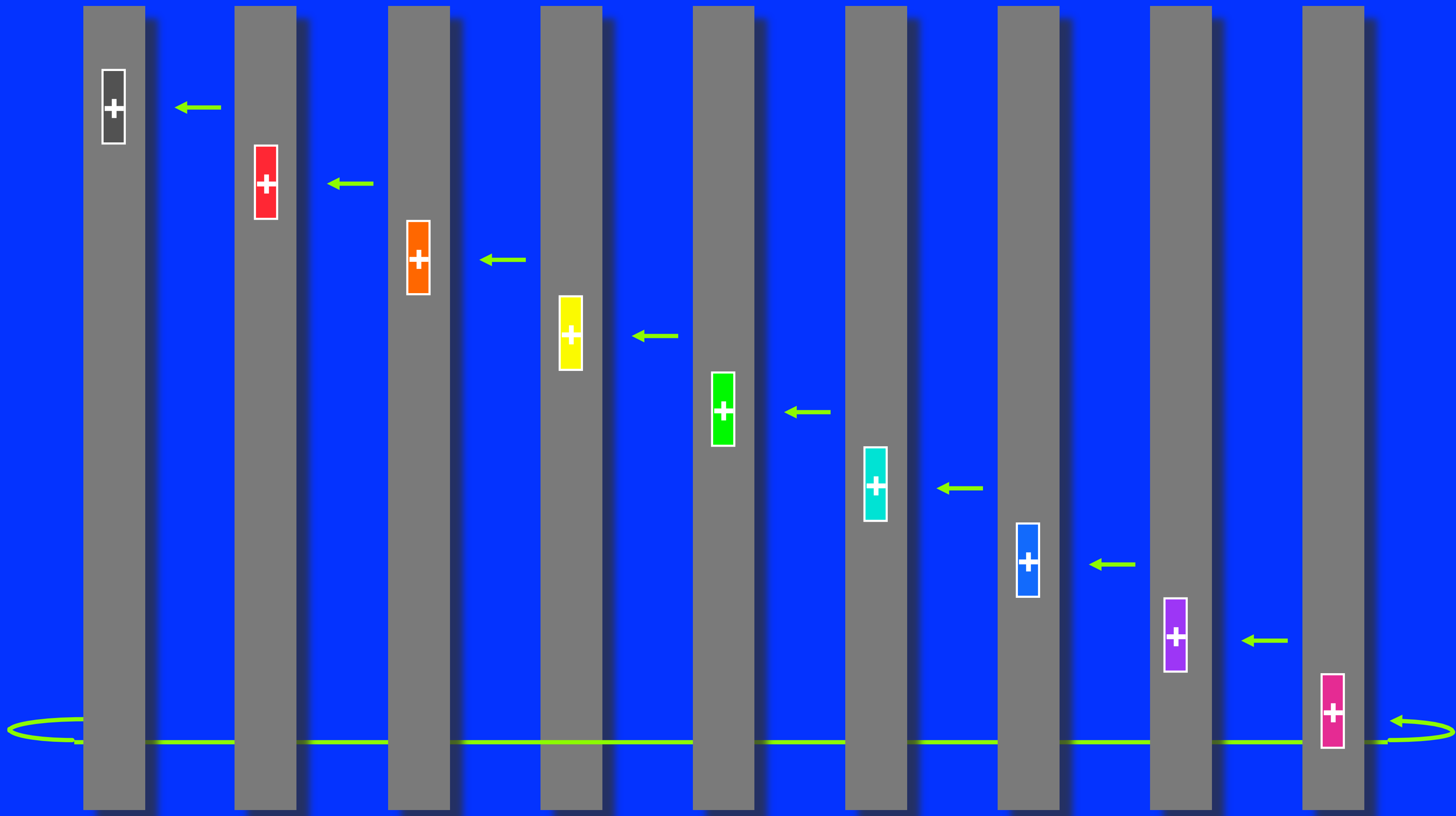


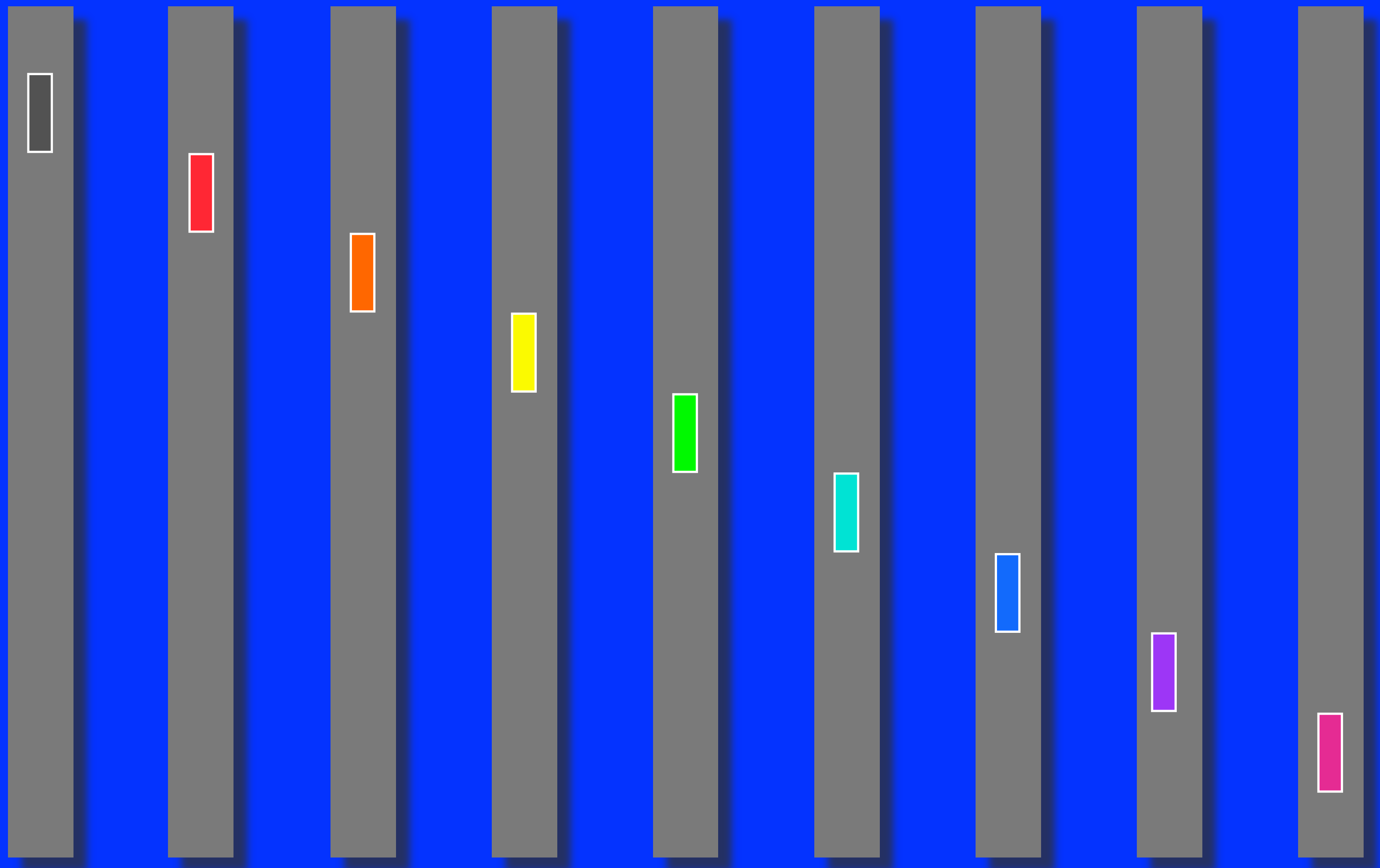












Cost of bucket distributed combine

$$\underbrace{(p-1)}_{\text{number of steps}}\underbrace{\left(\alpha+\frac{n}{p}\beta+\frac{n}{p}\beta\right)}_{\text{cost per steps}}$$

$$=$$

$$(p-1)\alpha+\frac{p-1}{p}n\beta+\frac{p-1}{p}n\gamma$$

Cost of bucket Reduce-scatter

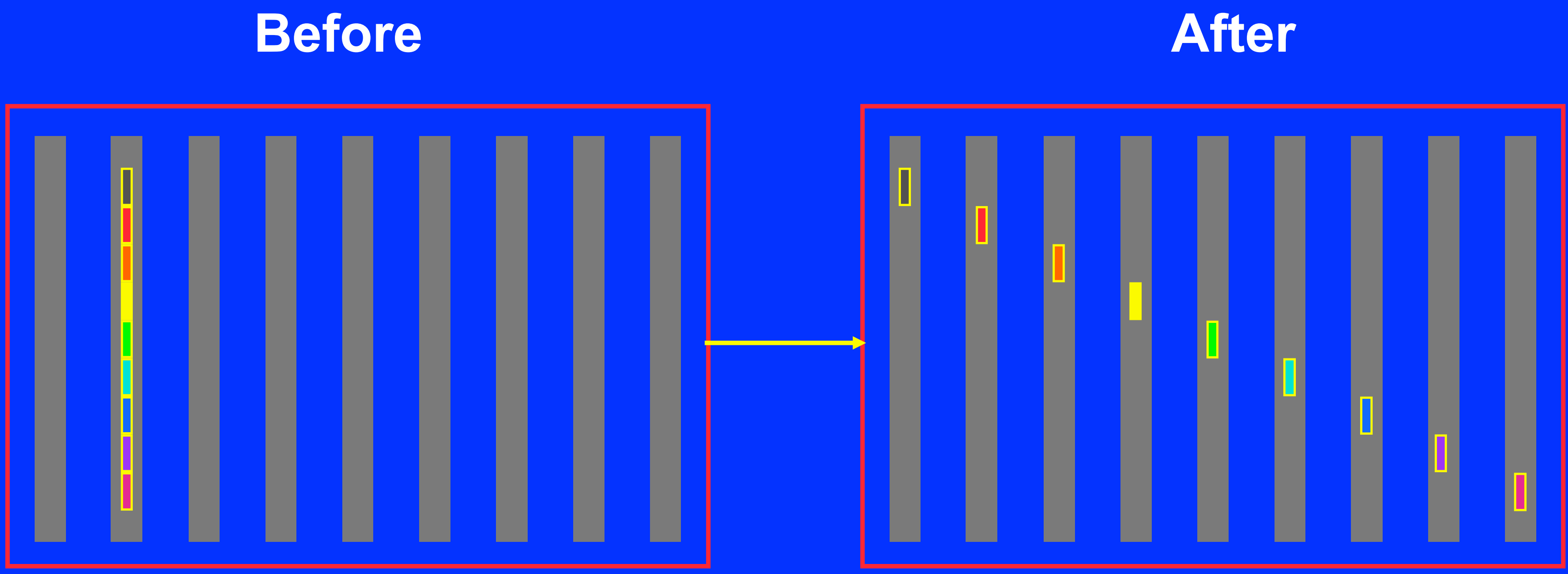
$$\underbrace{(p-1)}_{\text{number of steps}}\underbrace{\left(\alpha + \frac{n}{p}\beta + \frac{n}{p}\gamma\right)}_{\text{cost per steps}}$$

$$= (p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Notice: attains lower bound for bandwidth and computation component

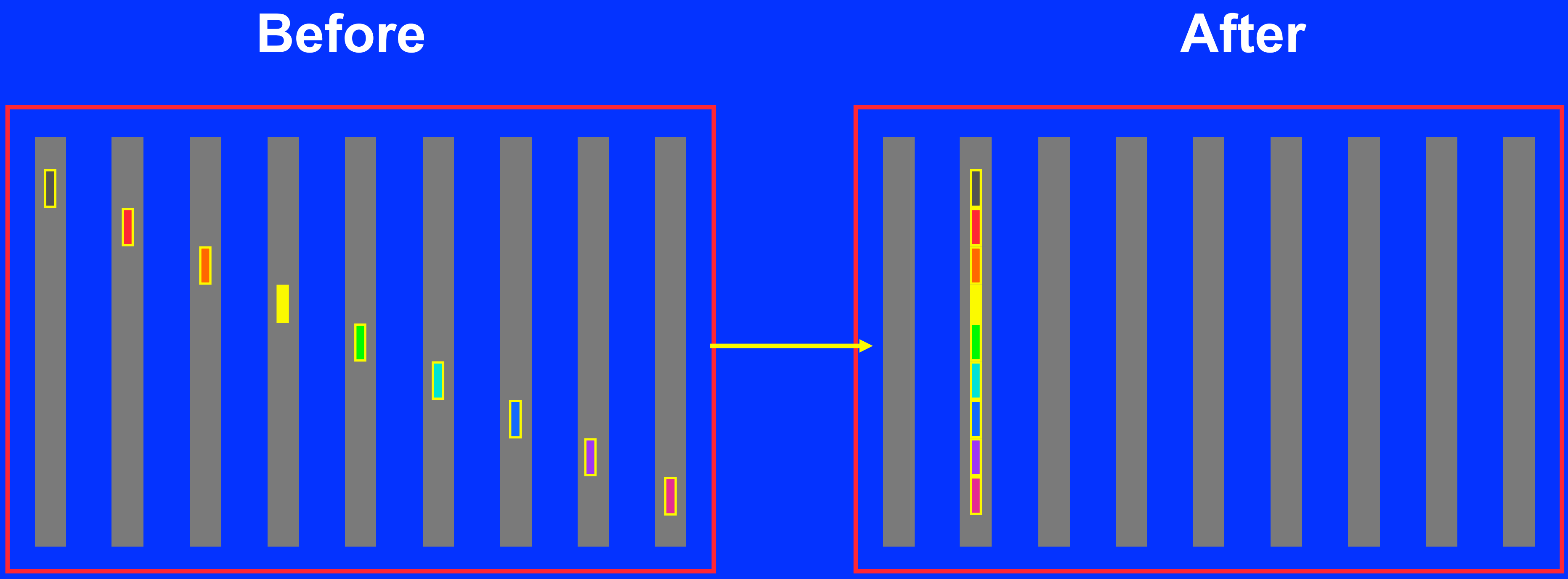
Scatter

Notice: Scatter as implemented before was optimal in latency and bandwidth components



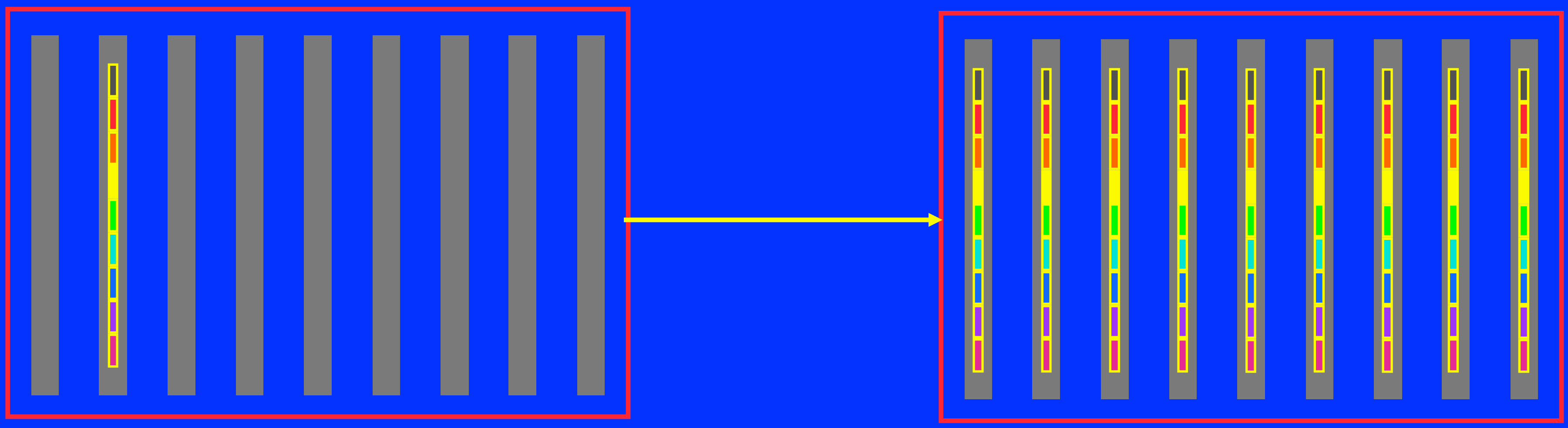
Gather

Notice: Gather as implemented before was optimal in latency and bandwidth components

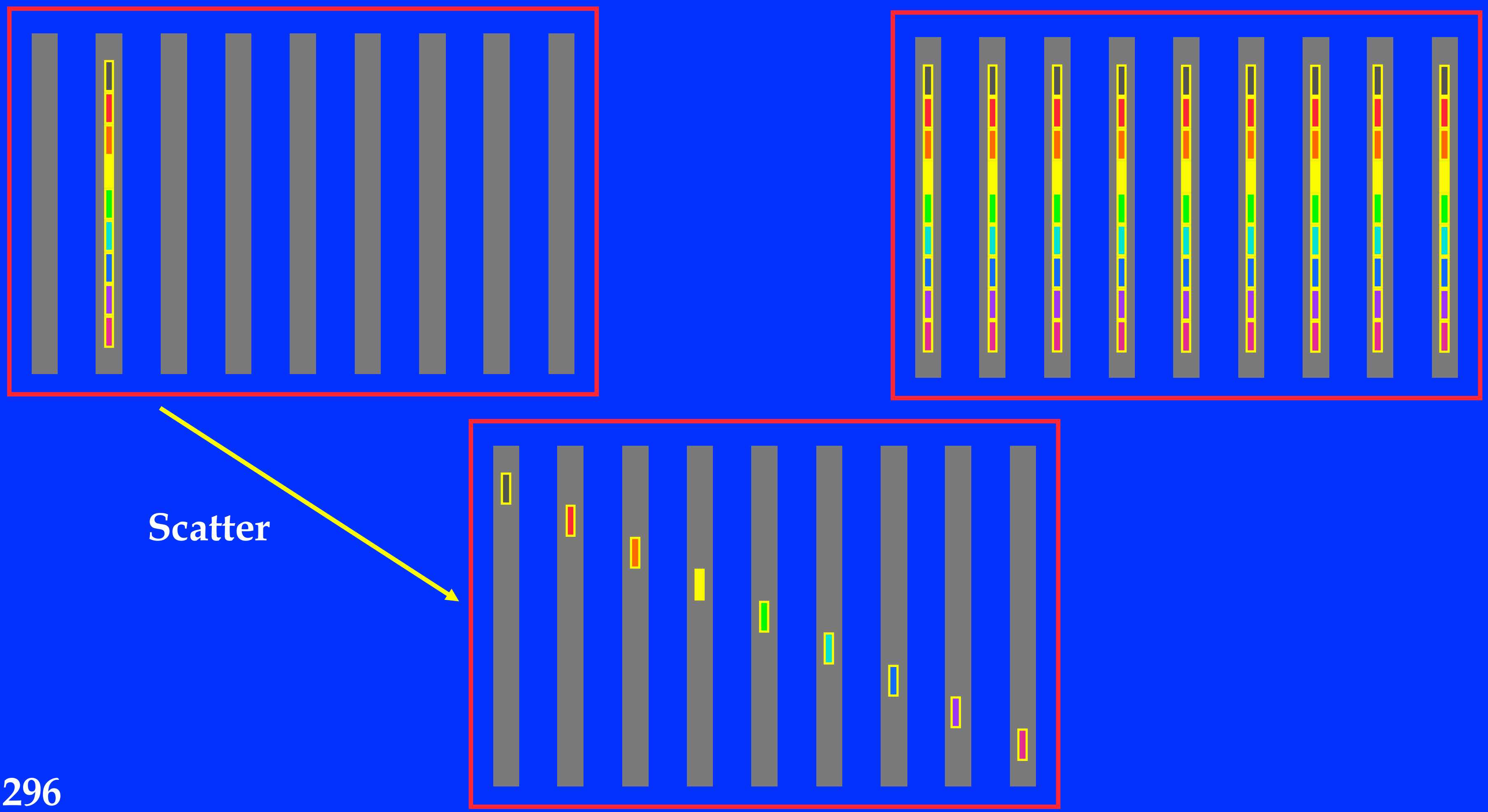


Using the building blocks

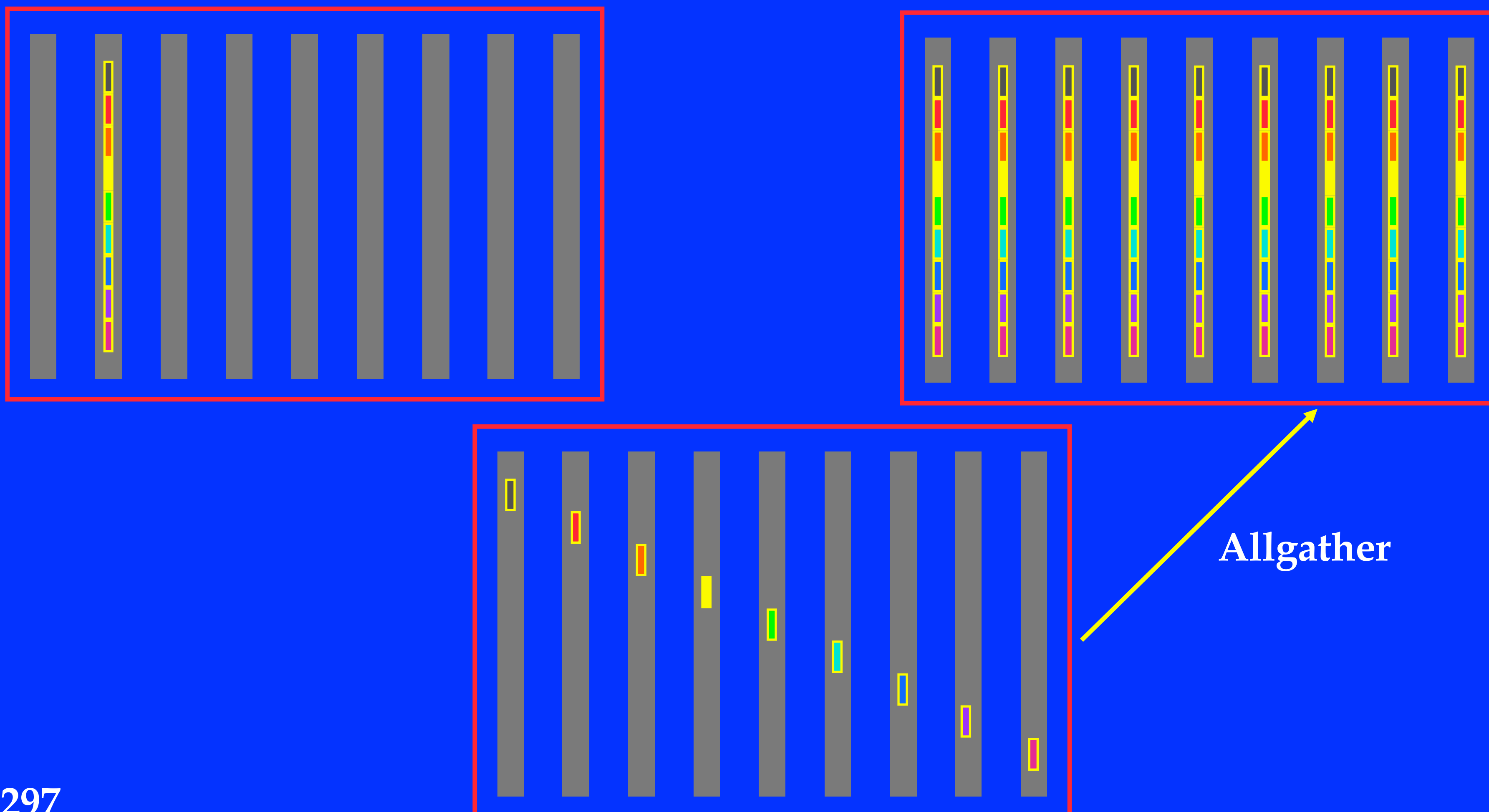
Broadcast (long vector)



Broadcast (long vector)



Broadcast (long vector)



Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

scatter
$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

allgather
$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta$$

Cost of scatter/allgather broadcast

- Assumption: power of two number of nodes

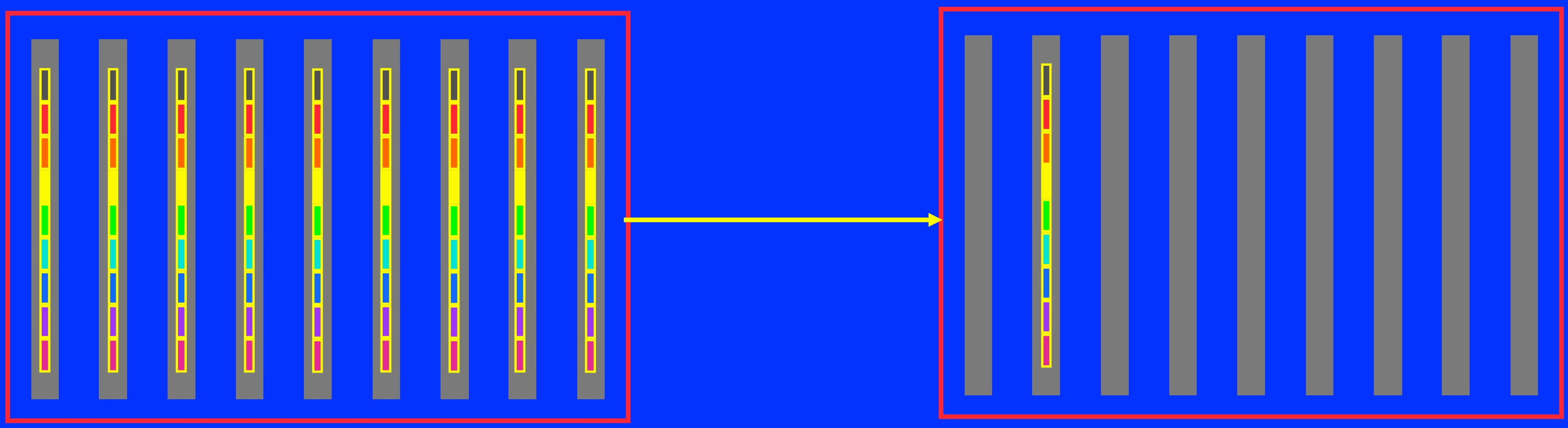
scatter
$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

allgather
$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

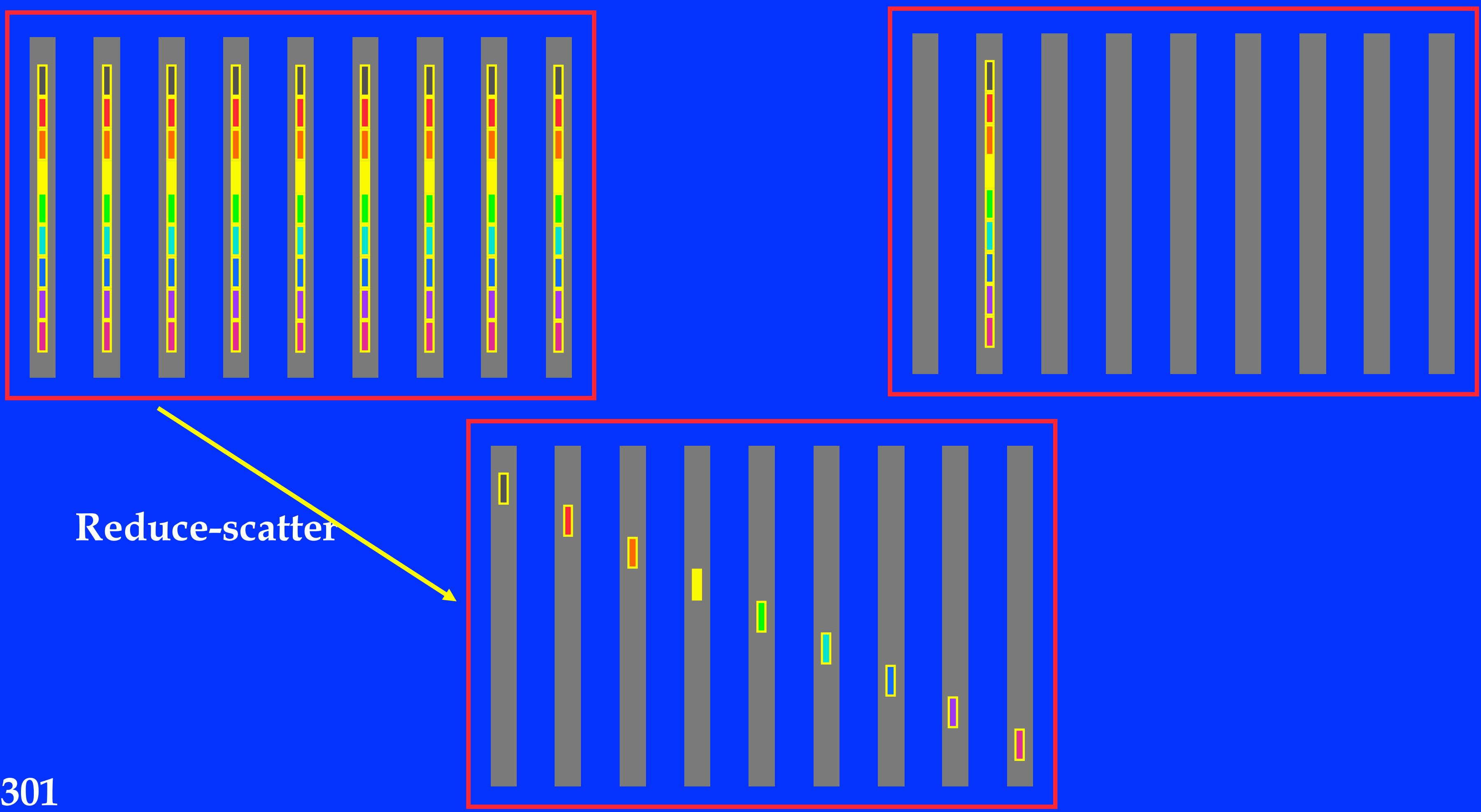
$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta$$

Notice: attains within a factor of two of the lower bound for bandwidth

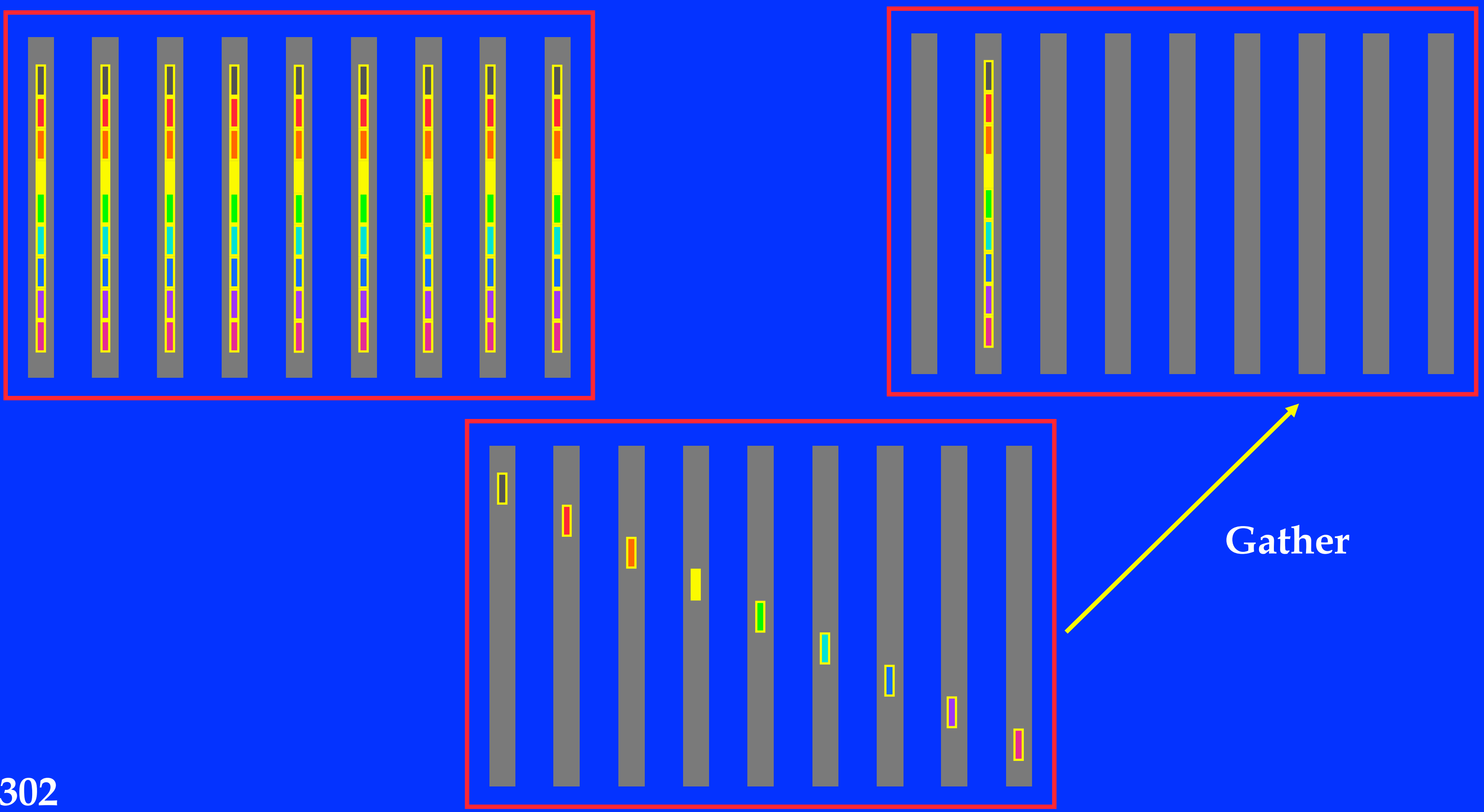
Reduce(-to-one) (long vector)



Combine-to-one (long vector)



Combine-to-one (long vector)



Cost of Reduce-scatter/Gather Reduce(-to-one)

- Assumption: power of two number of nodes

Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

gather $$\log(p)\alpha + \frac{p-1}{p} n\beta$$

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

Cost of Reduce-scatter/Gather Reduce(-to-one)

- Assumption: power of two number of nodes

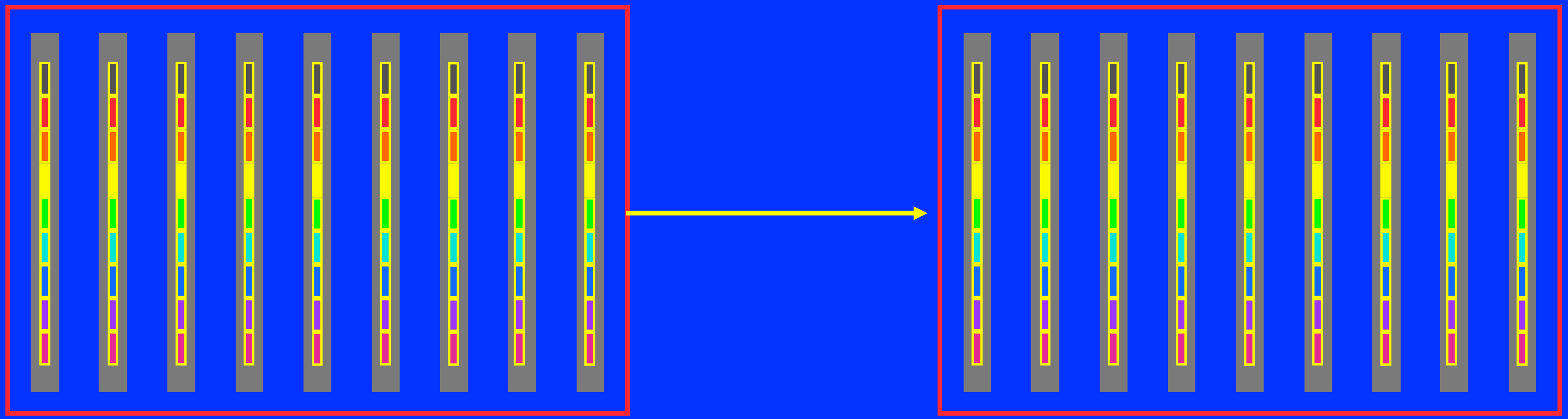
Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

gather $$\log(p)\alpha + \frac{p-1}{p}n\beta$$

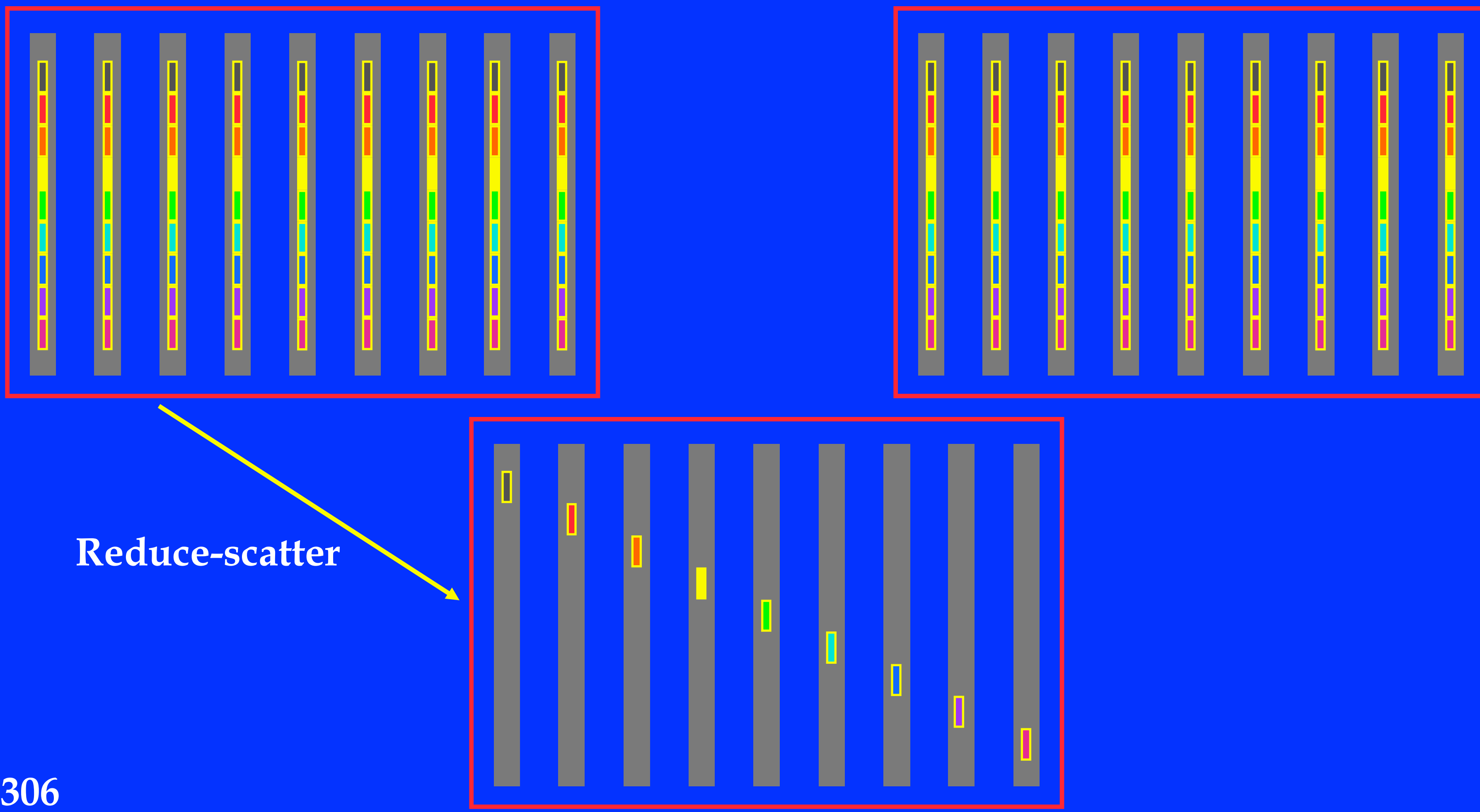
$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Notice: attains within a factor of two of the lower bound for bandwidth and attains lower bound for computation

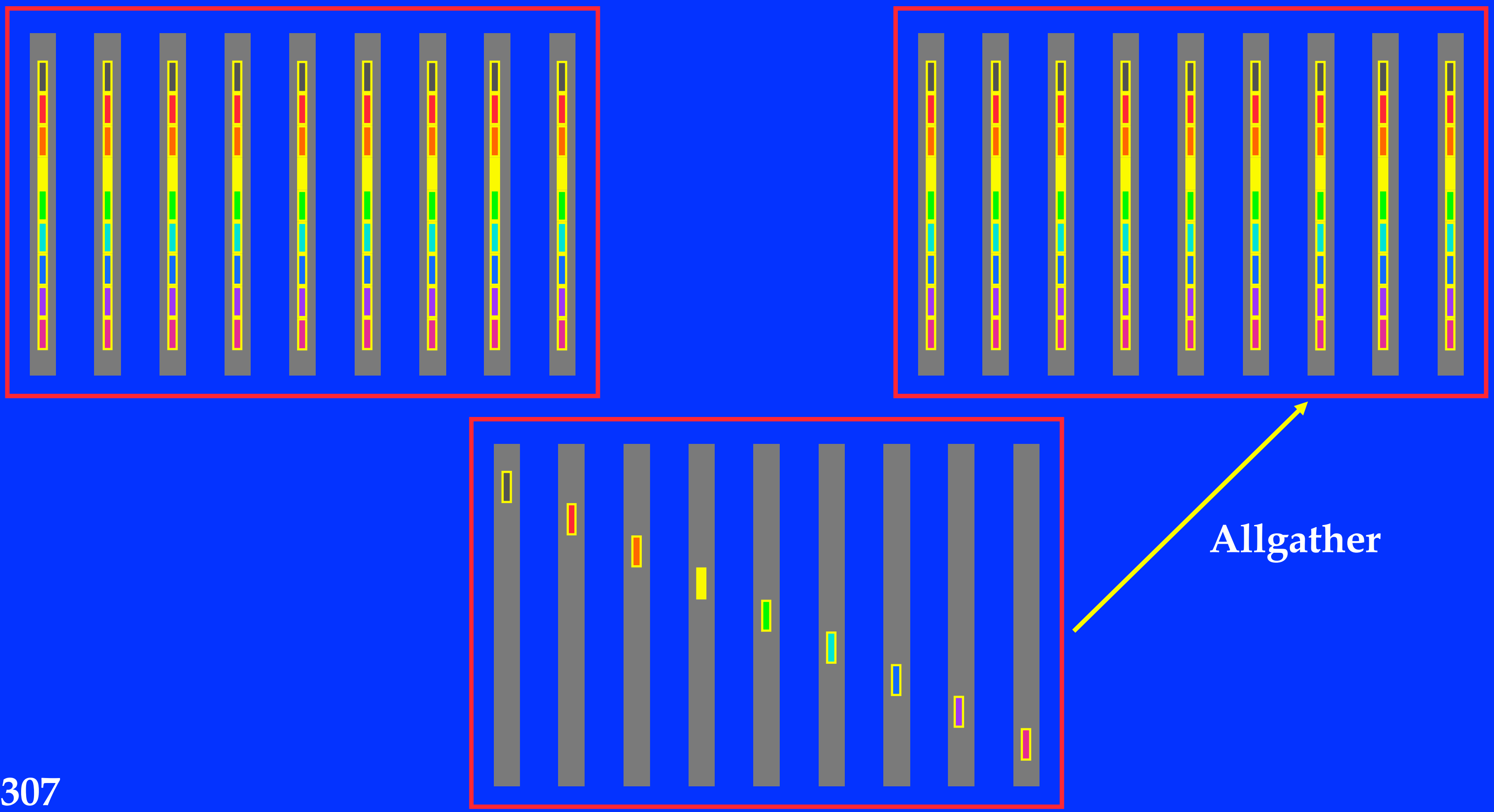
Allreduce
(long vector)



Allreduce (long vector)



Allreduce (long vector)



Cost of Reduce-scatter/Allgather Allreduce

- **Assumption: power of two number of nodes**

Reduce-scatter $$(p-1)\alpha + \frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Allgather $$(p-1)\alpha + \frac{p-1}{p}n\beta$$

$$2(p-1)\alpha + 2\frac{p-1}{p}n\beta + \frac{p-1}{p}n\gamma$$

Cost of Reduce-scatter/Allgather Allreduce

- Assumption: power of two number of nodes

Reduce-scatter $(p-1)\alpha + \frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$

Allgather $(p-1)\alpha + \frac{p-1}{p} n\beta$

$$2(p-1)\alpha + 2\frac{p-1}{p} n\beta + \frac{p-1}{p} n\gamma$$

Notice: attains the lower bound for bandwidth and computation

Recap

Reduce-scatter

$$(p-1)\alpha+\frac{p-1}{p}n(\beta+\gamma)$$

Scatter

$$\log(p)\alpha+\frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha+\frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha+\frac{p-1}{p}n\beta$$

Reduce(-to-one)

Allreduce

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

Reduce(-to-one)

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

Allreduce

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta + \gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

Reduce(-to-one)

$$(p - 1 + \log(p))\alpha + \frac{p-1}{p} n(2\beta + \gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p} n(2\beta + \gamma)$$

Broadcast

$$(\log(p) + p - 1)\alpha + 2\frac{p-1}{p} n\beta$$

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p} n(\beta+\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p} n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p} n\beta$$

Reduce(-to-one)

$$(p-1+\log(p))\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p} n(2\beta+\gamma)$$

Broadcast

Recap

Reduce-scatter

$$(p-1)\alpha + \frac{p-1}{p}n(\beta+\gamma)$$

Scatter

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Gather

$$\log(p)\alpha + \frac{p-1}{p}n\beta$$

Allgather

$$(p-1)\alpha + \frac{p-1}{p}n\beta$$

Reduce(-to-one)

$$(p-1+\log(p))\alpha + \frac{p-1}{p}n(2\beta+\gamma)$$

Allreduce

$$2(p-1)\alpha + \frac{p-1}{p}n(2\beta+\gamma)$$

Broadcast

$$(\log(p)+p-1)\alpha + 2\frac{p-1}{p}n\beta$$

Advanced Techniques:

Taking advantage of higher dimensions

Physical 2D meshes

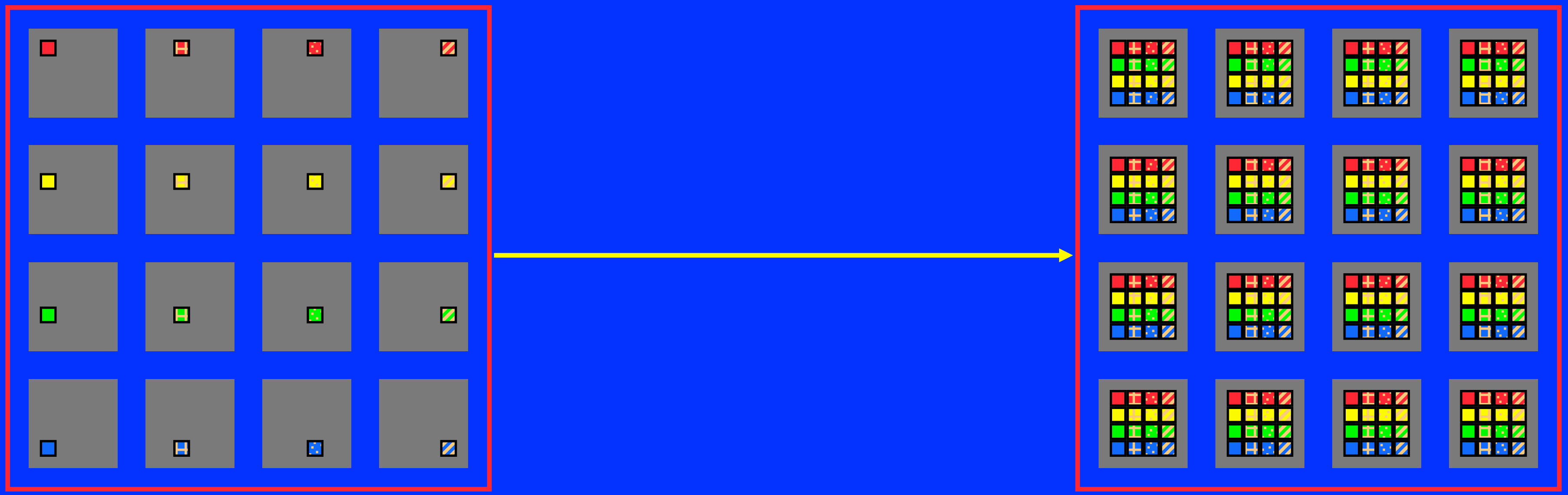
- Simple solution: embed logical linear array
  - problem: large p implies high latency for bucket algorithms

- Advanced solution: perform operation in each dimension
  - collect:

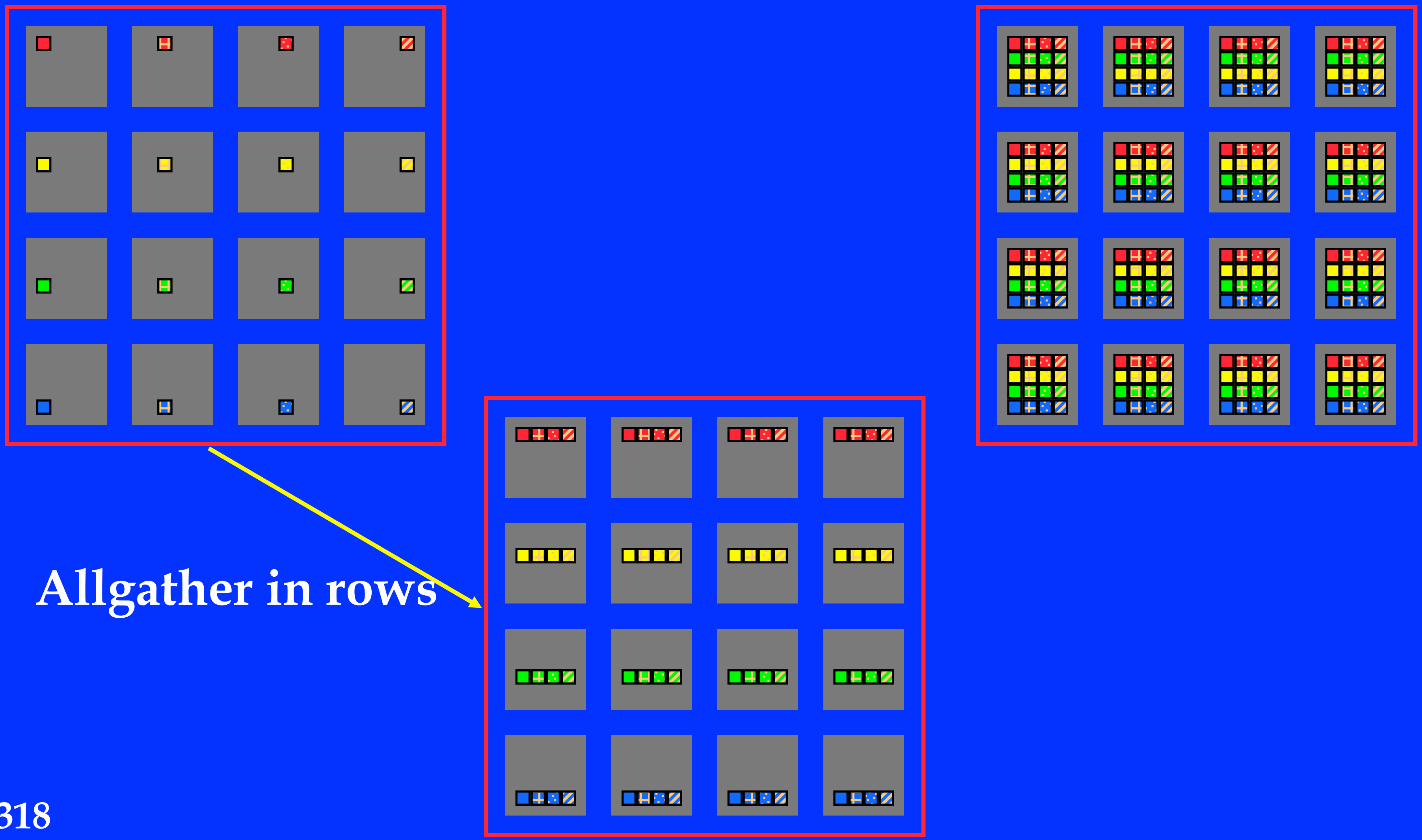
    collect within rows, followed by collect within columns
  - distributed combine:

    same, in reverse

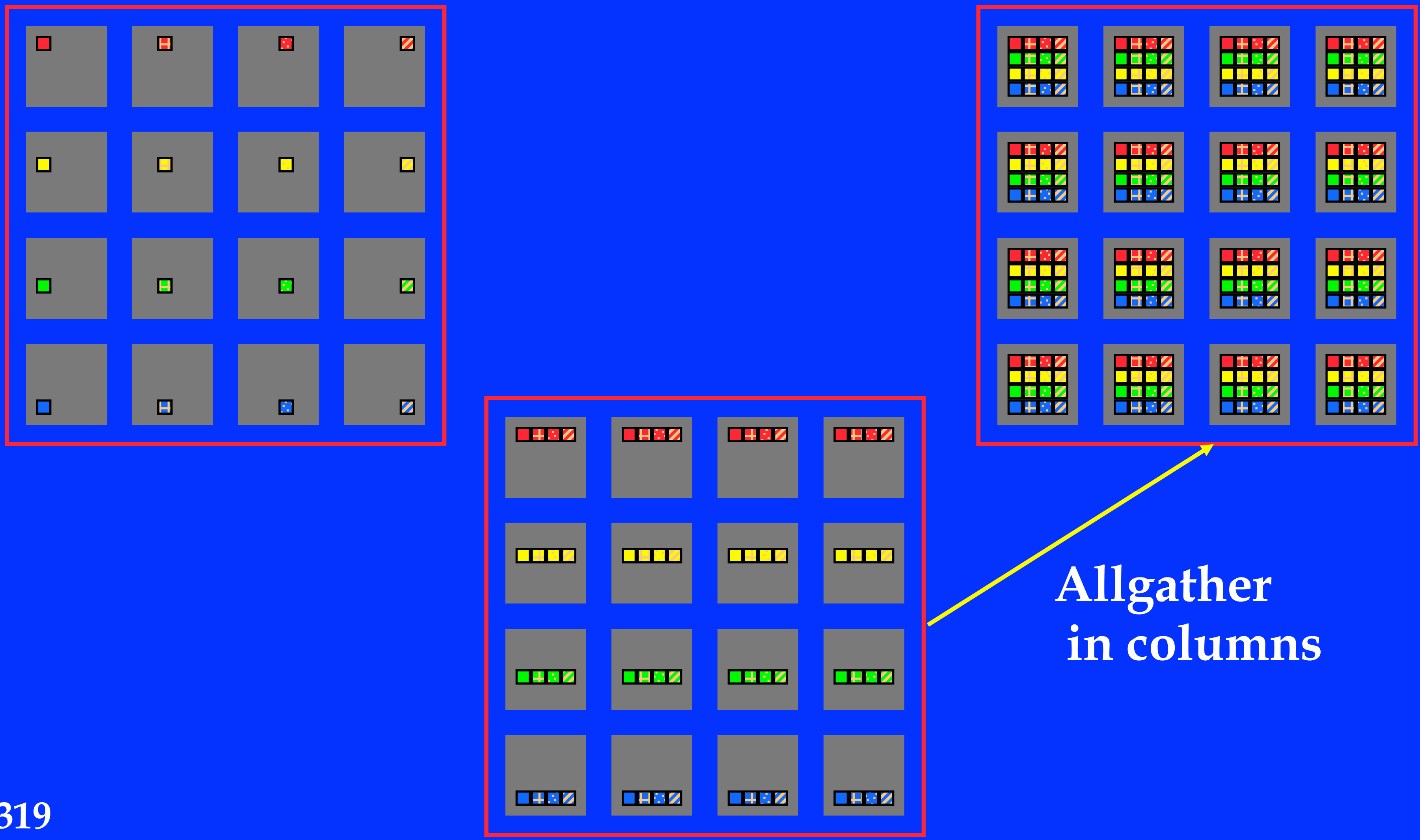
Example: 2D Allgather



Example: 2D Allgather



Example: 2D Collect



Cost of 2D Allgather

row Allgather $$(c-1)\alpha + (c-1)\frac{n}{p}\beta$$

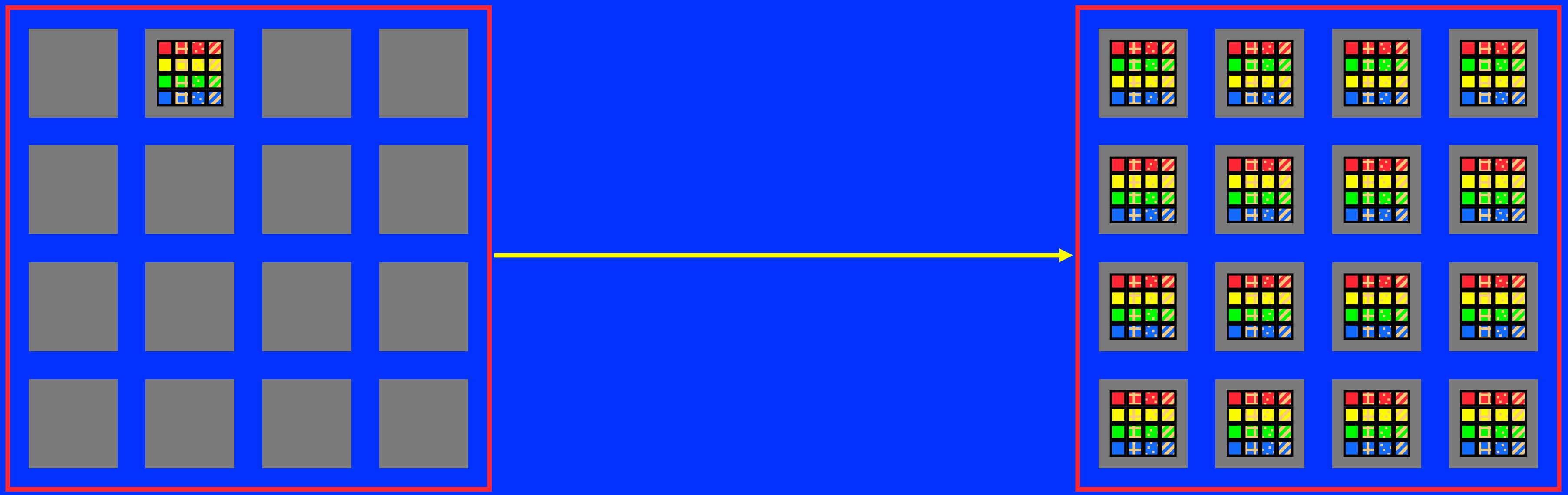
column Allgather $$(r-1)\alpha + (r-1)\frac{c}{p}n\beta$$

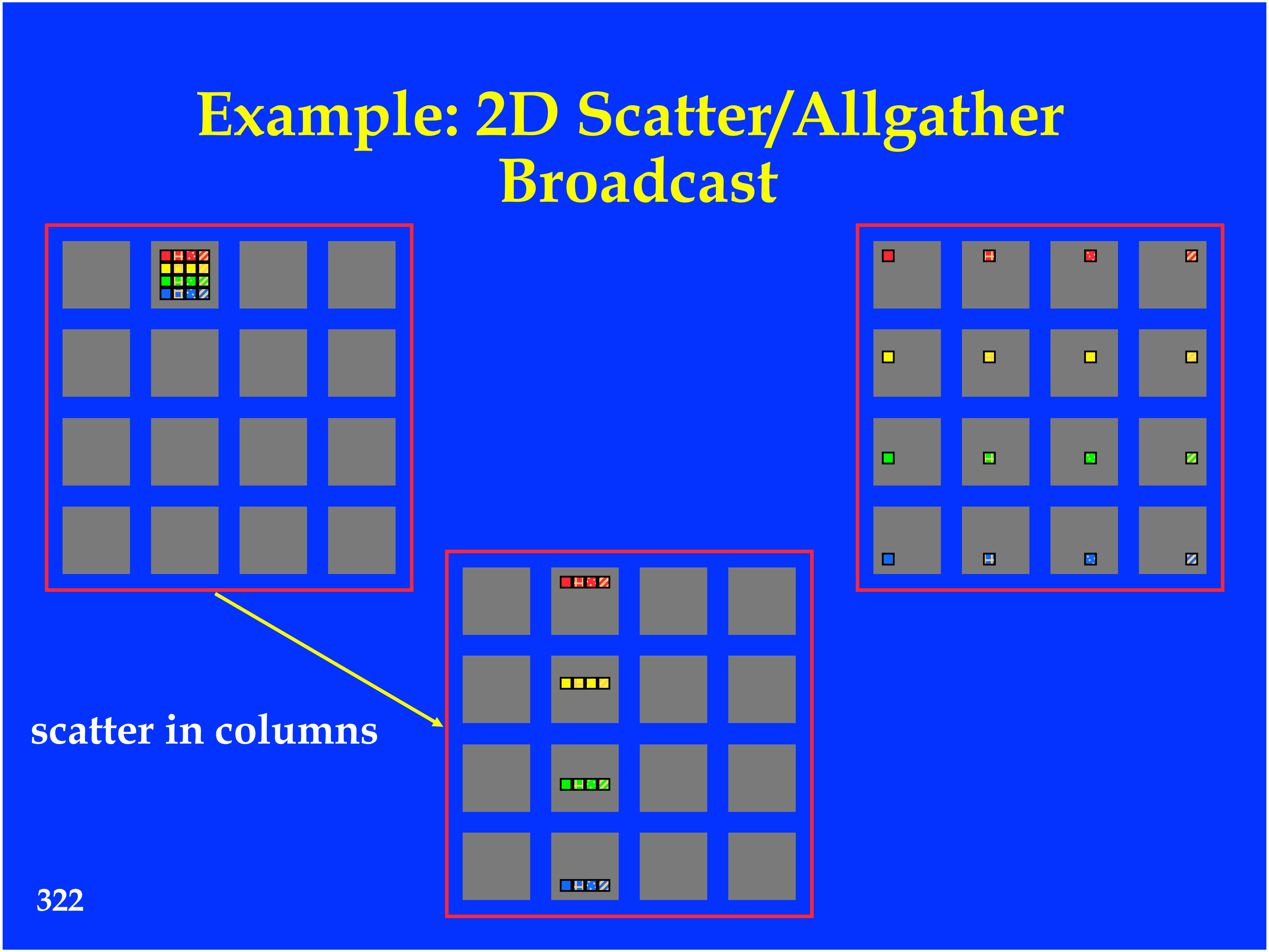
$$(r+c-2)\alpha + \frac{p-1}{p}n\beta$$

latency term is reduced

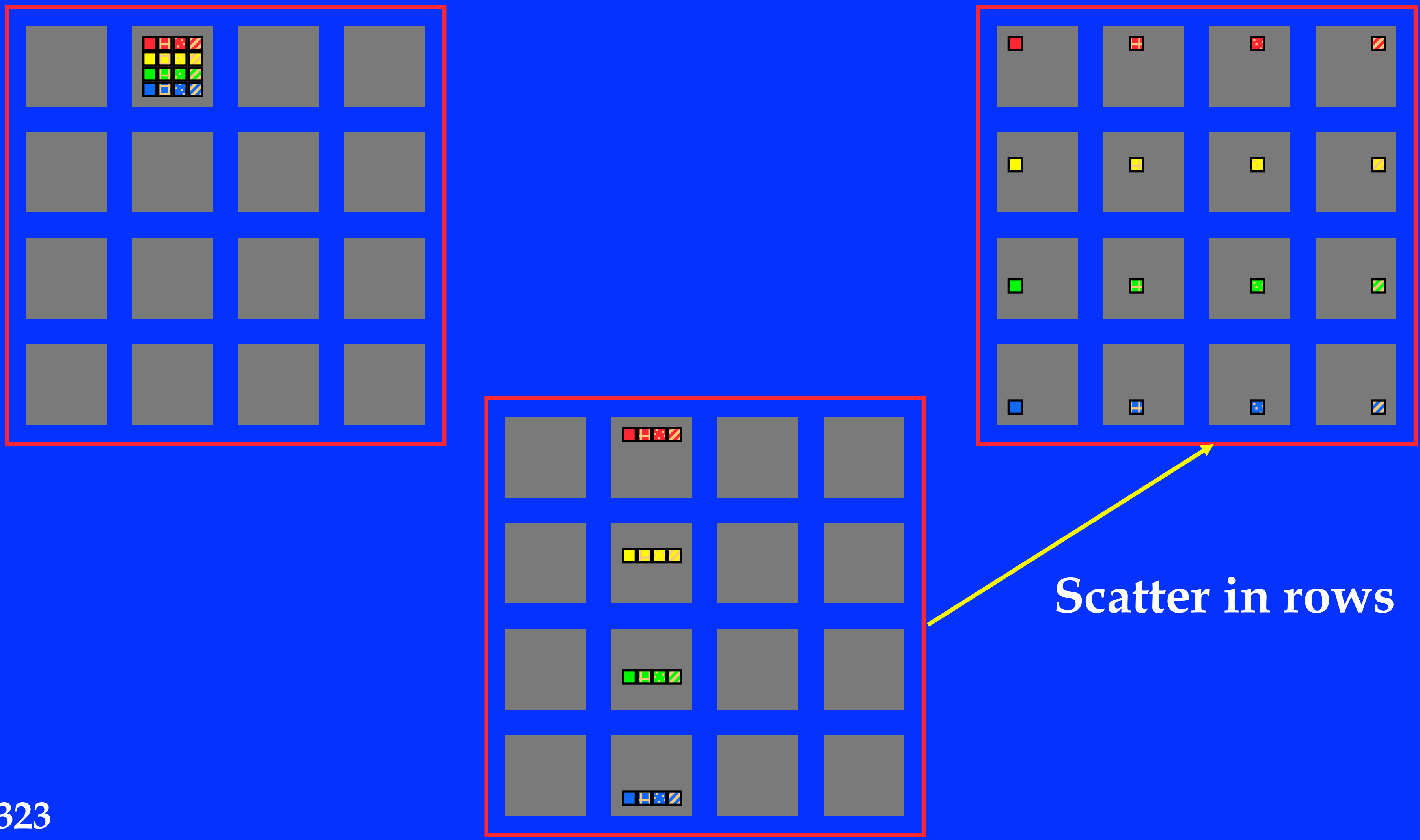
bandwidth term is unaffected

Example: 2D Scatter/Allgather Broadcast

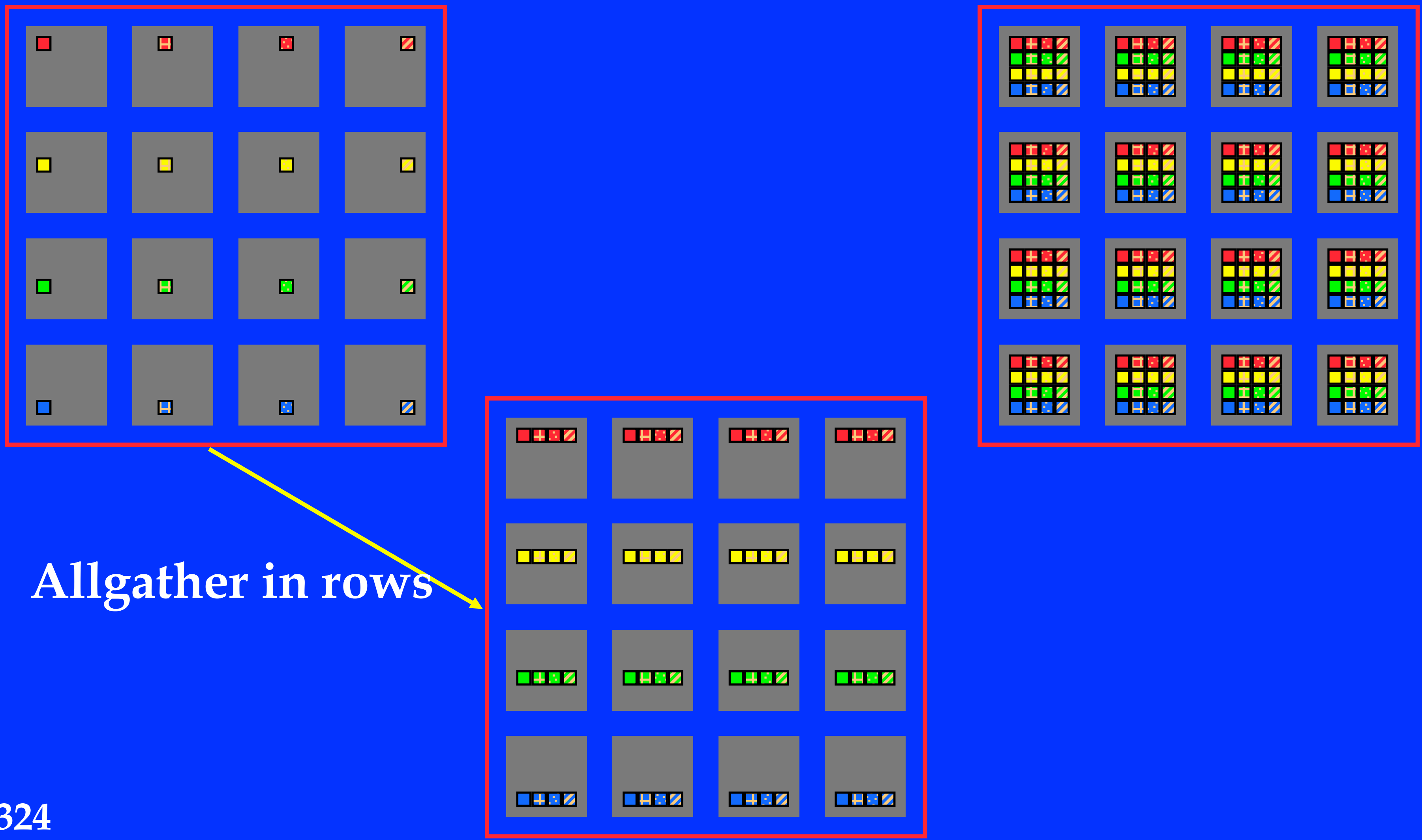



Example: 2D Scatter/Allgather
Broadcast
scatter in columns
322

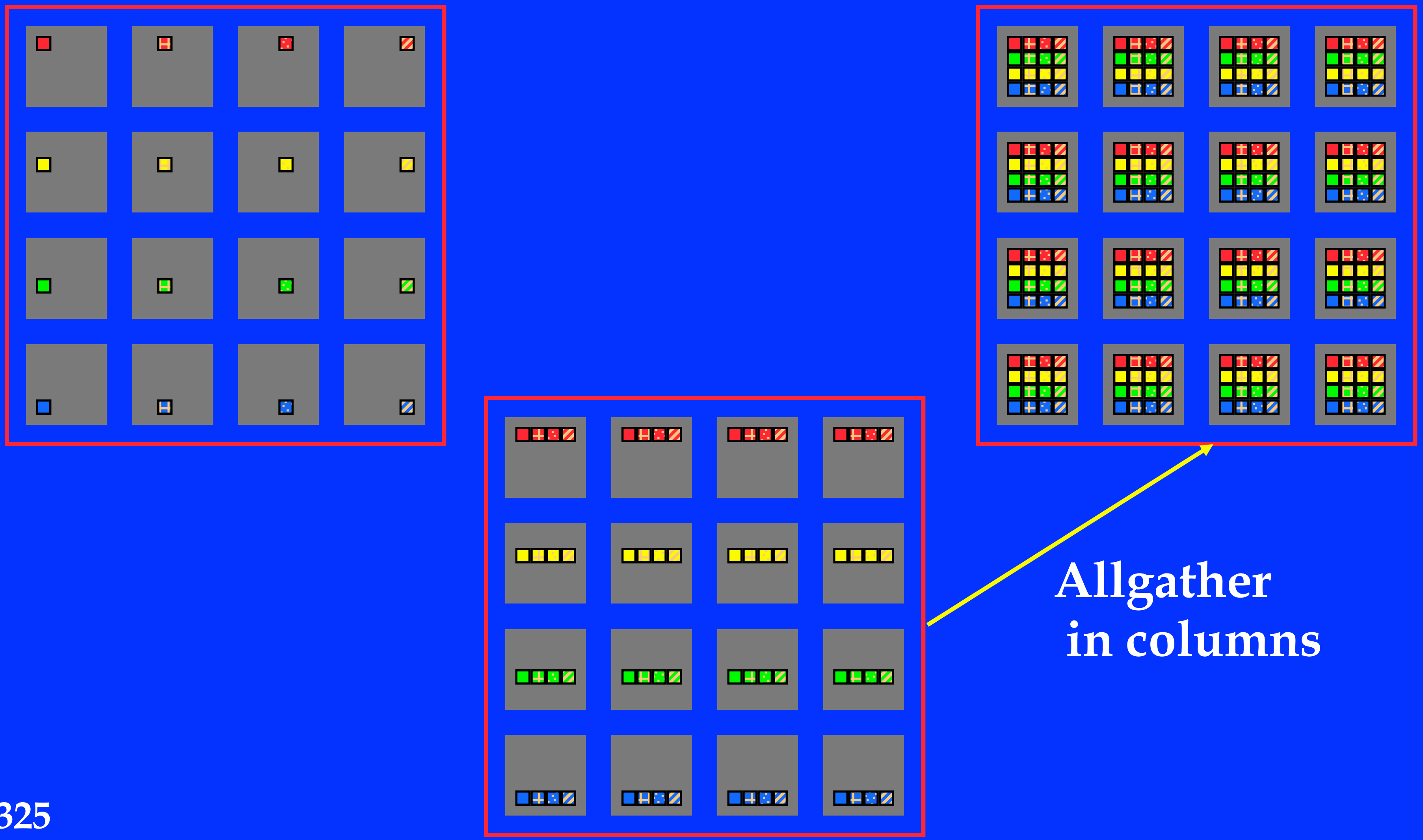
Example: 2D Scatter/ Allgather Broadcast



Example: 2D Scatter/ Allgather Broadcast



Example: 2D Scatter/Collect Broadcast



Cost of 2D scatter/Allgather broadcast

$$(\log(p) + r + c - 2)\alpha + 2\frac{p-1}{p}n\beta$$

A building block approach to library implementation

- Short vector case
- Long vector case
- **Hybrid algorithms**

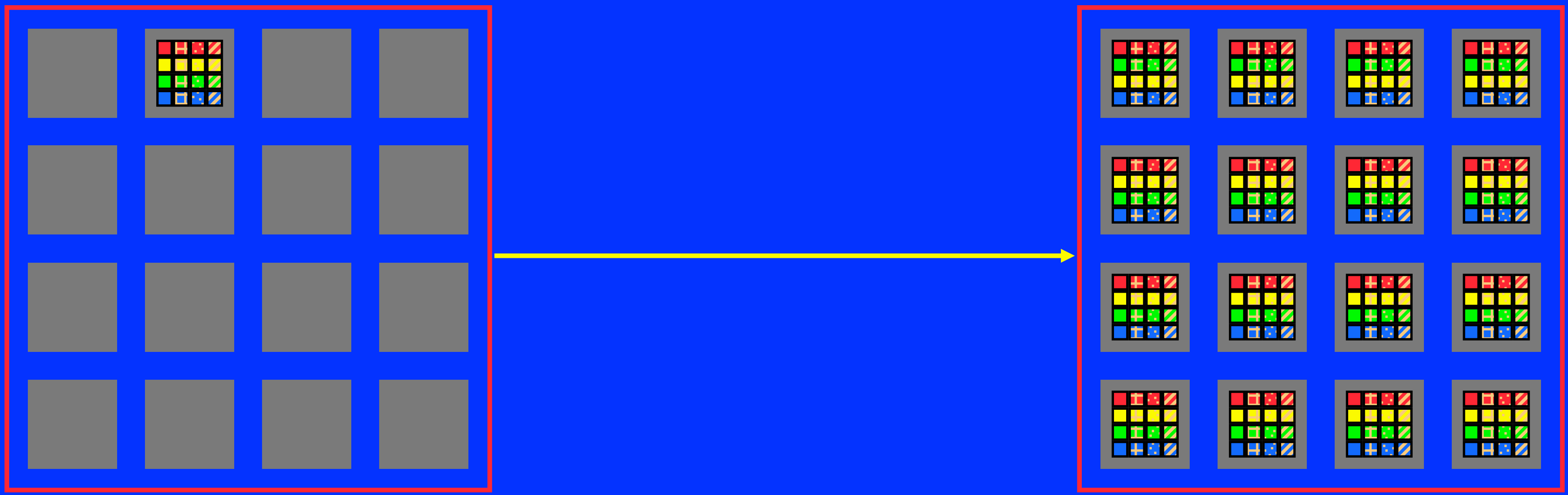
Hybrid algorithms (intermediate length case)

- algorithms must balance latency, cost due to vector length, and network conflicts

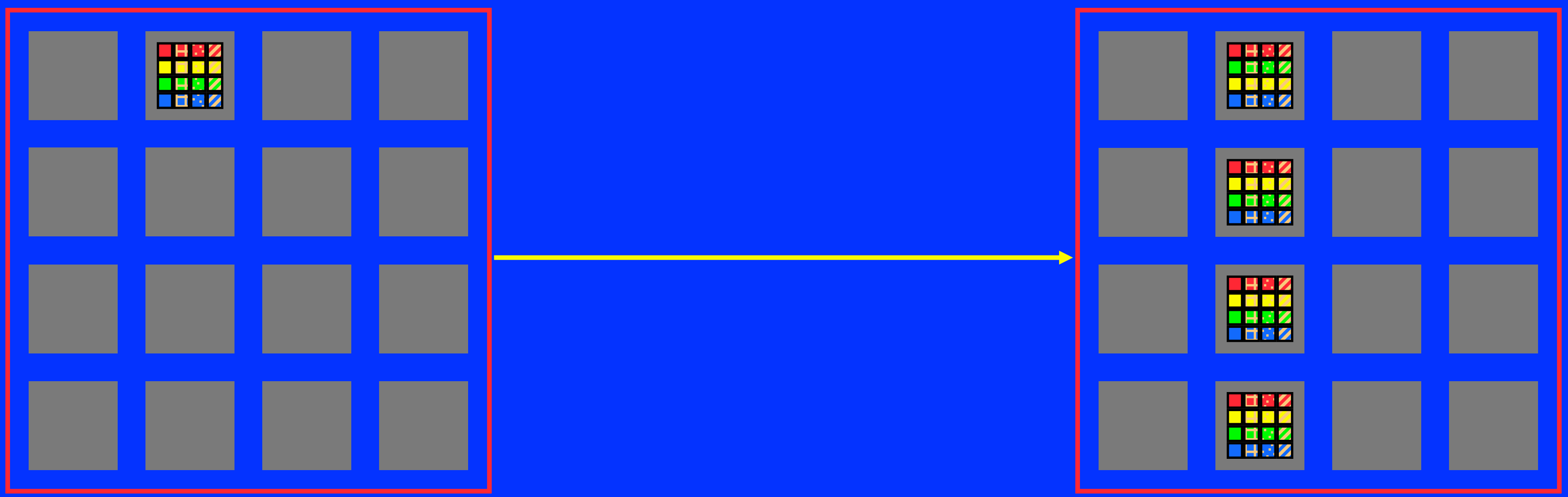
Example

- We will illustrate the techniques using the broadcast as an example
  - short vector: minimum spanning tree broadcast

Example: 2D Broadcast

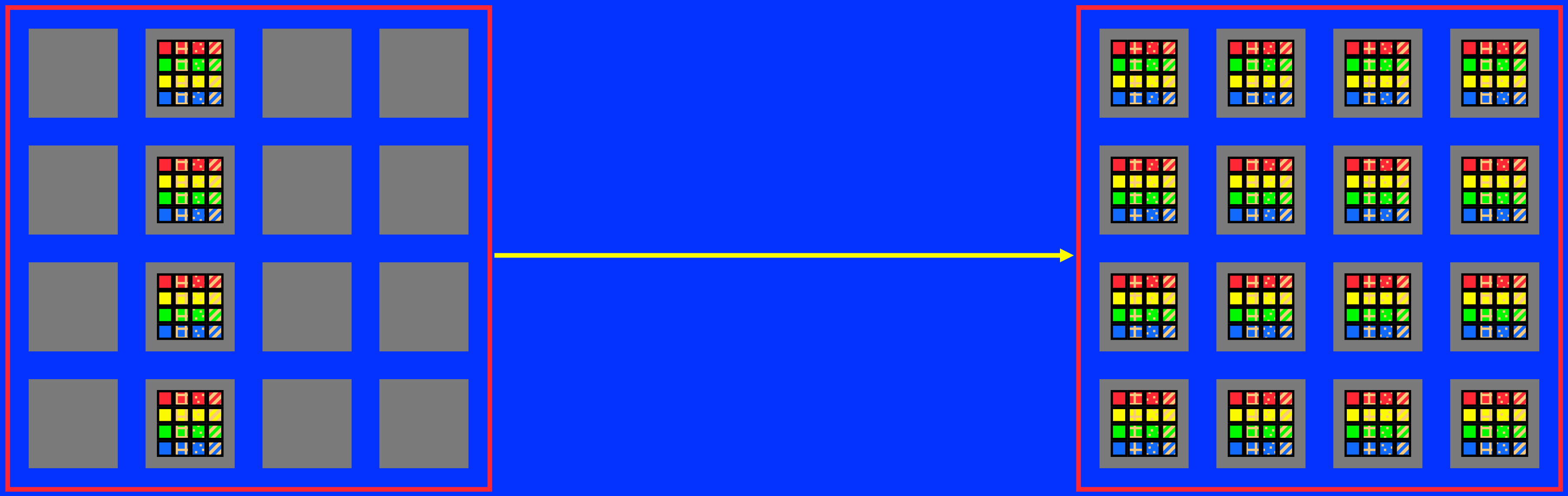


Example: 2D Broadcast



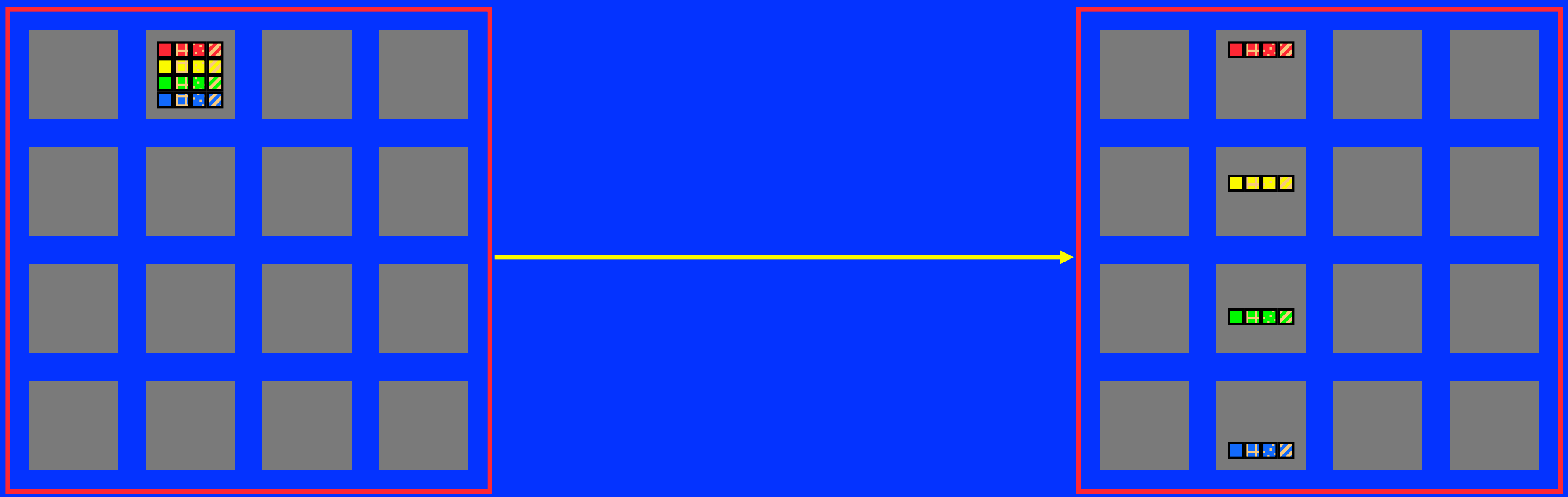
- **Option 1:**
  - **MST broadcast in column**
  - **MST broadcast in rows**

Example: 2D Broadcast



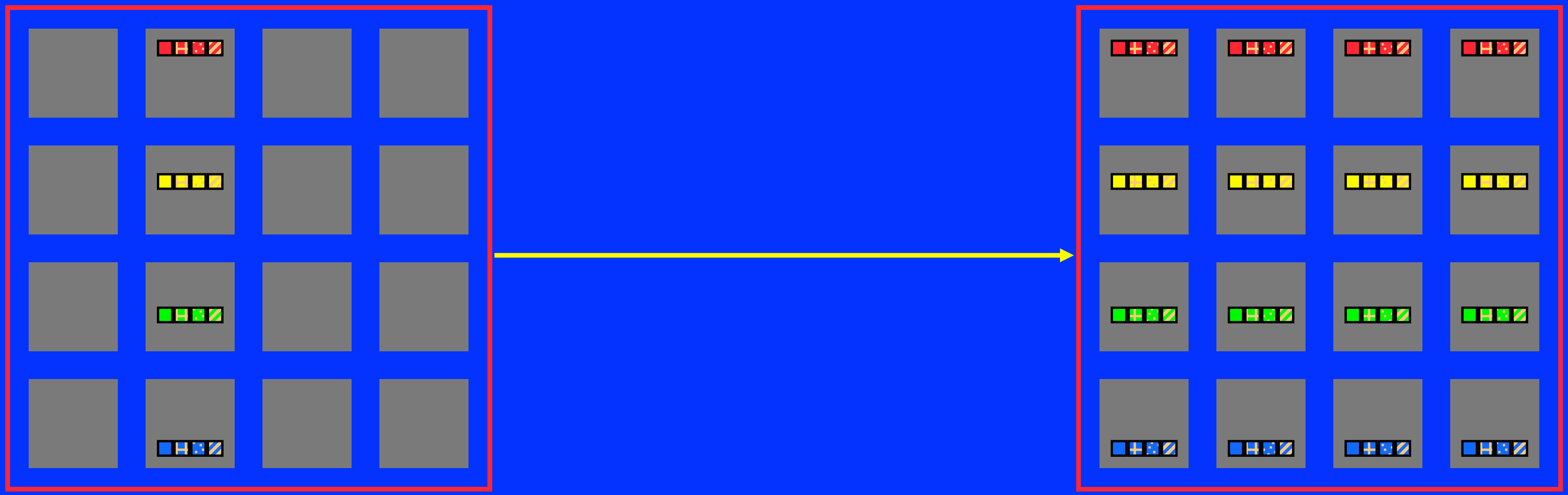
- Option 1:
  - MST broadcast in column
  - MST broadcast in rows

Example: 2D Broadcast



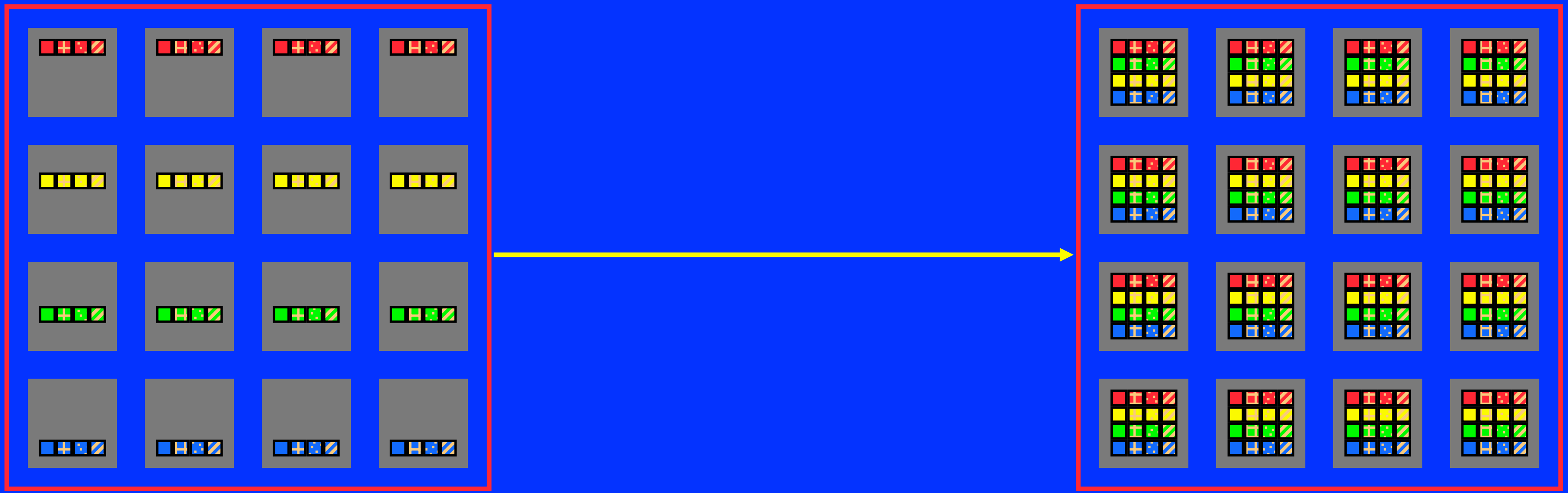
- Option 2:
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns

Example: 2D Broadcast



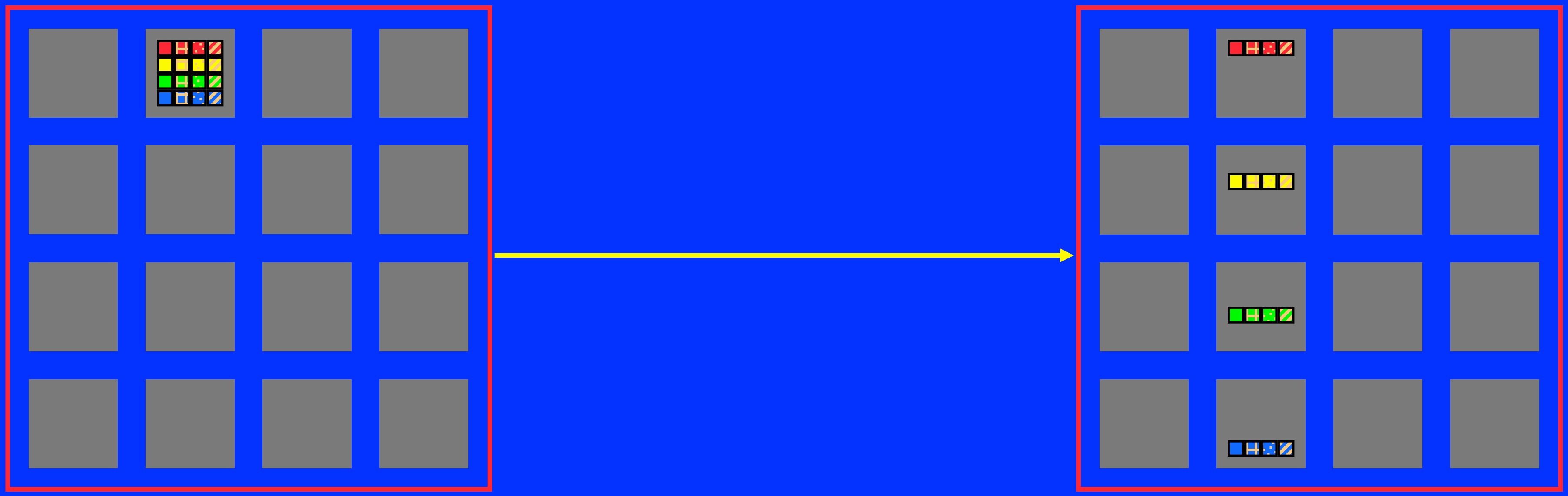
- Option 2:
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns

Example: 2D Broadcast



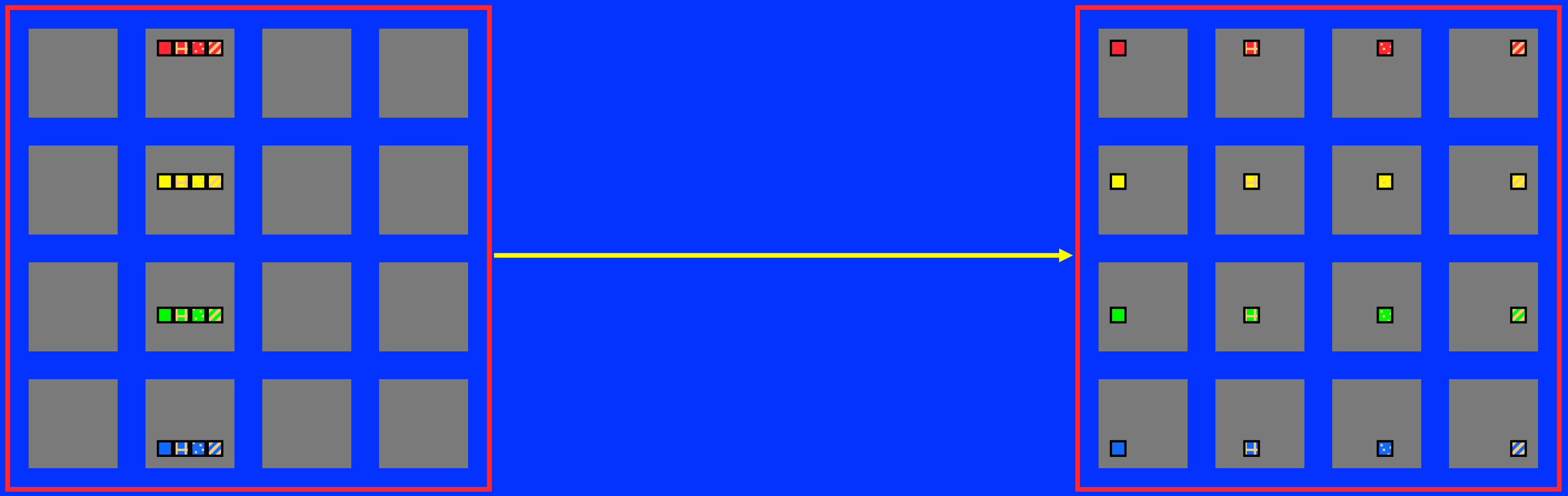
- Option 2:
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns

Example: 2D Broadcast



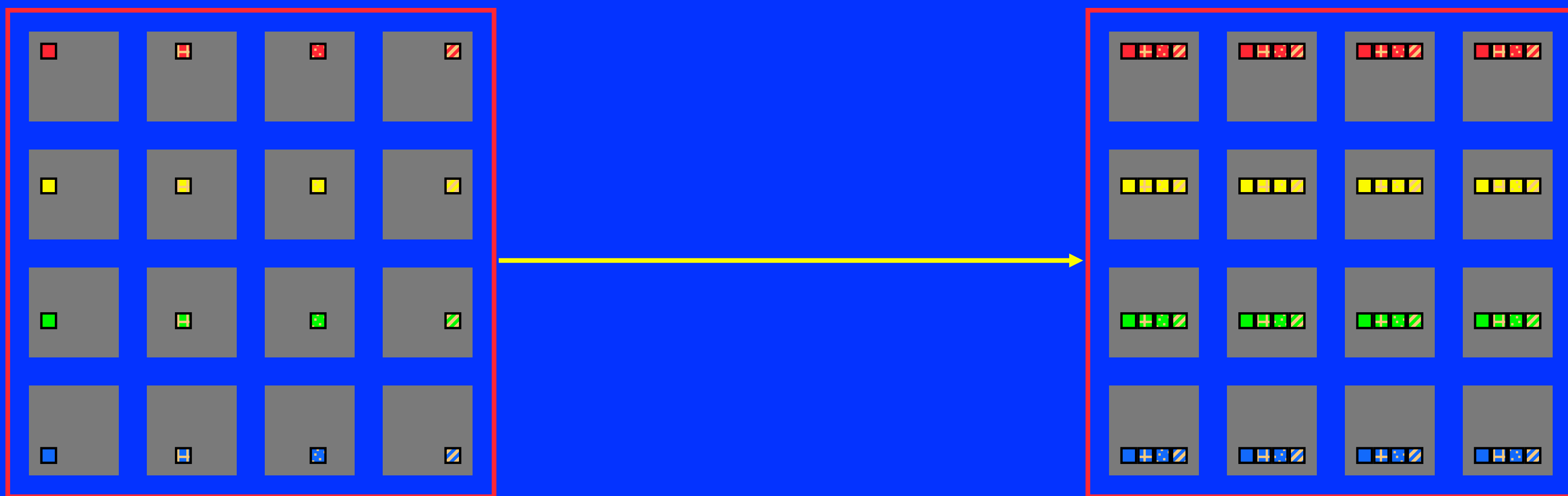
- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

Example: 2D Broadcast



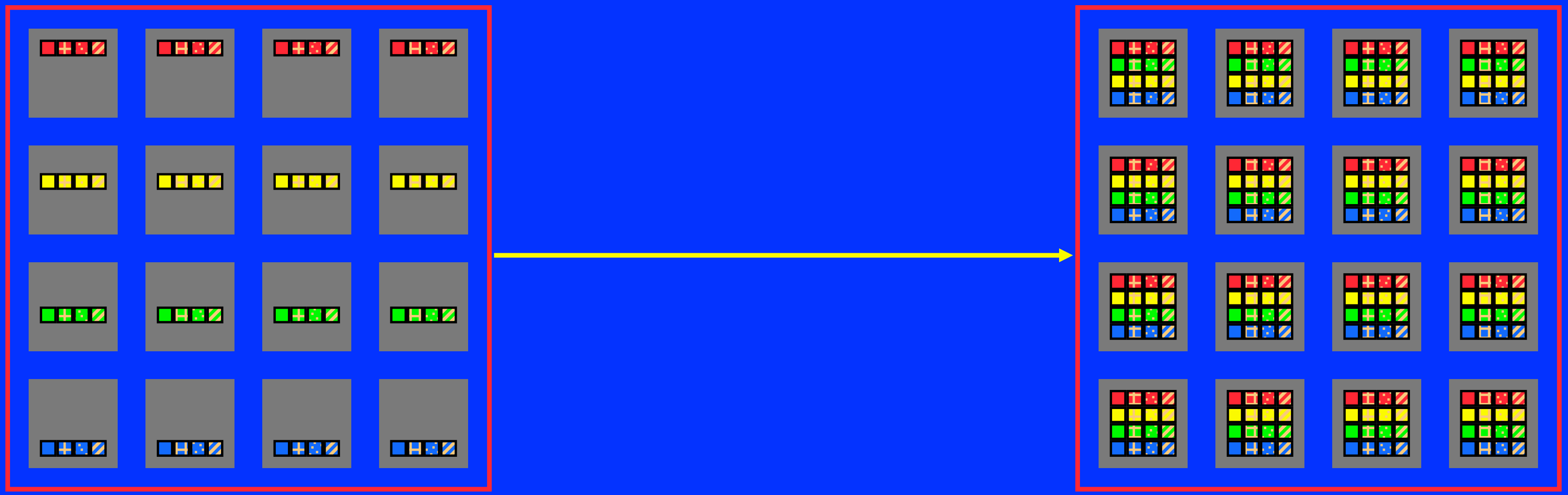
- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

Example: 2D Broadcast



- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

Example: 2D Broadcast



- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

Cost comparison

- Option 1:
  - MST broadcast in column
  - MST broadcast in rows
- Option 2:
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns
- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

$$log(c)\alpha + log(c)n\beta$$

$$log(r)\alpha + log(r)n\beta$$

$$\overline{log(p)\alpha + log(p)n\beta}$$

Cost comparison

- **Option 1:**
  - MST broadcast in column
  - MST broadcast in rows
- **Option 2:**
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns
- **Option 3:**
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

$$log(c)\alpha + \frac{c-1}{c} n\beta$$

$$log(r)\alpha + log(r)\frac{n}{c}\beta$$

$$(c-1)\alpha + \frac{c-1}{c} n\beta$$

$$\left(log(p) + c - 1\right)\alpha + \left(2\frac{c-1+log(r)}{c}\right) n\beta$$

Cost comparison

- Option 1:
  - MST broadcast in column
  - MST broadcast in rows
- Option 2:
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns
- Option 3:
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

$$log(c)\alpha + \frac{c-1}{c} n\beta$$

$$log(r)\alpha + \frac{r-1}{r}\frac{n}{c}\beta$$

$$(r-1)\alpha + \frac{r-1}{r}\frac{n}{c}\beta$$

$$(c-1)\alpha + \frac{c-1}{c} n\beta$$

$$\overline{(log(p) + r + c - 2)\alpha + 2\frac{p-1}{p} n\beta}$$

Cost comparison

- **Option 1:**
  - MST broadcast in column
  - MST broadcast in rows

$$log(p)\alpha + log(p)n\beta$$

- **Option 2:**
  - Scatter in column
  - MST broadcast in rows
  - Allgather in columns

$$\left(log(p) + c - 1\right)\alpha + \left(2\frac{c - 1 + log(r)}{c}\right)n\beta$$

- **Option 3:**
  - Scatter in column
  - Scatter in rows
  - Allgather in rows
  - Allgather in columns

$$(log(p) + r + c - 2)\alpha + 2\frac{p-1}{p}n\beta$$

Higher dimensions

- This technique can be extended by viewing one- and two-dimensional meshes logically as higher dimensions
  - reduces latency
  - incurs network conflicts
  - can be used to create faster short vector implementations

- Details require more time that is available today

Other techniques

- Pipelined algorithms
  - can be used to further reduce the cost of broadcast and combine-to-one for long vectors
  - very effective on hypercubes
    - » (Ho and Johnsson)
  - effective on meshes with low latency
    - » (Watts and van de Geijn)
  - complicated to implement, analyze and explain

Outline

Part I: Theory

- **Model of computation**
- **Collective communications**
- **A building block approach to library implementation**

Part II: Practice

- **Implementation on the Paragon**
- **Performance results**
- **Applications**

Outline

Part I: Theory

- Model of computation
- Collective communications
- A building block approach to library implementation

Part II: Practice

- Implementation on the Paragon
- Performance results
- Applications

Theory is nice, but how does it work in practice?

- Paragon does not match our model
  - Bad news:
    - » sending and receiving more complex then the model indicates
    - » forced messages vs. unforced messages
    - » preposted messages vs. nonpreposted messages
    - » etc.
  - Good news:
    - » excess bandwidth in the network

Interprocessor Collective Communication (InterCom) Project

Implementation on the Paragon

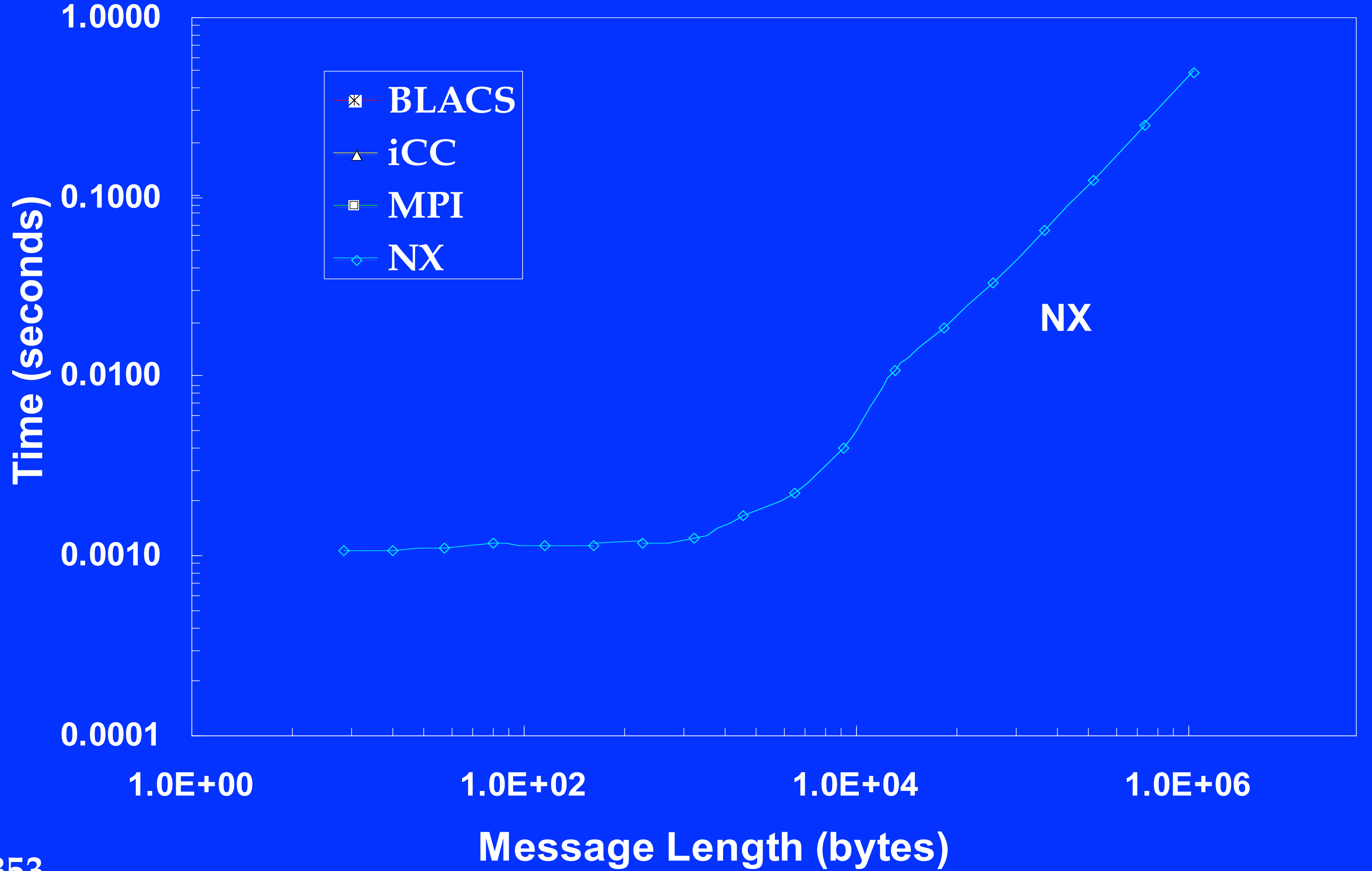
- Short vector building blocks
  - reduce latency by *not* preposting and synchronizing
- Long vector building blocks
  - improve bandwidth by preposting and synchronizing
- Incorporate more complex issues into model
  - various startups, bandwidths, depending on situation
- Use simple heuristic to choose hybrid strategy
  - because of excess bandwidth, the mesh acts more like a hypercube, for which some solid theory exists
    - » (van de Geijn)
  - details go beyond this tutorial.

Performance

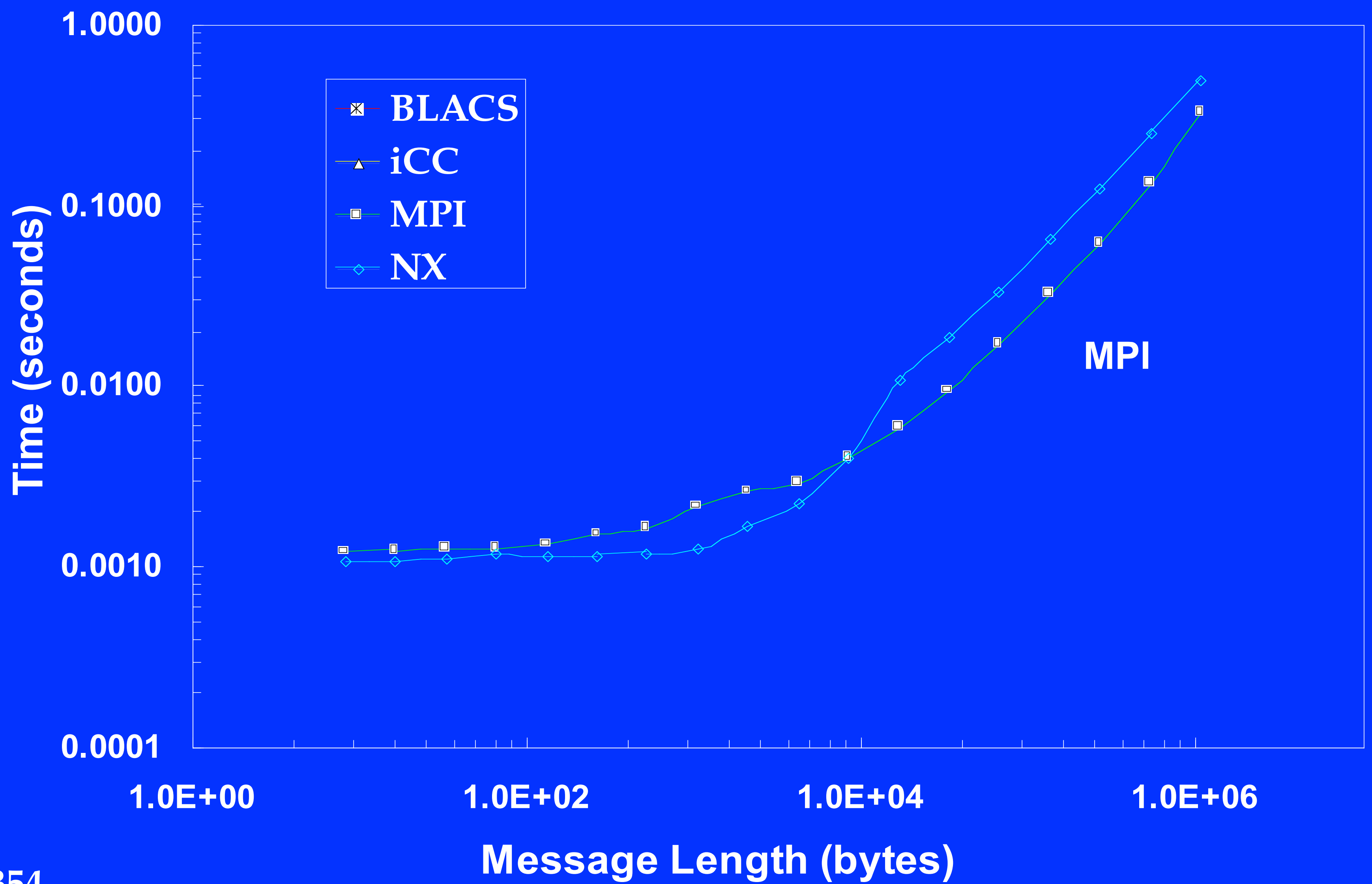
Performance comparison

- NX collective communication
- Message Passing Interface (MPI)
  - Reference implementation from ANL and MSU
  - Bill Gropp, Rusty Lusk, and Tony Skjellum
- Basic Linear Algebra Communication Subprograms (BLACS)
  - Communication library of ScaLAPACK
  - Reference implementation from the Univ. of TN
  - Jack Dongarra and Clint Whaley
- Interprocessor Collective Communication (iCC) Library
  - High performance implementation by the InterCom team

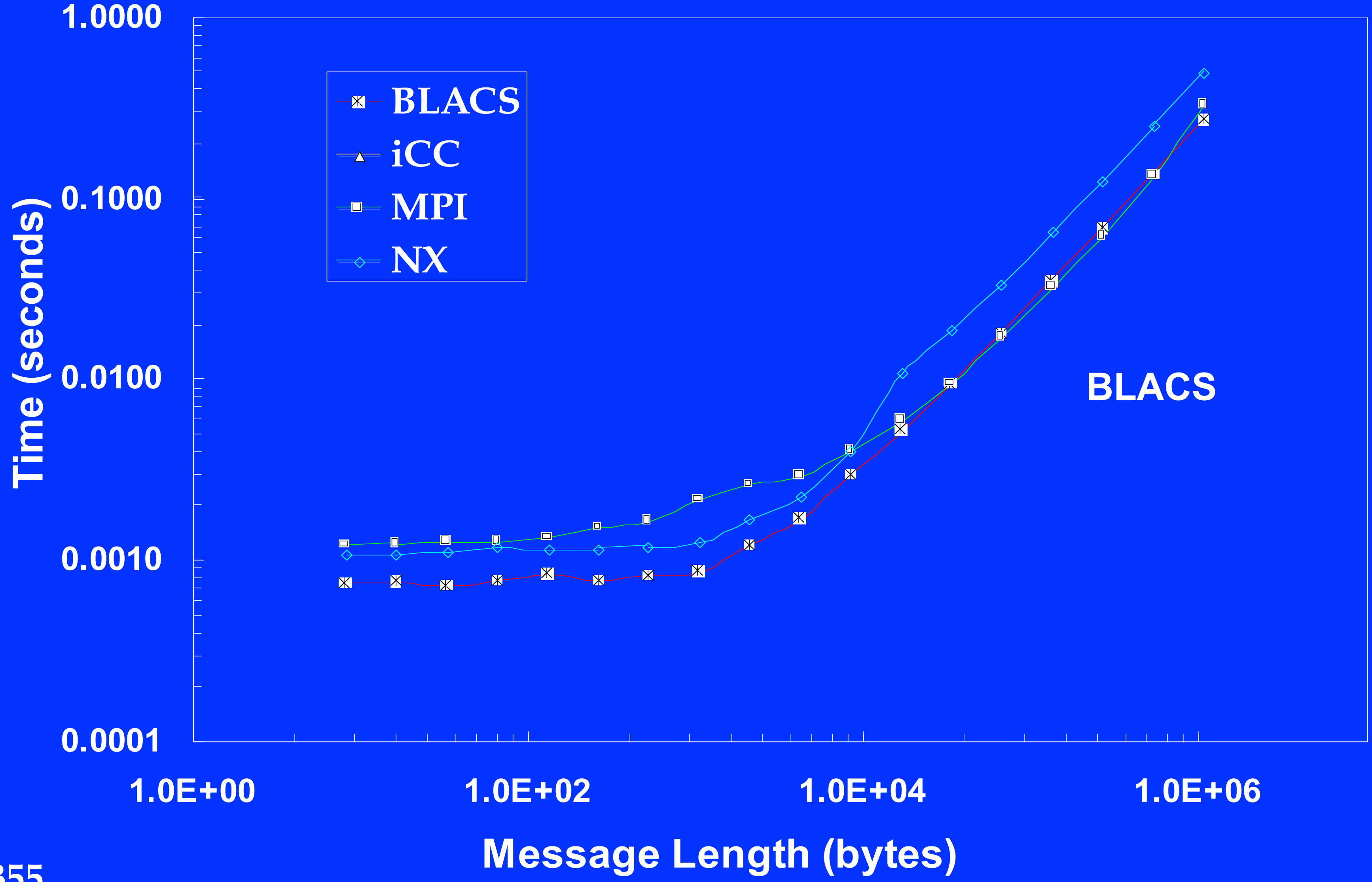
Broadcast on 16 x 32 mesh Paragon



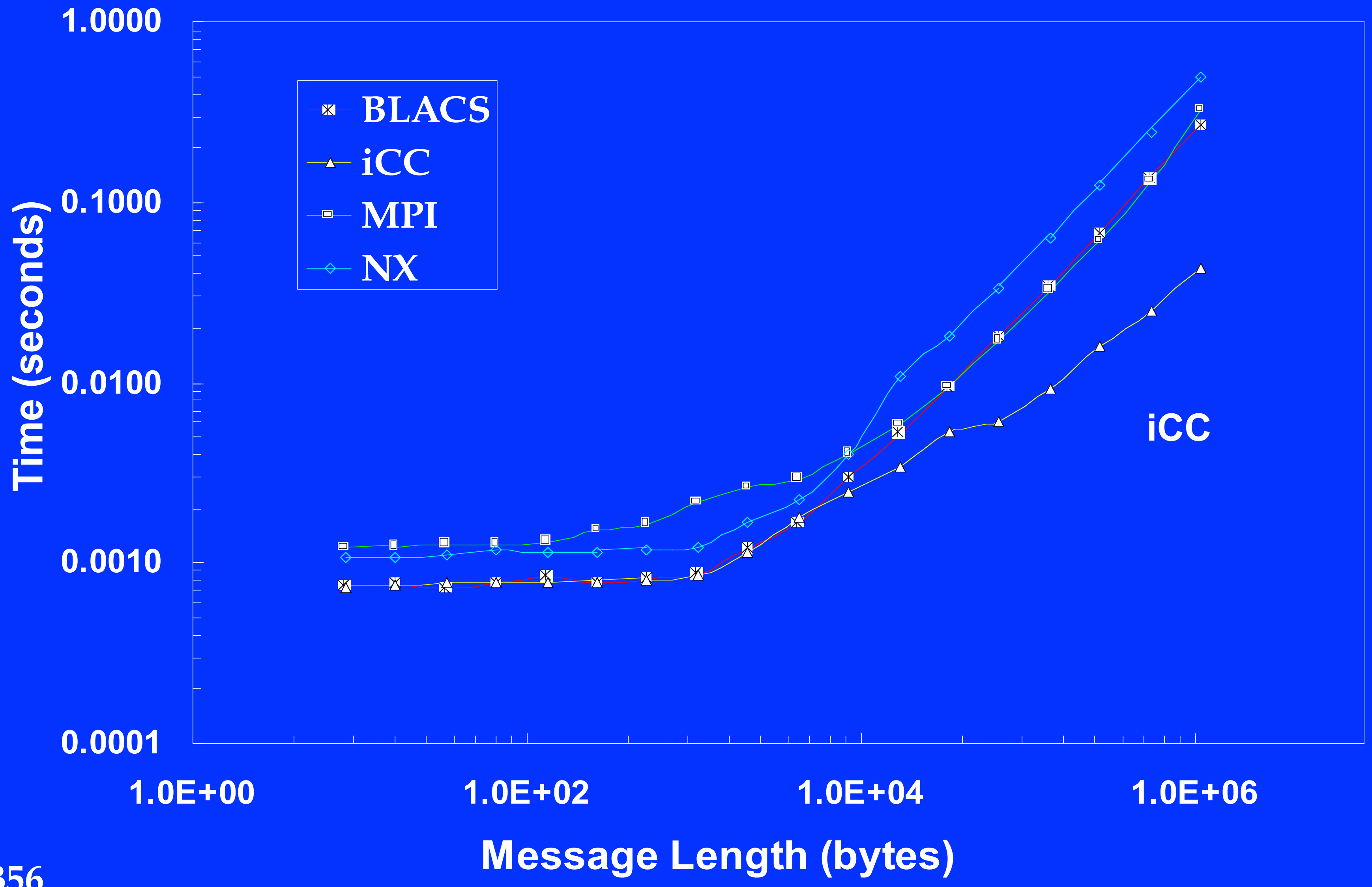
Broadcast on 16 x 32 mesh Paragon



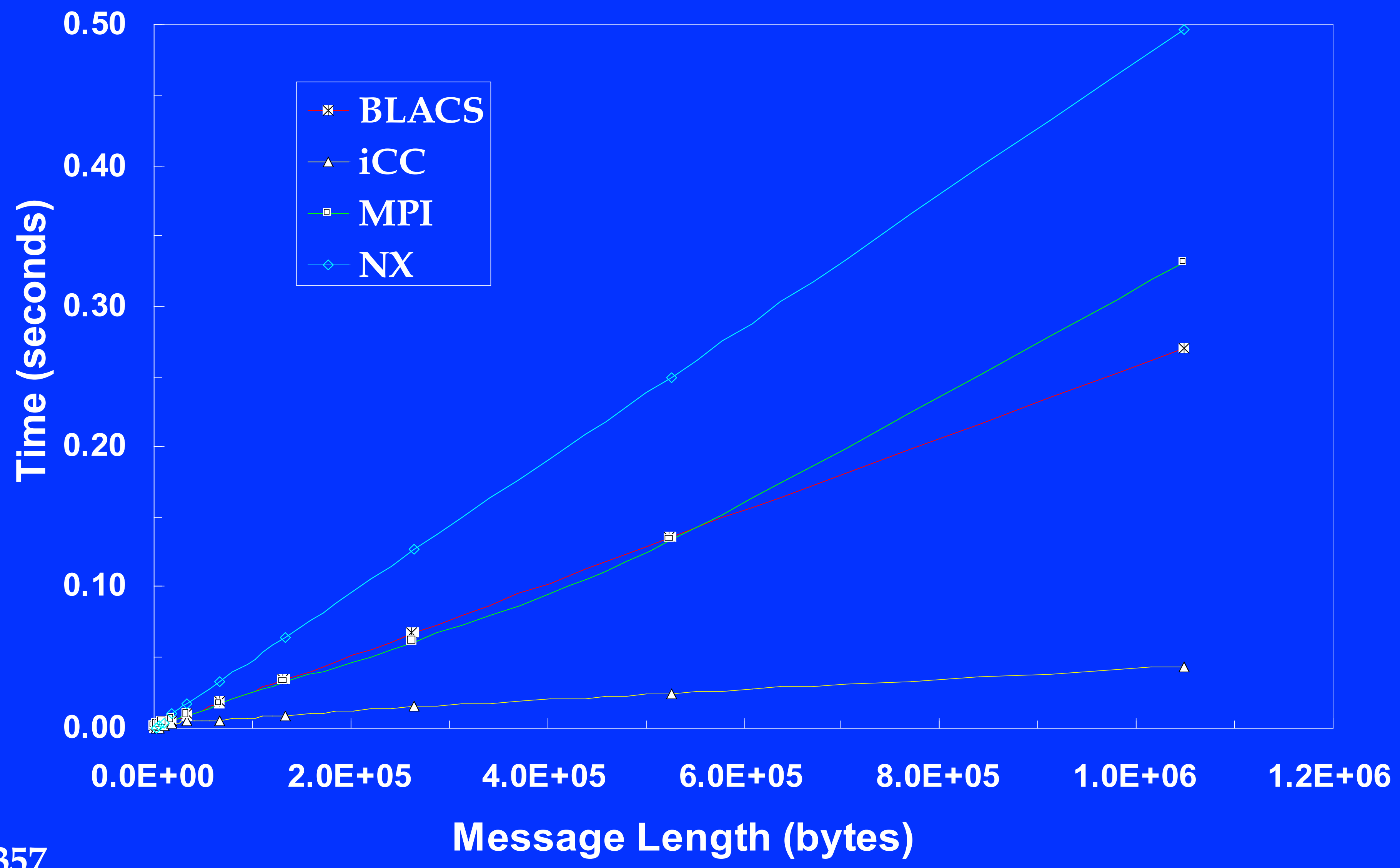
Broadcast on 16 x 32 mesh Paragon



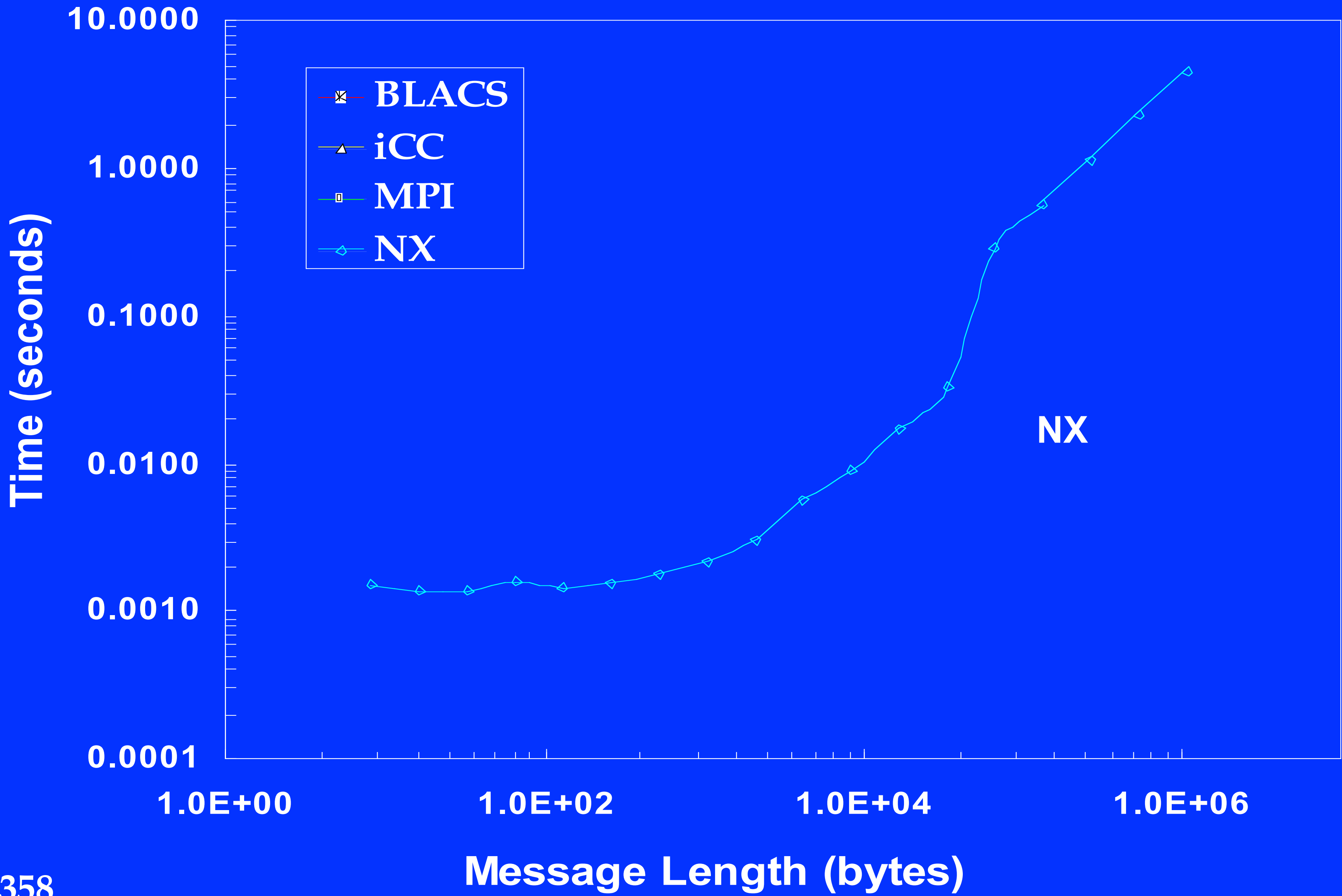
Broadcast on 16 x 32 mesh Paragon



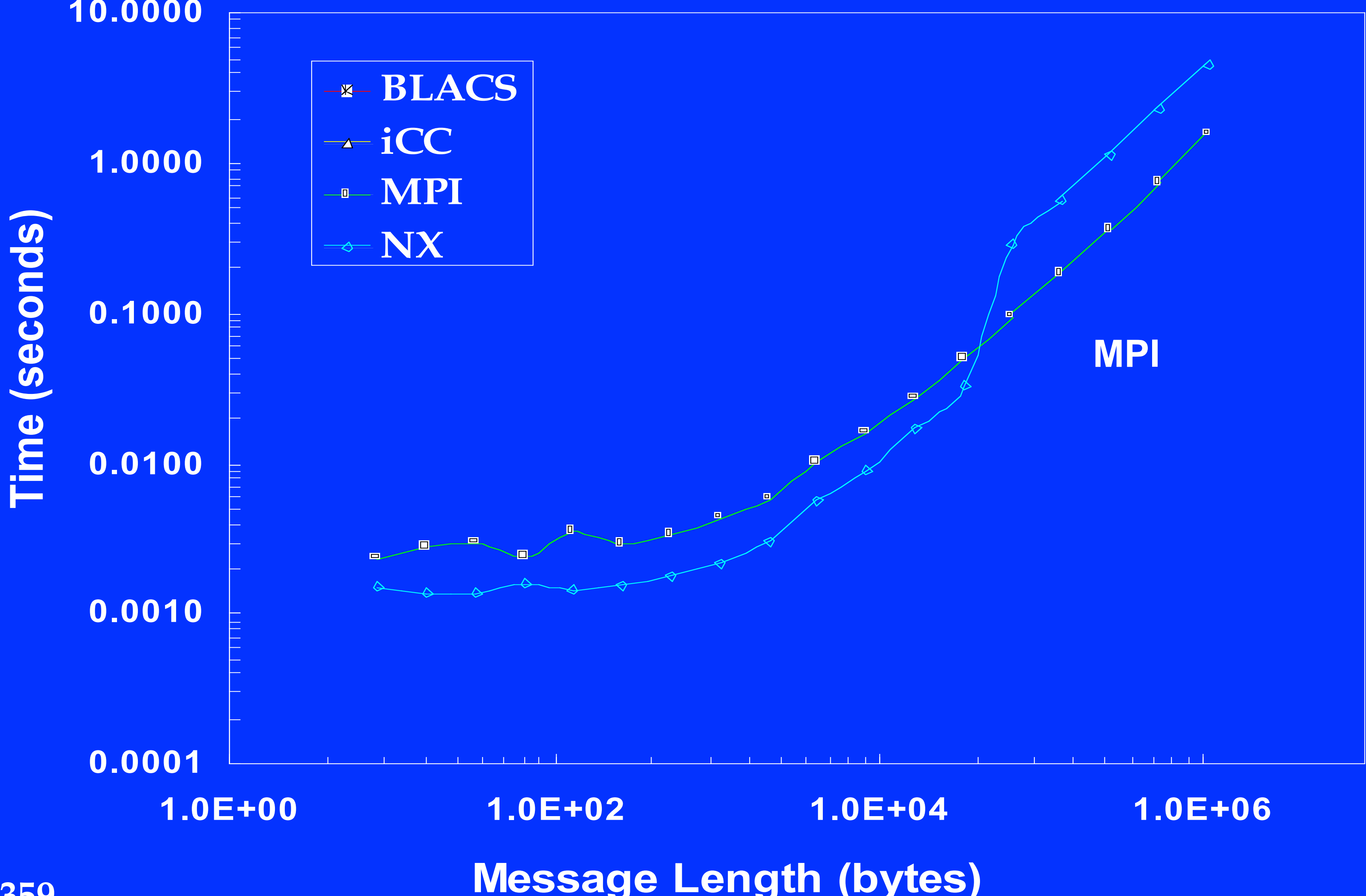
Broadcast on 16 x 32 mesh Paragon



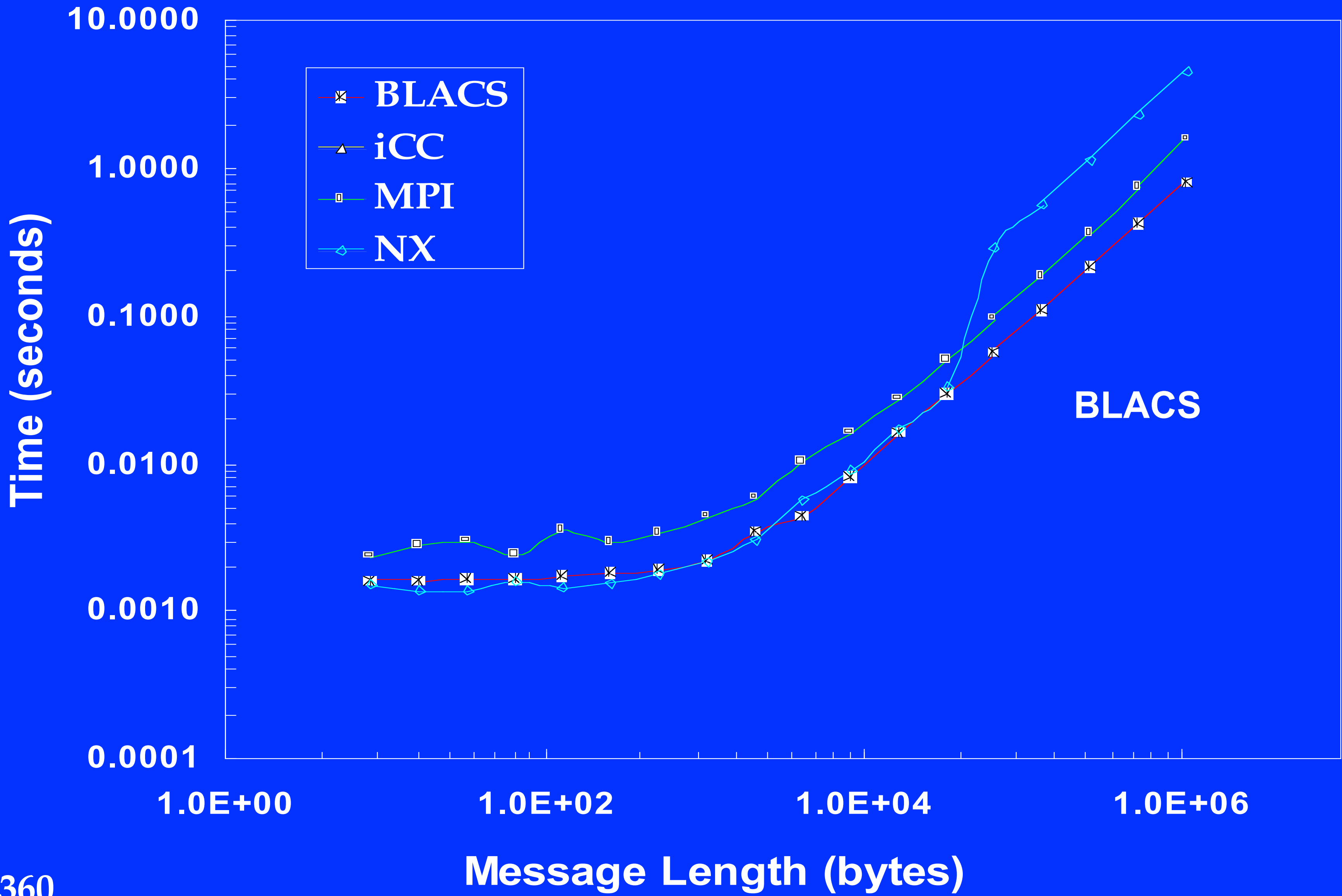
Allgather on 16 x 32 mesh Paragon



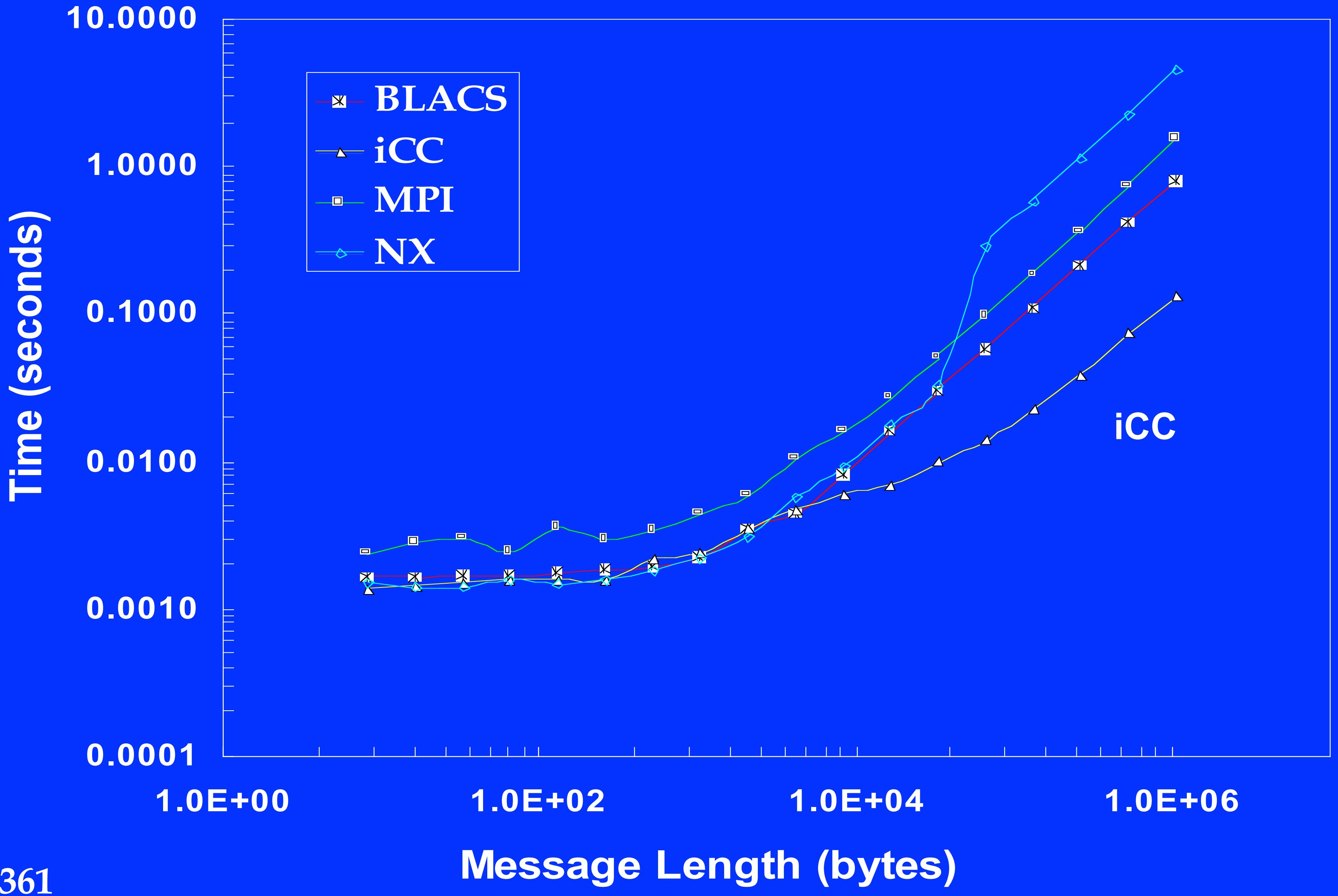
Allgather on 16 x 32 mesh Paragon



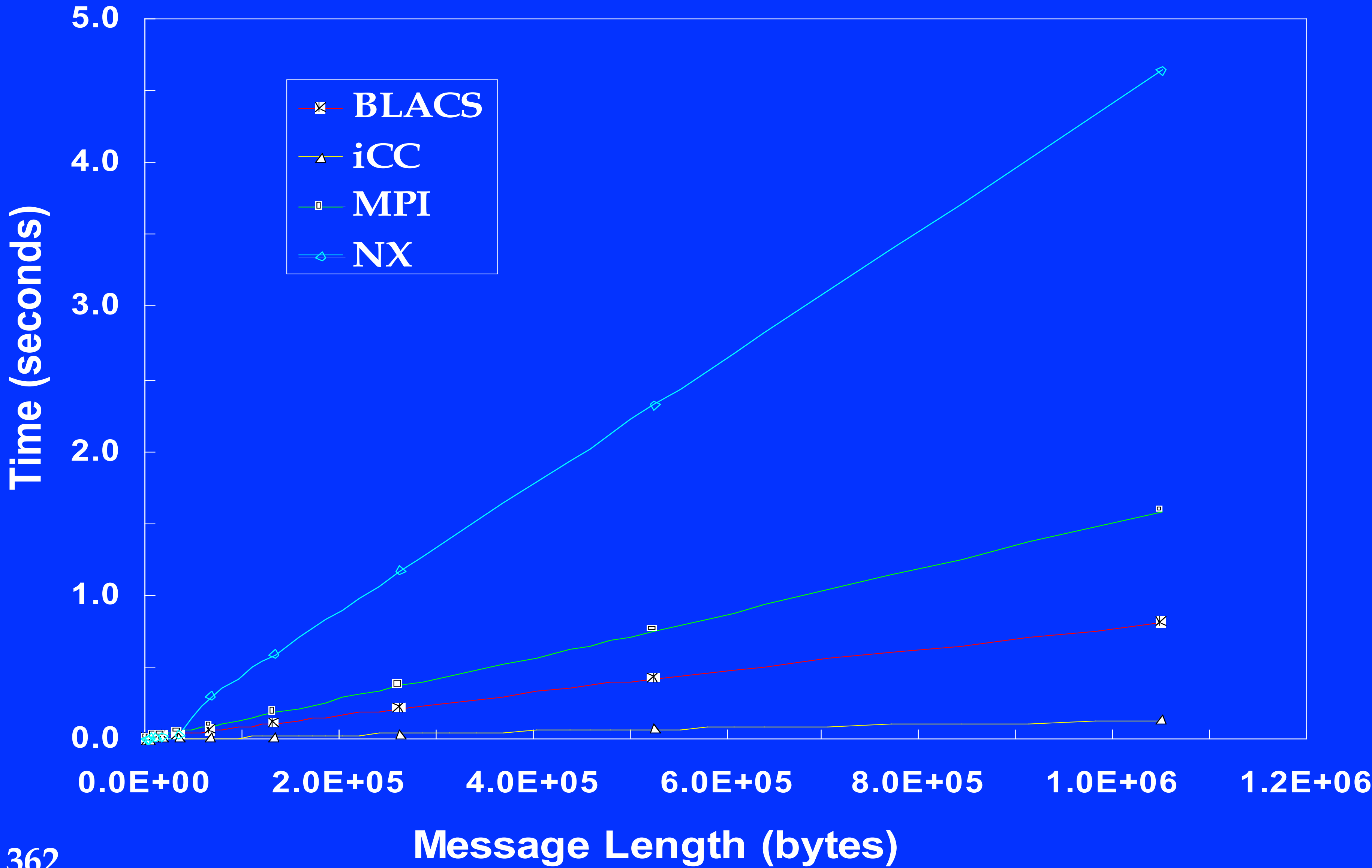
Allgather on 16 x 32 mesh Paragon



Allgather on 16 x 32 mesh Paragon



Allgather on 16 x 32 mesh Paragon



For information, contact

rvdg@cs.utexas.edu

CollMark: Collective Communicaton Benchmark

A look at the current state-of-the-art

(spring 2000)

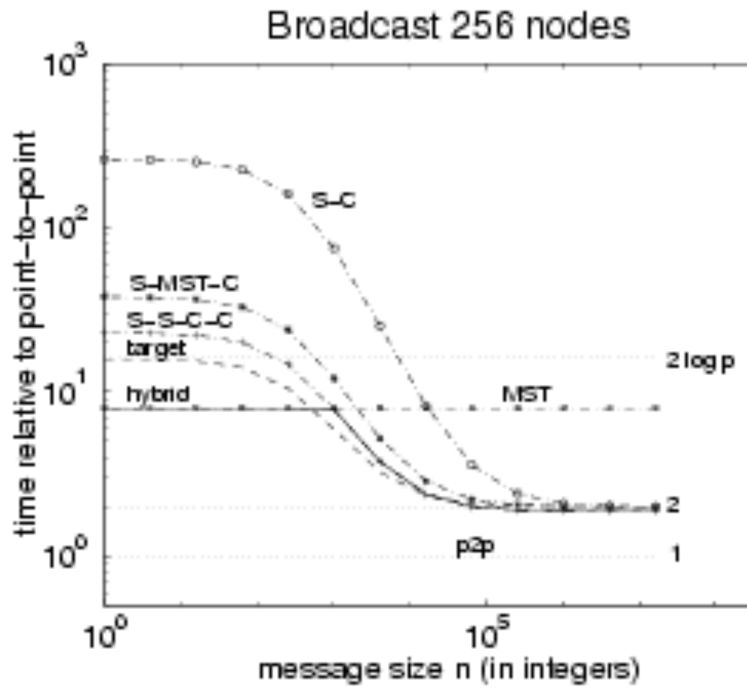
How to measure the quality of an implementation

- **Architecture independent measure of the quality of the implementation:**

$$\frac{T_{comm}(n,p)}{T_{p2p}(n)}$$

- **Ideally:**

$$\frac{T_{comm}(n,p)}{T_{p2p}(n)} \xrightarrow[n\to\infty]{} 1 \qquad or \qquad 2$$



Broadcast 256 nodes
time relative to point-to-point
message size n (in integers)
S-C
S-MST-C
S-S-C-C
target
hybrid
MST
2 log p
p2p
2
1