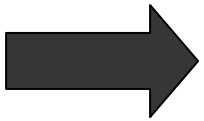
Efficient Bufferless Routing on Leveled Networks

Costas Busch
Shailesh Kelkar
Malik Magdon-Ismail

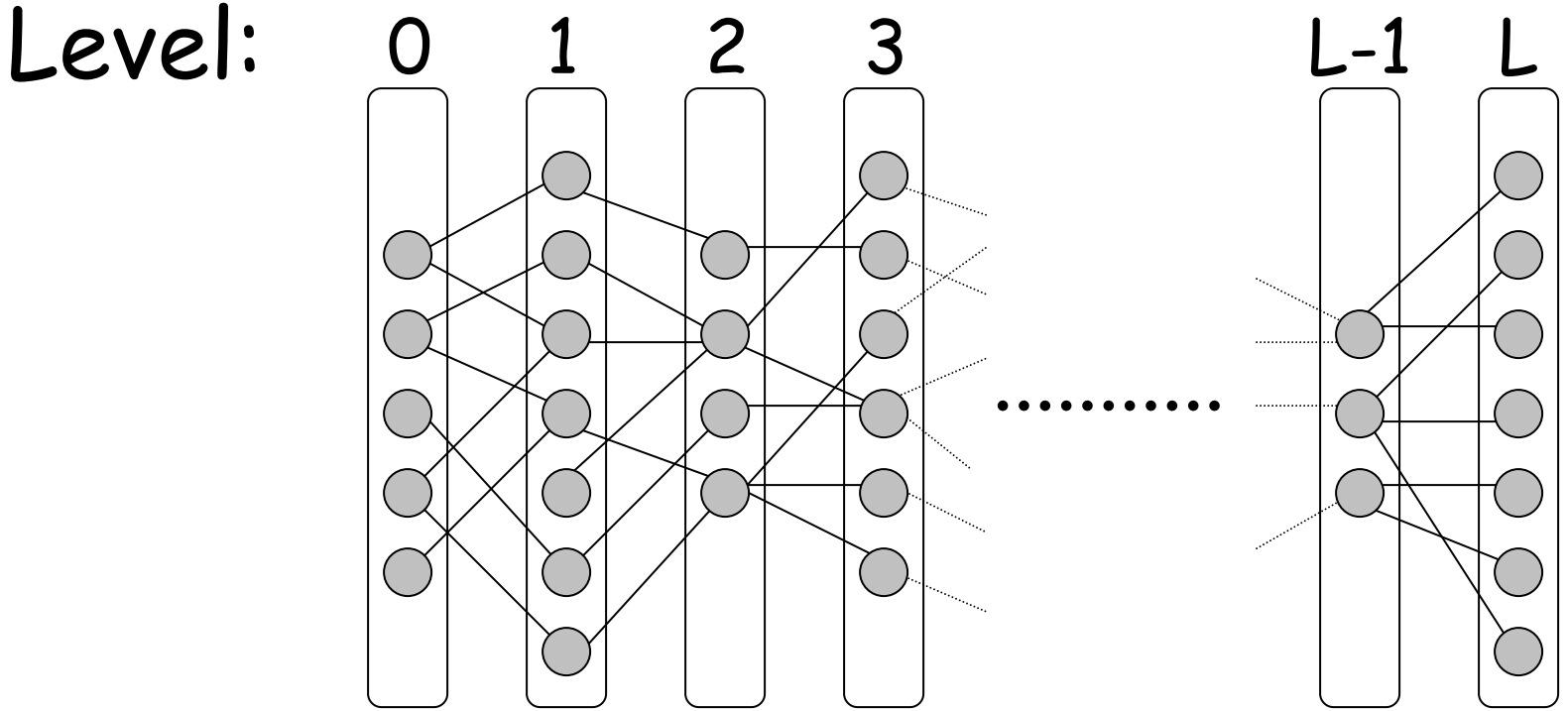
Rensselaer Polytechnic Institute

Talk Outline

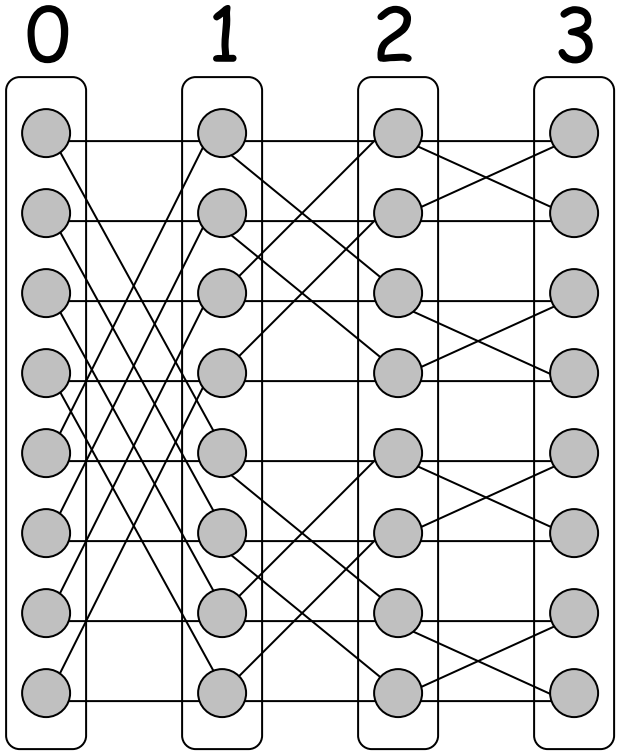


- Introduction
- Centralized Algorithm
- Distributed Algorithm
- Conclusion

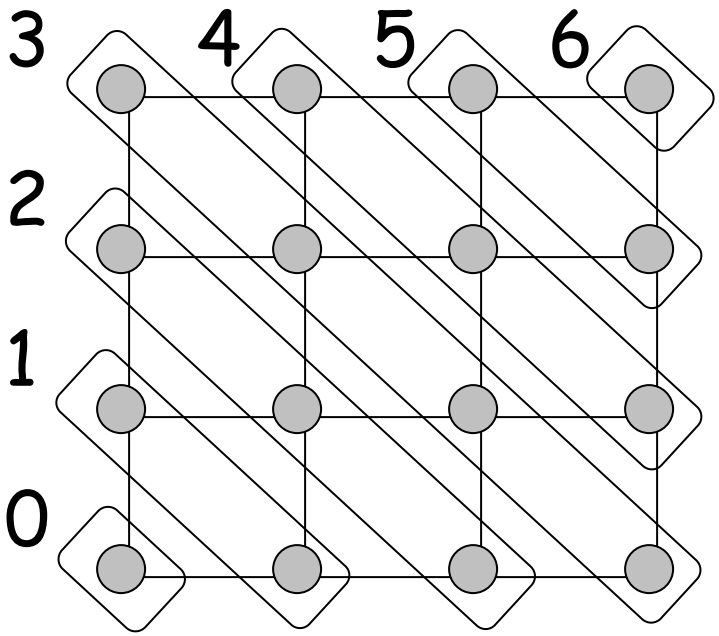
Leveled Networks



Examples of Leveled Networks



Butterfly

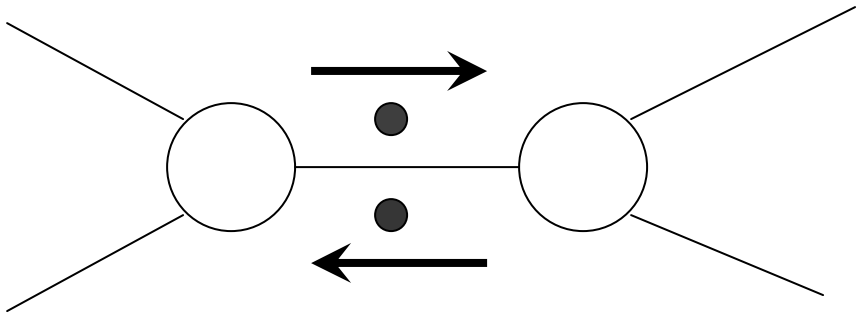


Mesh

Network Model

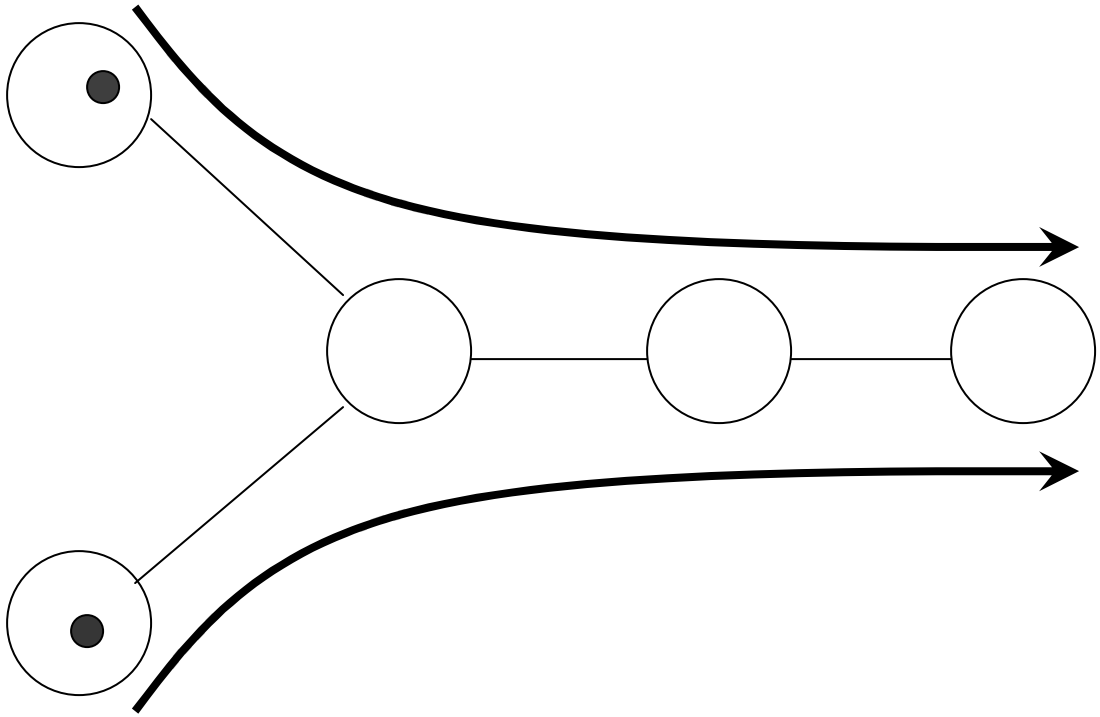
- Synchronous network (time steps)
- Bi-directional links

One packet per direction, per time step



Buffer-less nodes

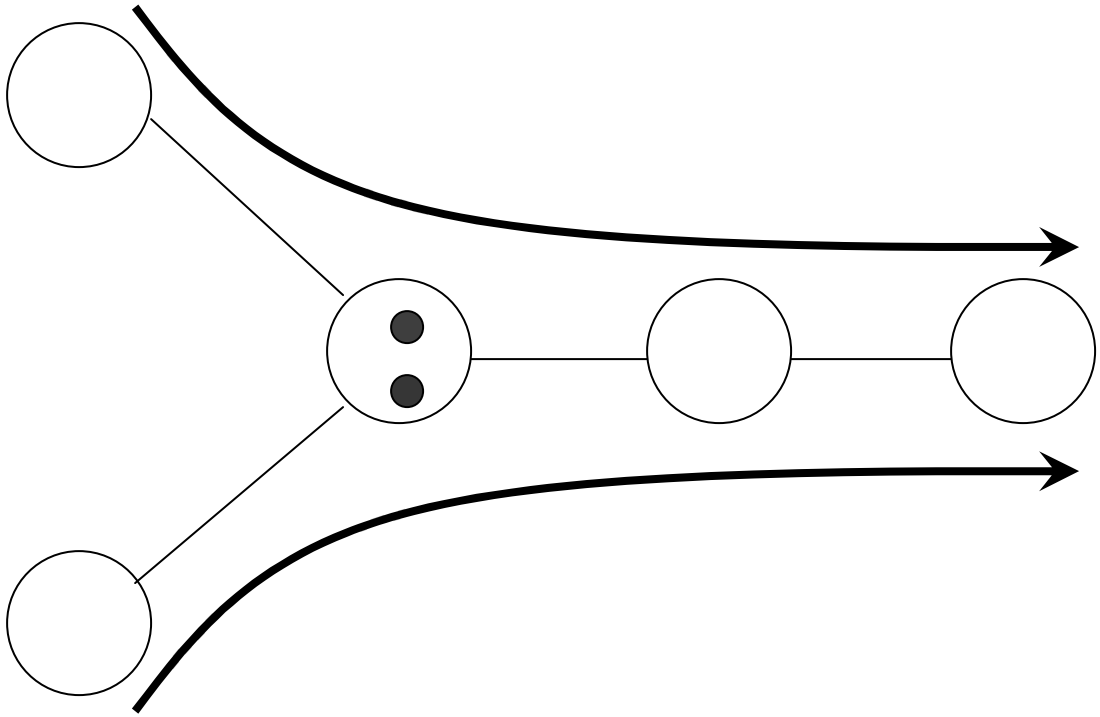
Time 0



Packets are always moving

Buffer-less nodes

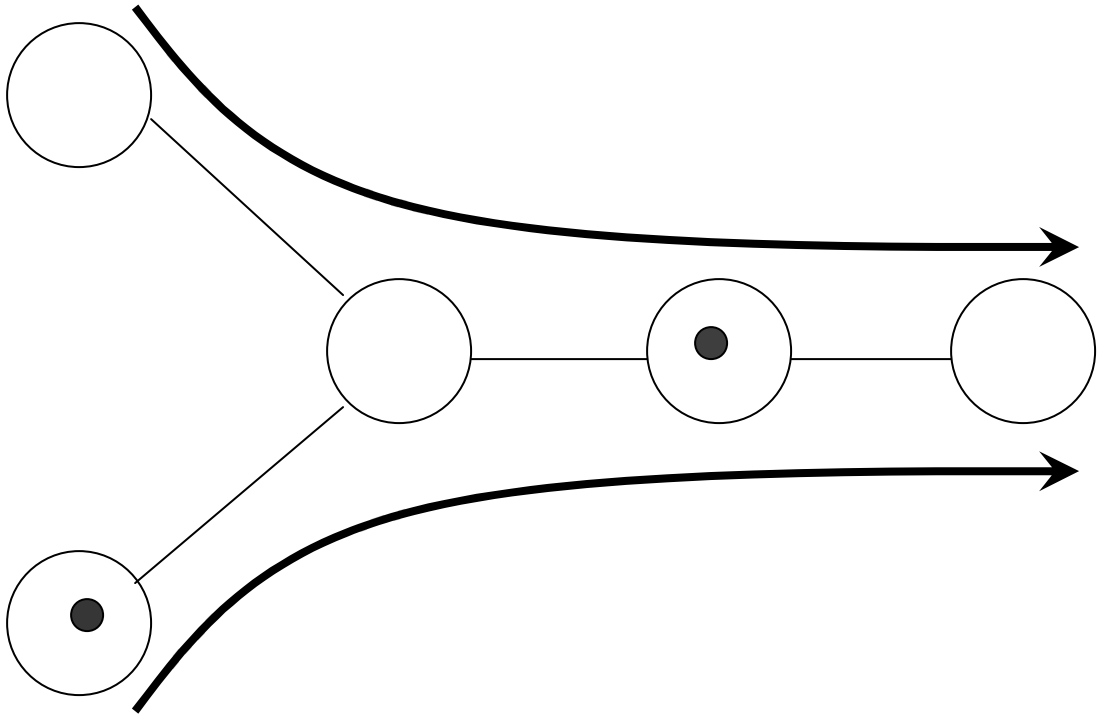
Time 1



Packets are always moving

Buffer-less nodes

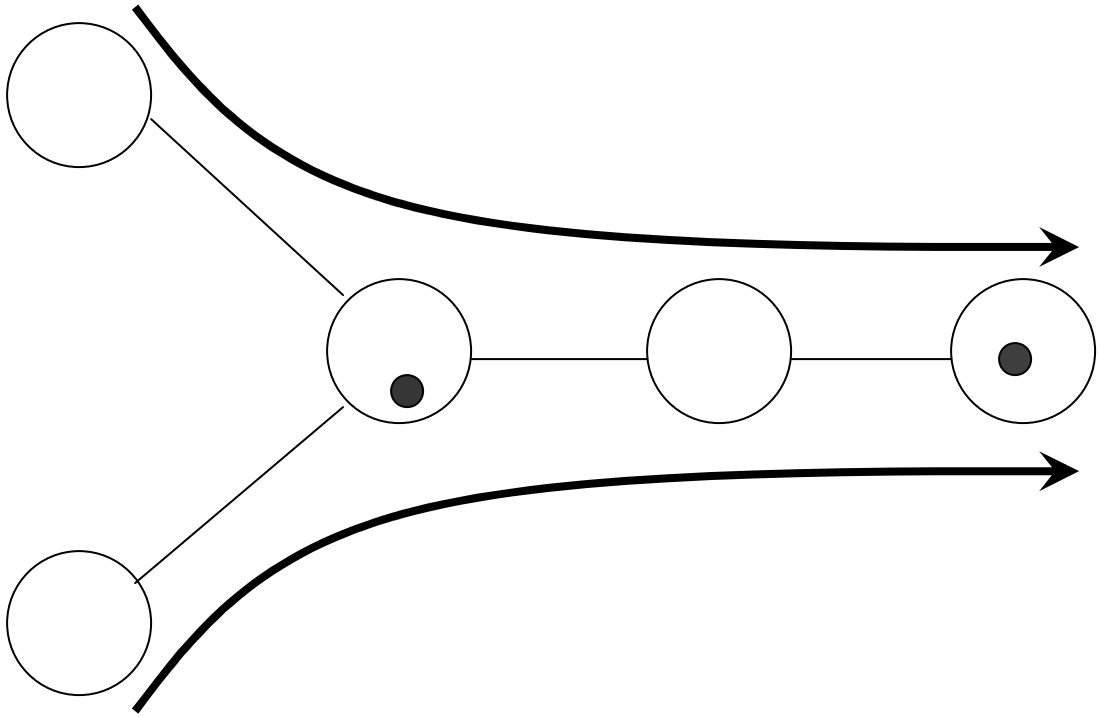
Time 2



Packets are always moving

Buffer-less nodes

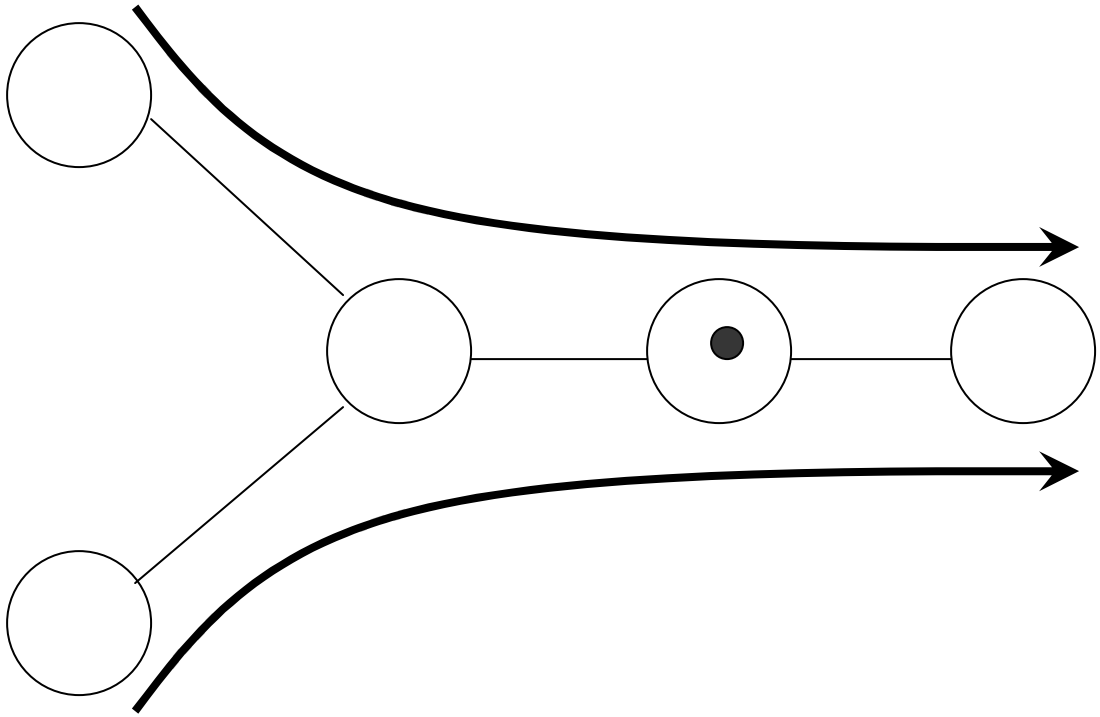
Time 3



Packets are always moving

Buffer-less nodes

Time 4



Packets are always moving

Bufferless routing is interesting:

- Optical networks
- Simple hardware implementations
- Works well in practice:

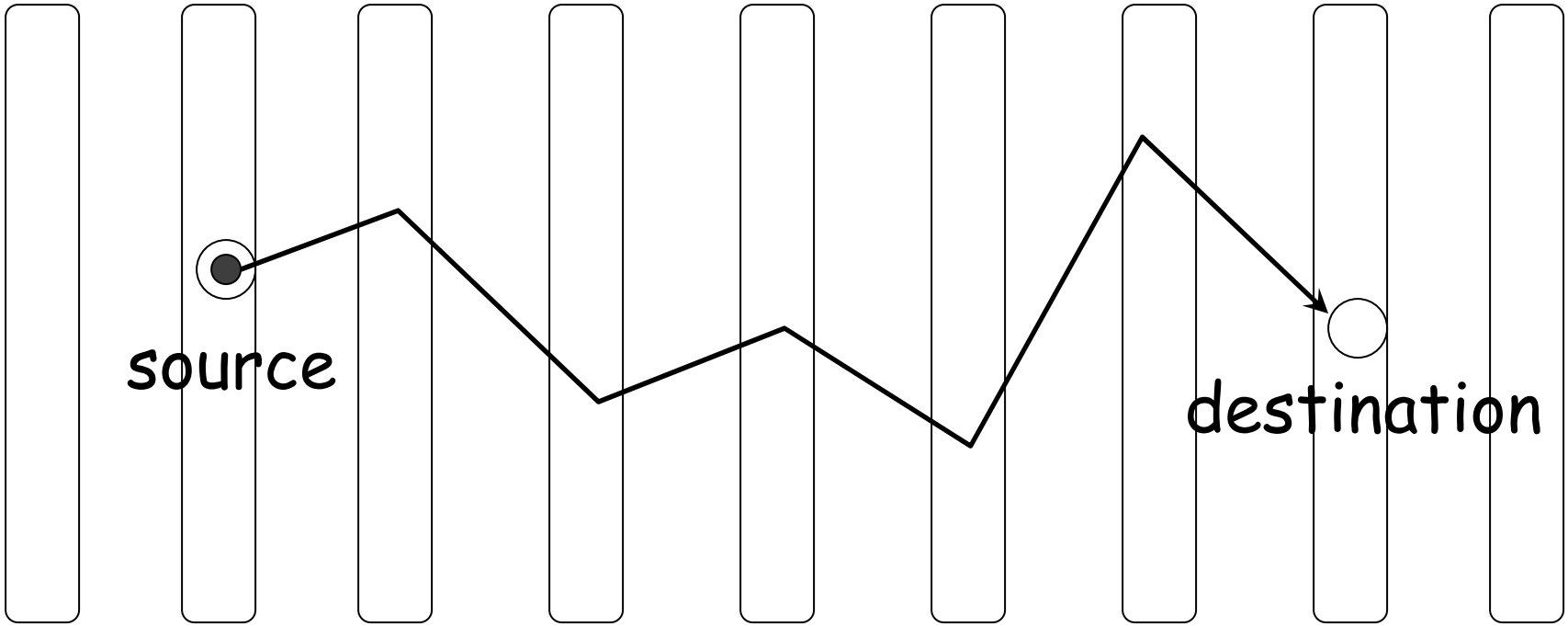
  Bartzis *et al.*: *EUROPAR 2000*

  Maxemchuck: *INFOCOM 1989*

Routing Time: the time until the last packet is absorbed

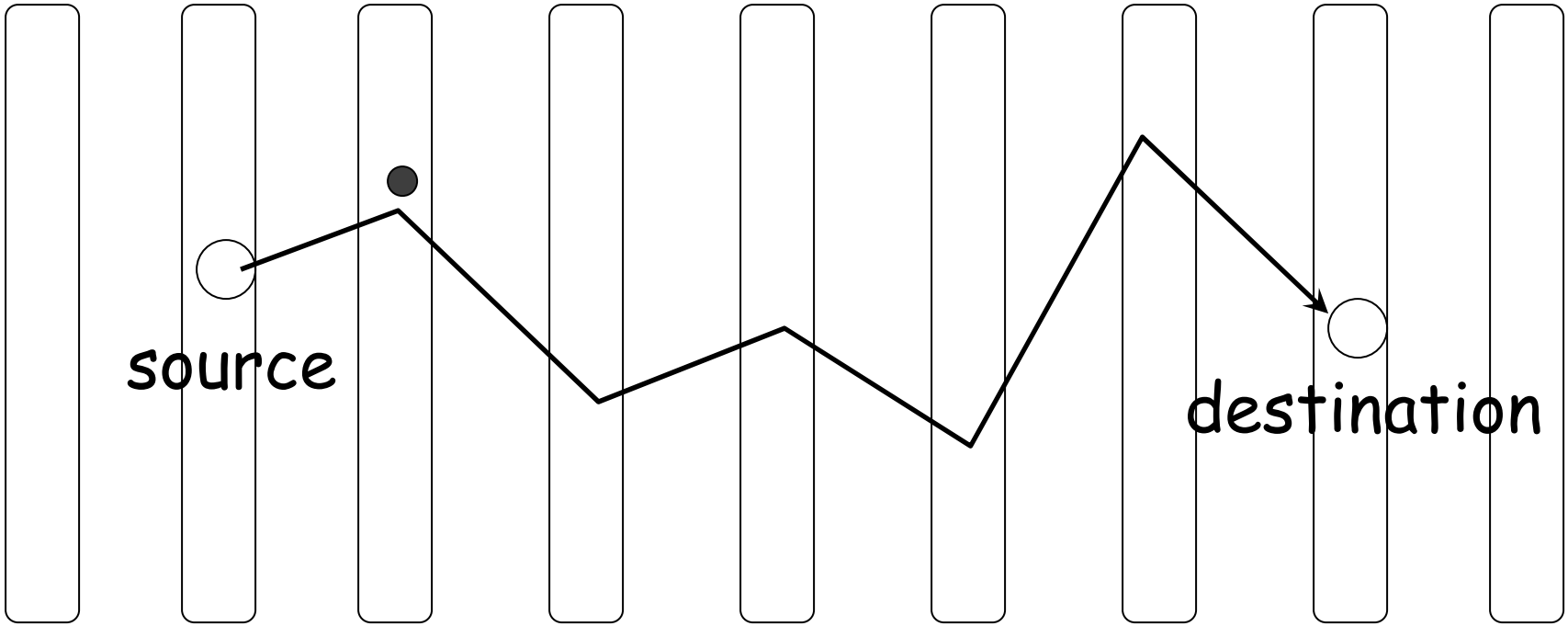
Objective: Minimize Routing Time

Each packet has a pre-selected path

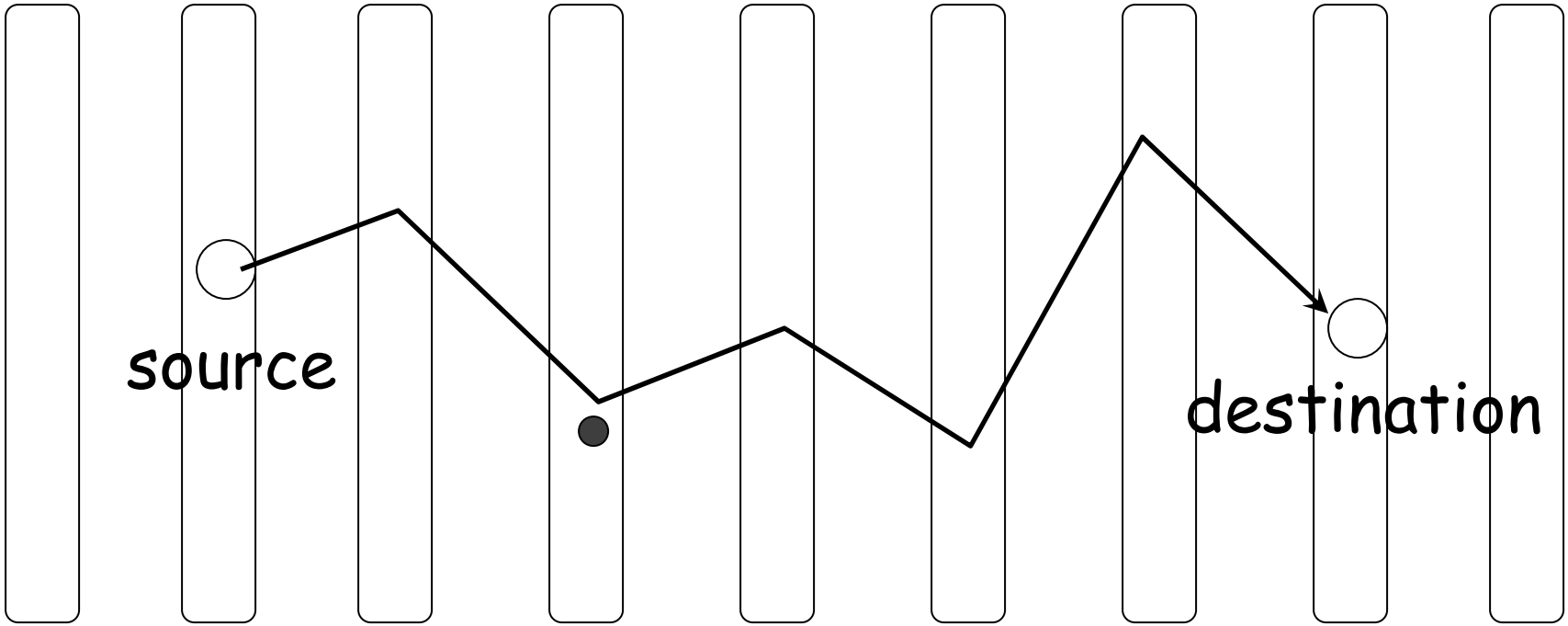


Packet path is from left to right

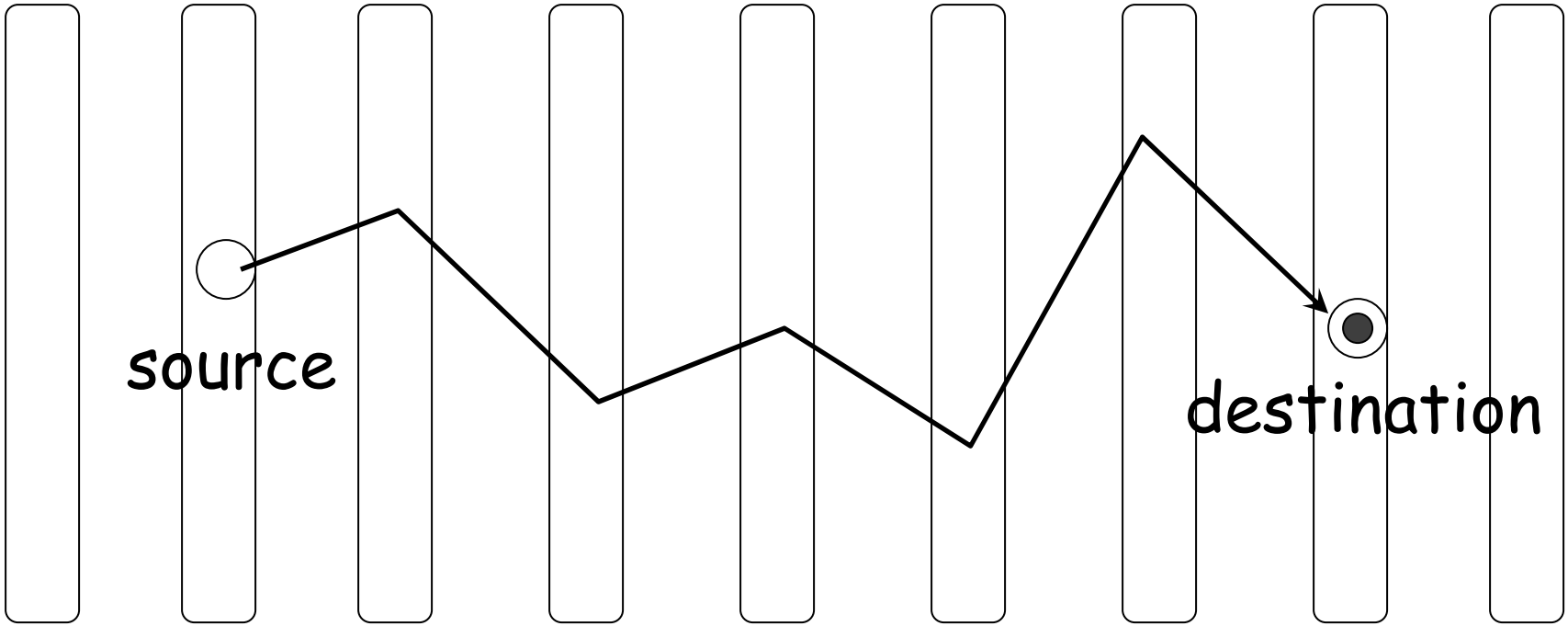
The packet follows the pre-selected path



The packet follows the pre-selected path

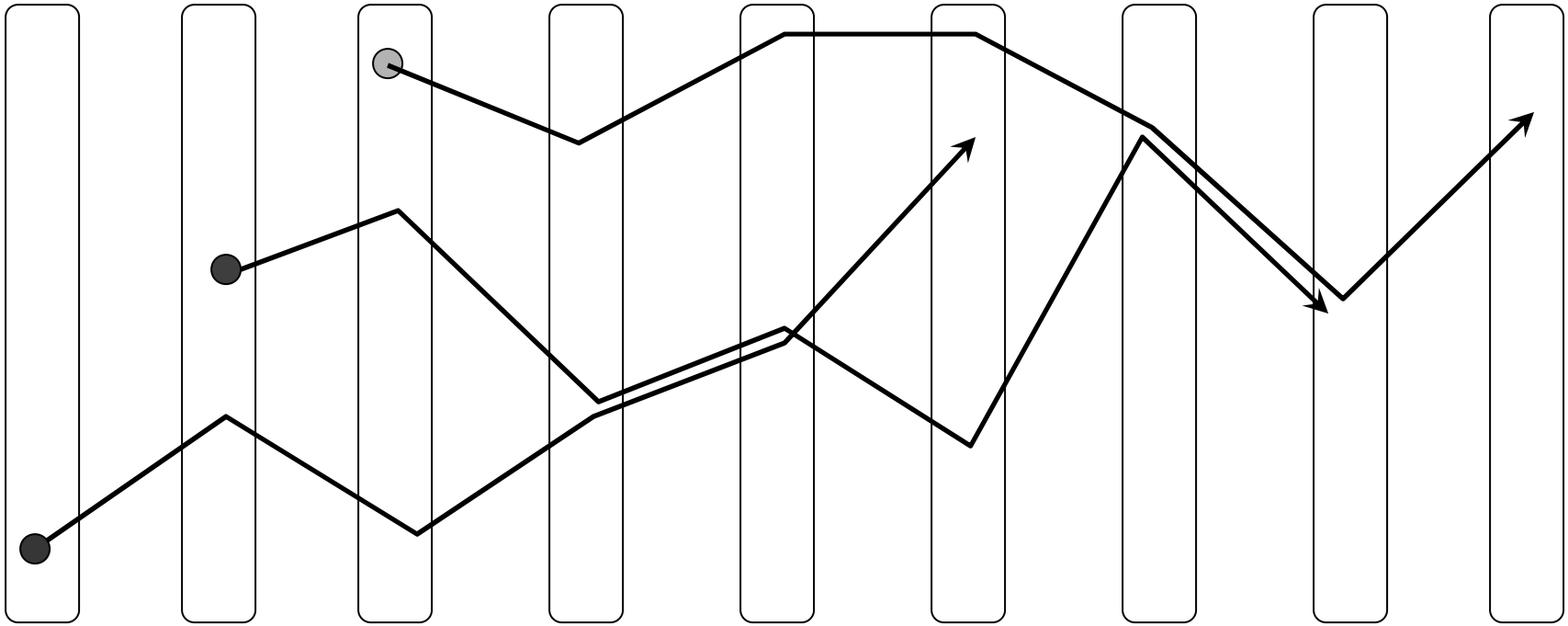


The packet follows the pre-selected path



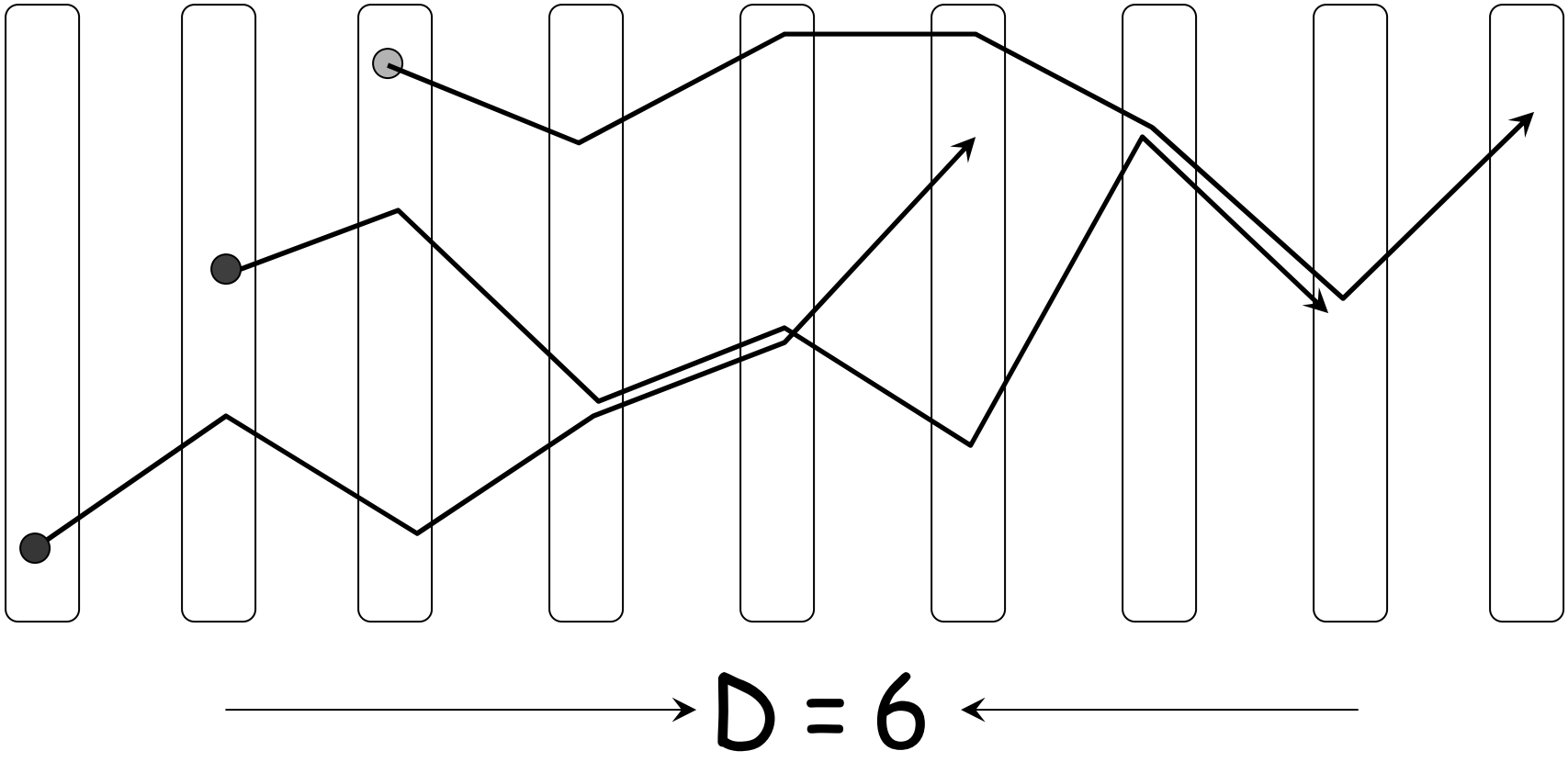
There are N packets

Each packet has its own path



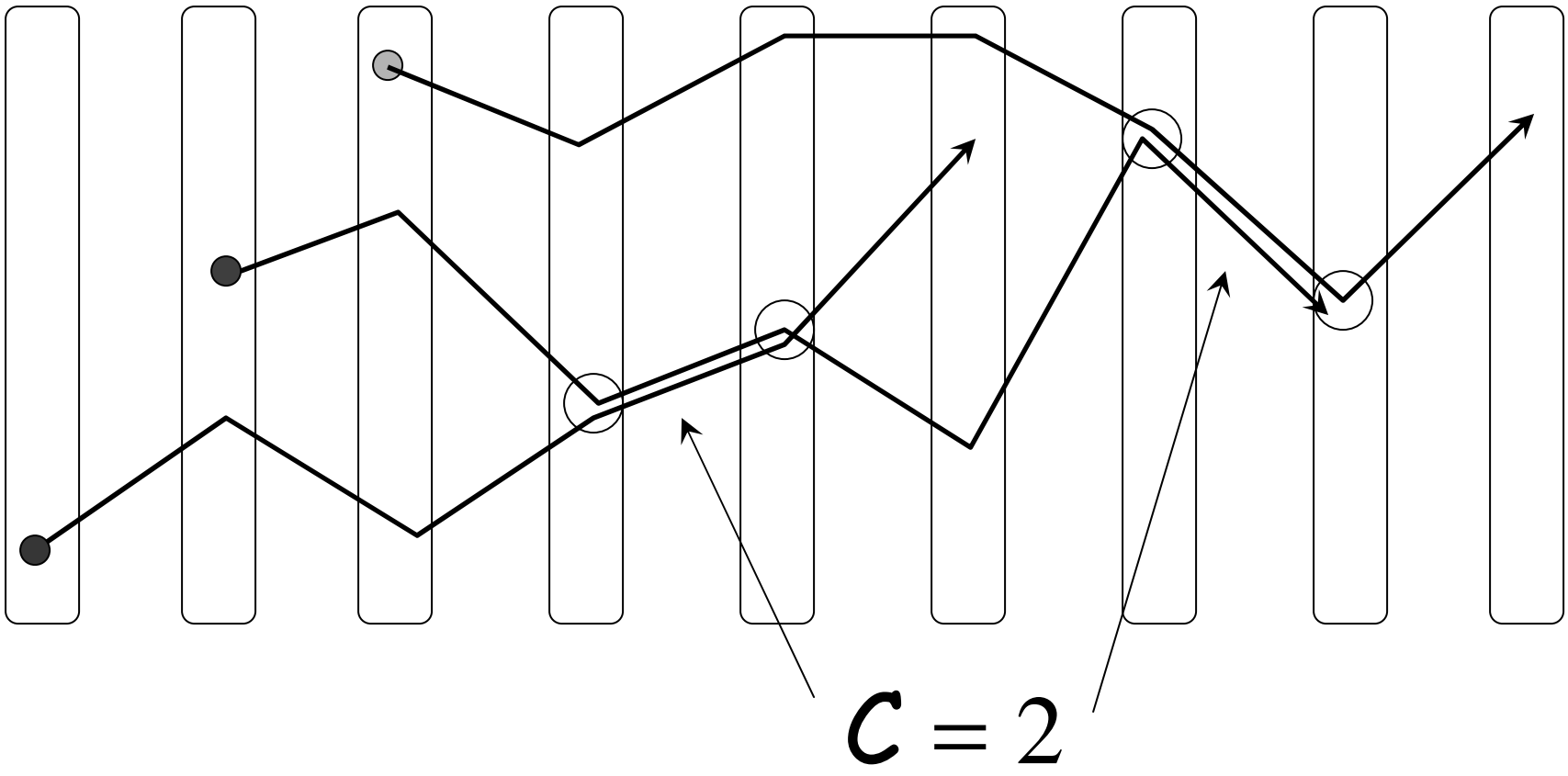
N = 3

Dilation D: The maximum length of any path



Routing time: $\Omega(D)$

Congestion C: The maximum number of packets traversing any edge



Routing time: $\Omega(C)$

Lower bound on Routing Time:

$$\Omega(C + D)$$

Congestion Dilation

We want algorithms with Routing Time close to:

$$C + D$$

Our Contributions

- Centralized Algorithm:

$$O(C \cdot \log(DN) + D)$$

- Distributed Algorithm:

$$O(C \cdot \log^2(DN) + D \cdot \log(DN))$$

Both algorithms are randomized

Results hold with high probability $\left(1 - \frac{1}{DN}\right)$

N : number of packets

Related Work

Networks with buffers

Leveled networks: $O(C + L + \log N)$

Leighton, Maggs, Ranade, Rao: *J. Algorithms 1992*

Arbitrary networks: $O(C + D)$

[Leighton - Maggs - Rao, Combinatorica 94]

[BS99, LMR99, MV99, OR97, RT96]

Bufferless networks

Mesh	[BRST93, BES97, BHS98, BU96, BHW00]
Hypercube	[BH85, BC95, FR92, H91]
Trees	[BMMW04, RSW00, BMMW]
Leveled	[BBPRRS96, B02]

Vertex-symmetric [MS95]

Arbitrary networks [BMM04]

Most related work

Arbitrary networks: $O((C+D)\cdot \log^3(n+N))$

Busch, Magdon-Ismail, Mavronicolas WAOA'04

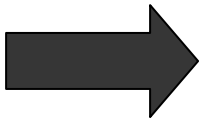
Leveled Networks: $O((C+D)\cdot \log^9(LN))$

Busch TCS'04

Leveled networks in different routing model:

Bhatt, Bilardi, Pucci, Ranade, Rosenberg, Scwabe TC'96

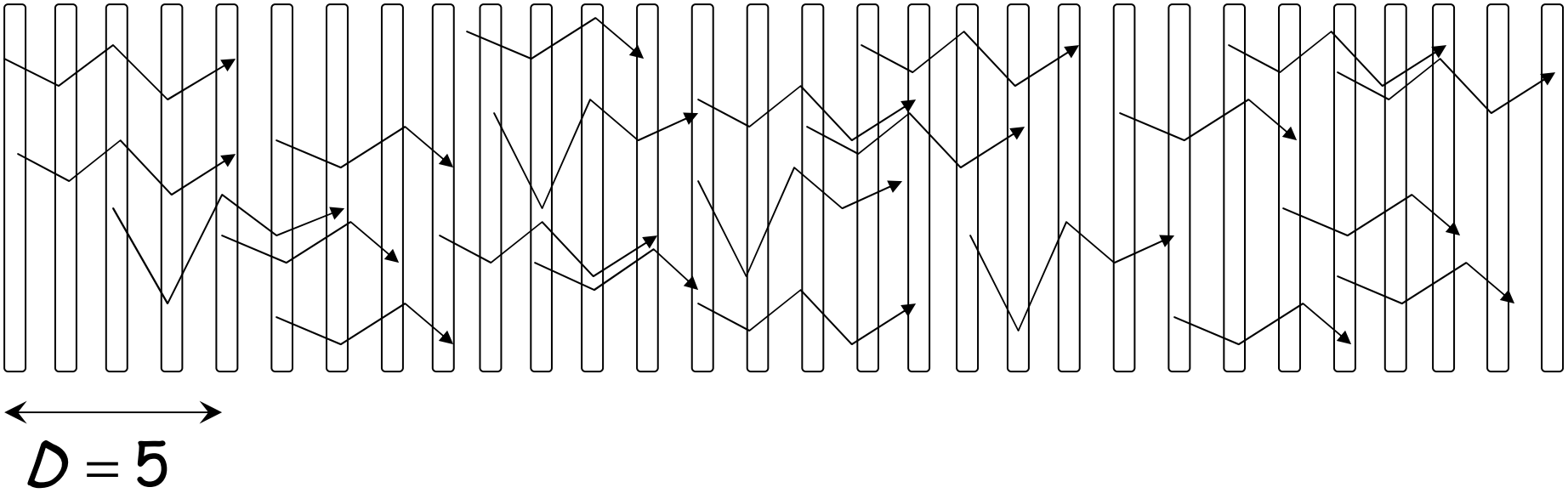
Talk Outline



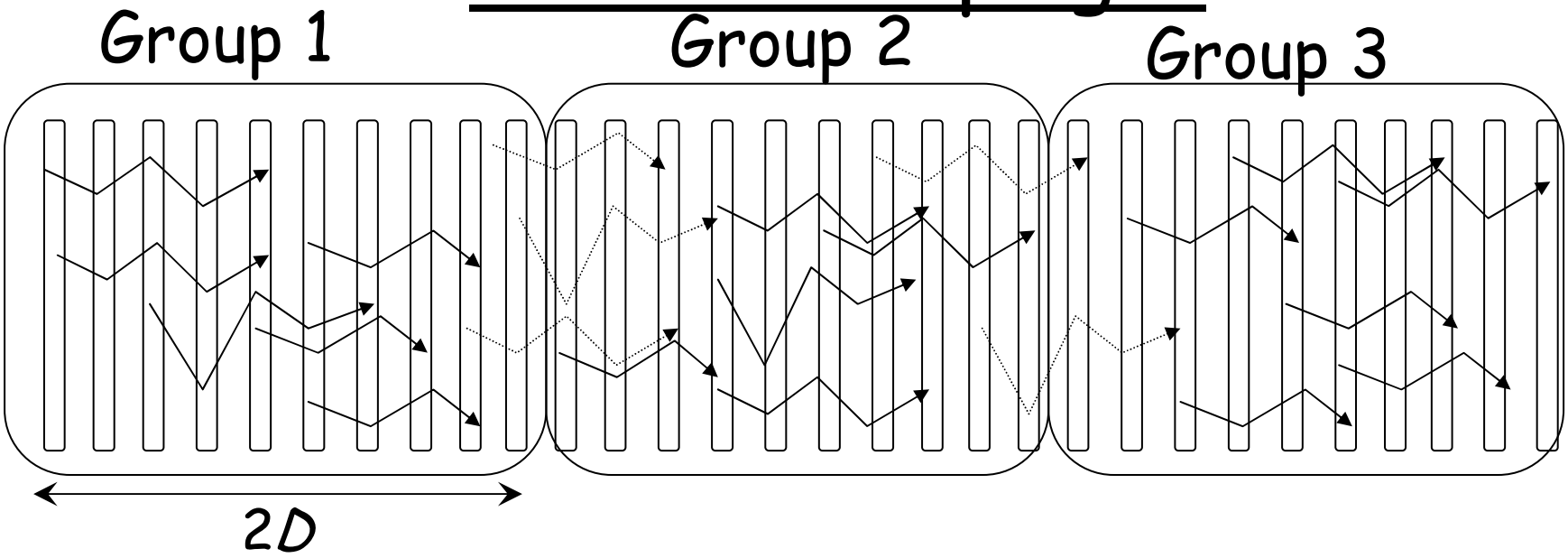
- Introduction
- Centralized Algorithm
- Distributed Algorithm
- Conclusion

Centralized Algorithm

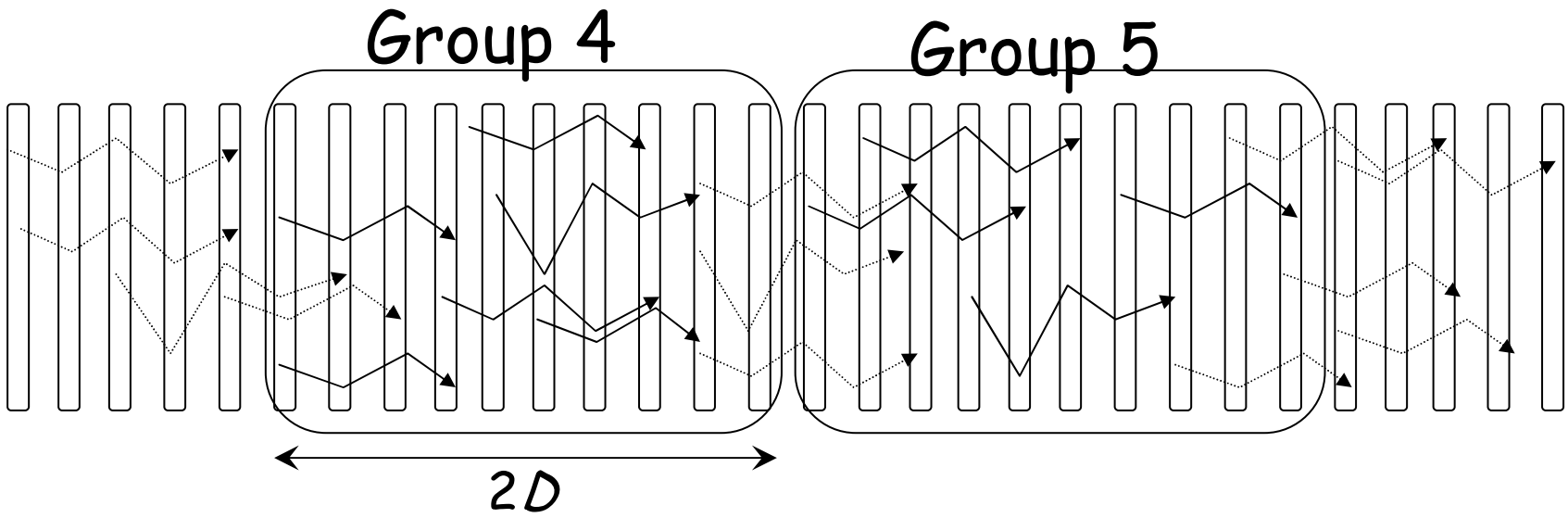
A central node knows all the parameters of the problem and computes a packet schedule


D = 5

Packet Grouping A



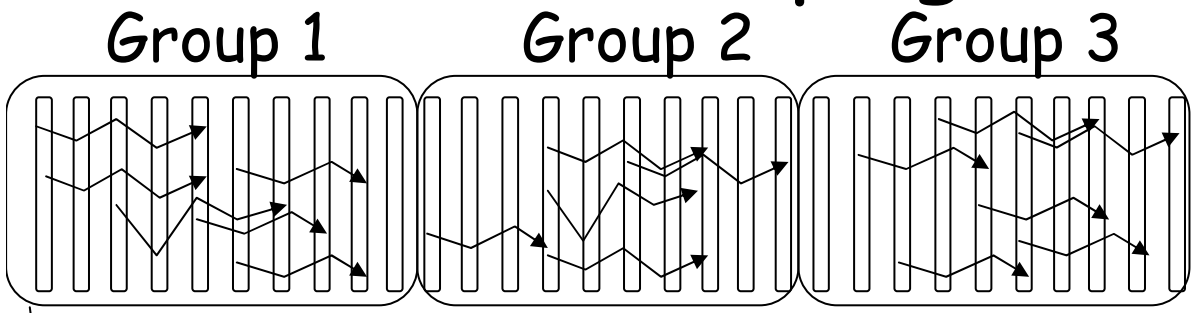
Packet Grouping B



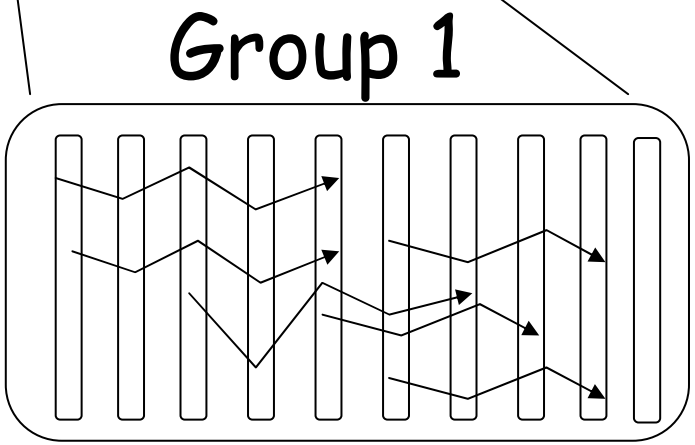
1. Send packets of Grouping A
2. Send packets of Grouping B

Increases routing time by only a factor of 2

Packet Grouping A

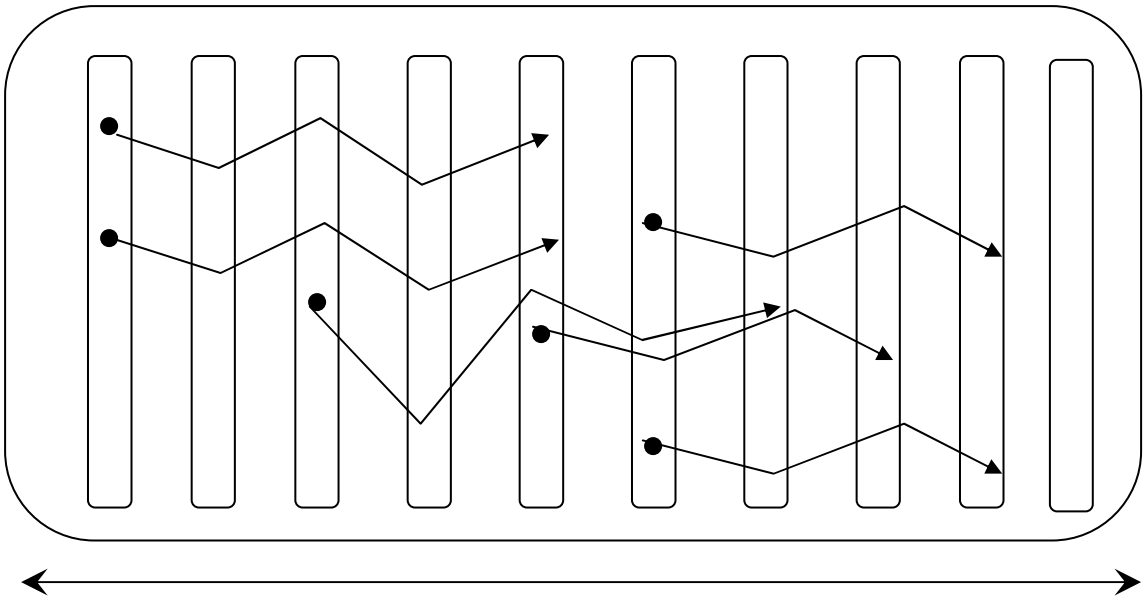


Packets in different groups can be sent simultaneously

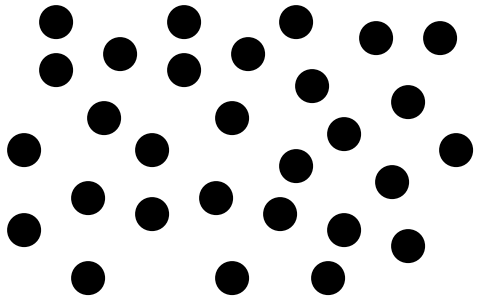


We focus only in one group

Group 1



Π set of packets

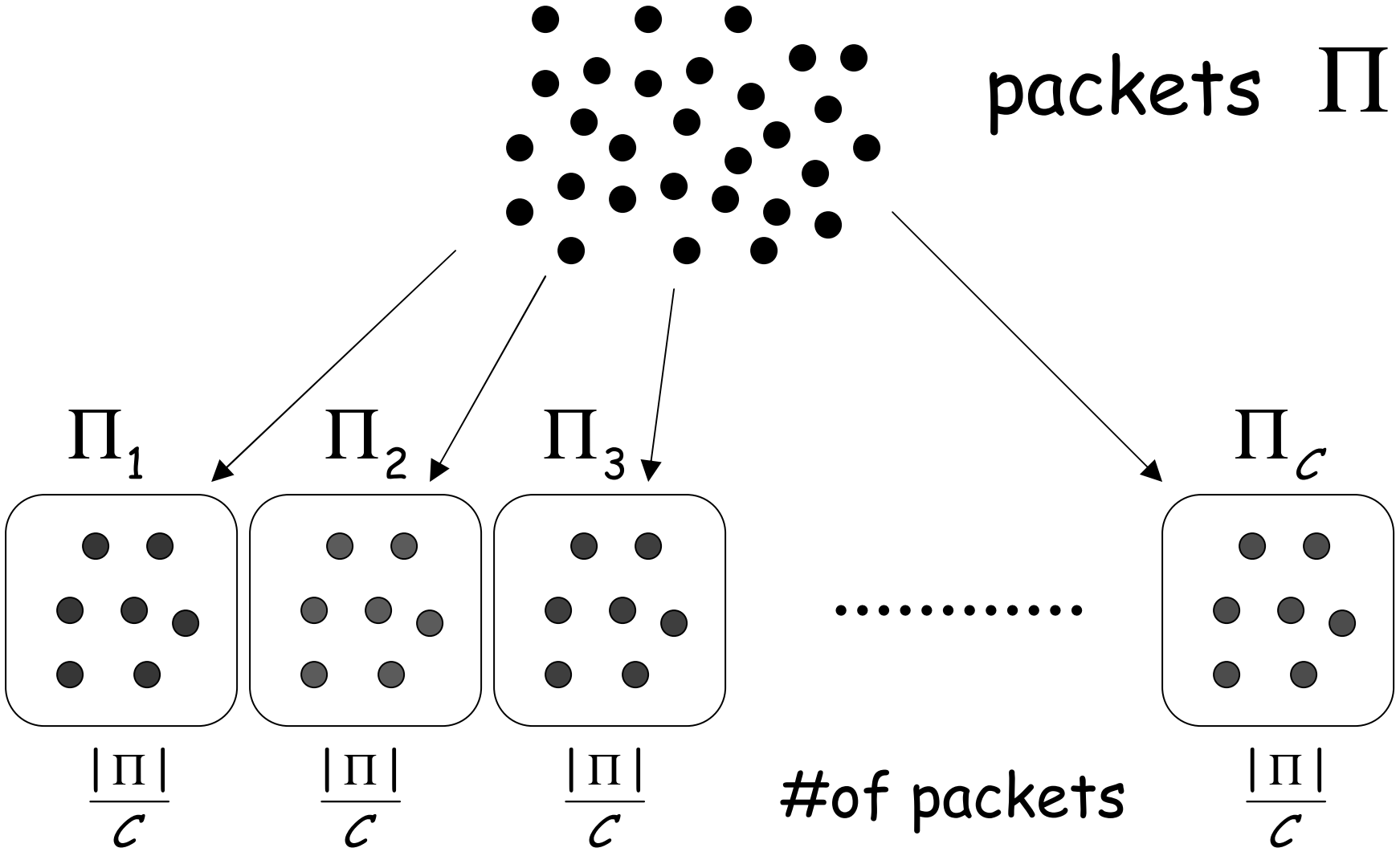


$|\Pi| \leq N$

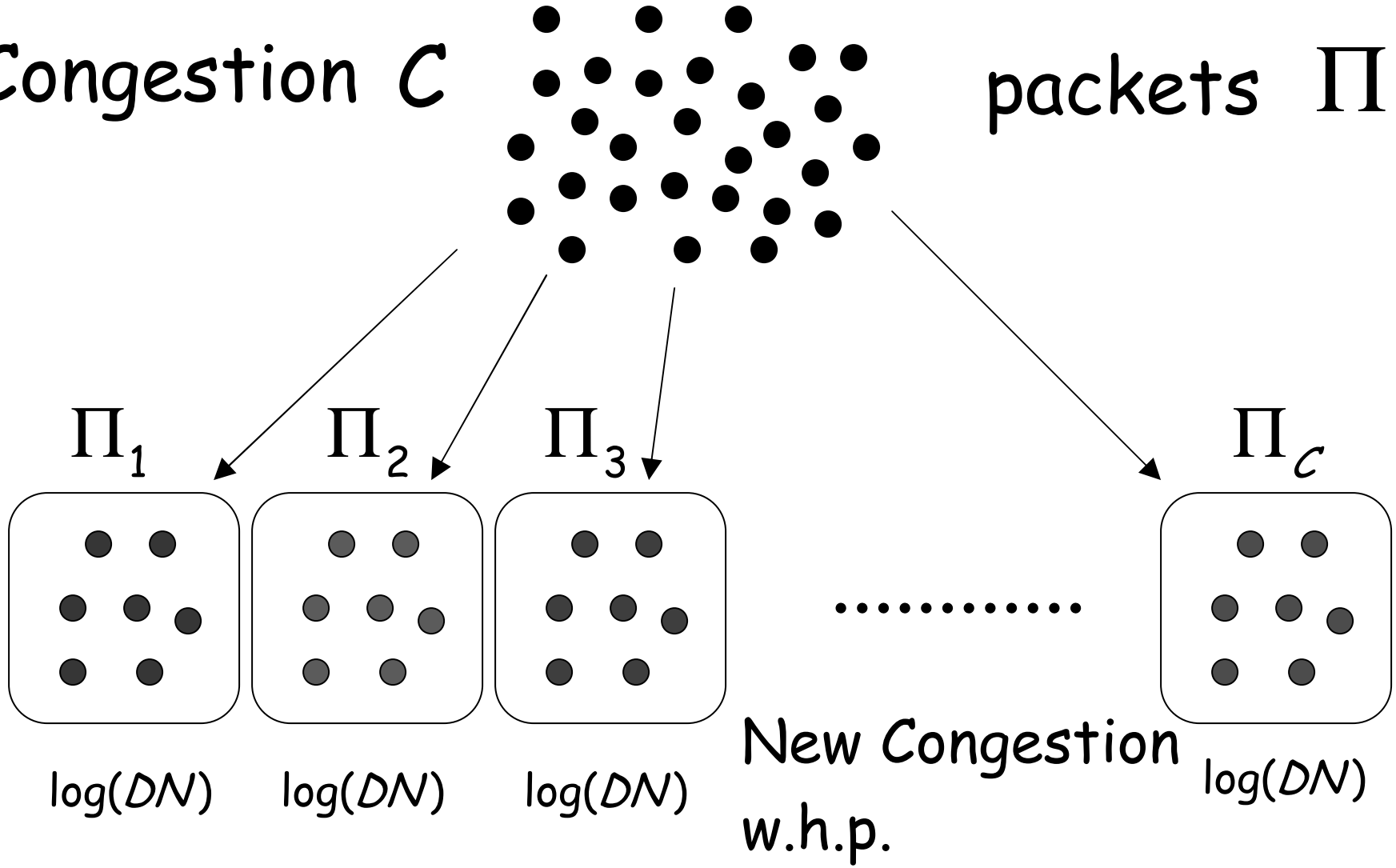
Congestion $\leq C$

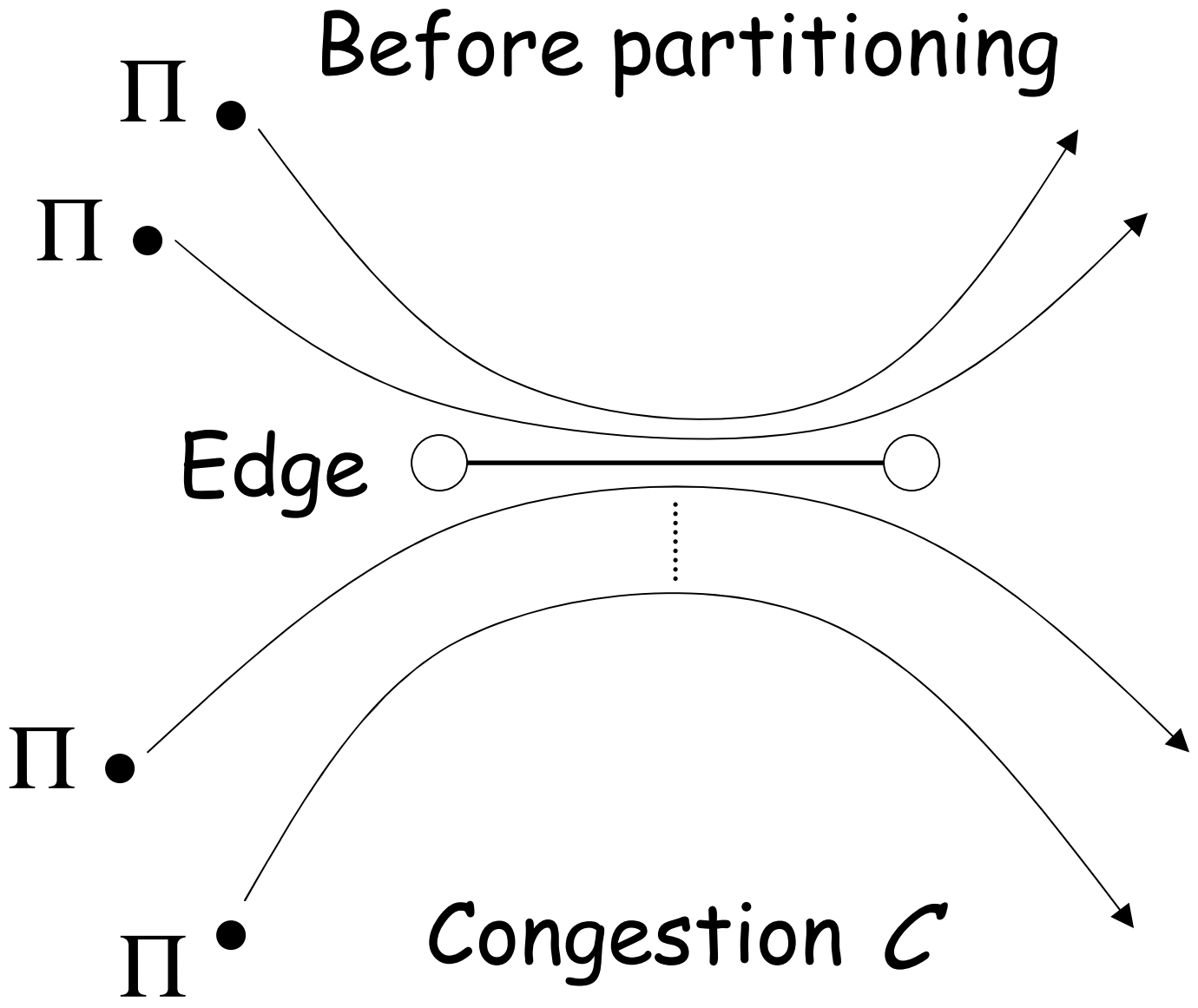
Dilation $\leq D$

Partition the packets randomly and uniformly into C sets

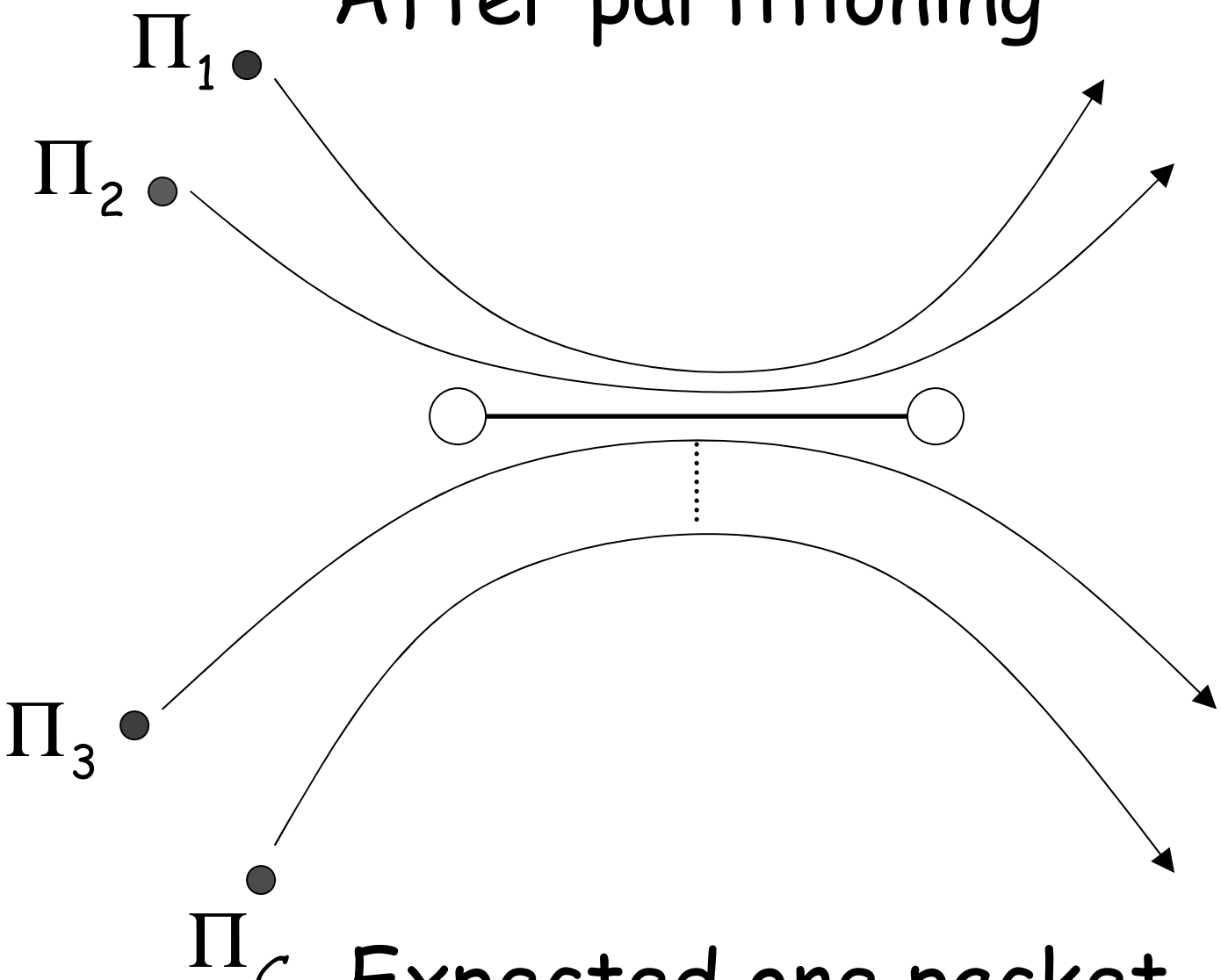


Benefit: congestion drops

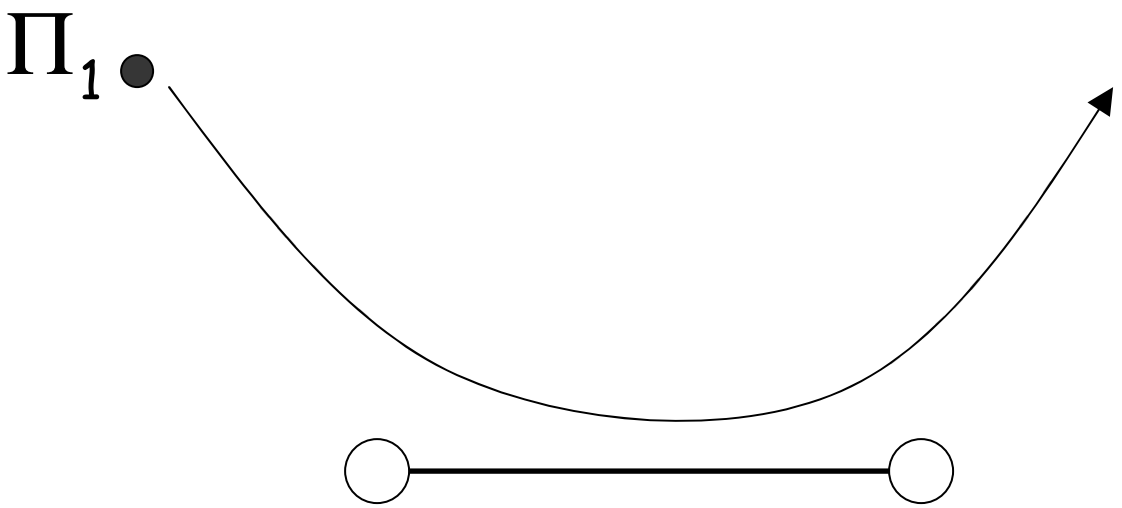



Before partitioning
Π
Π
Edge
Π
Π
Congestion C

After partitioning



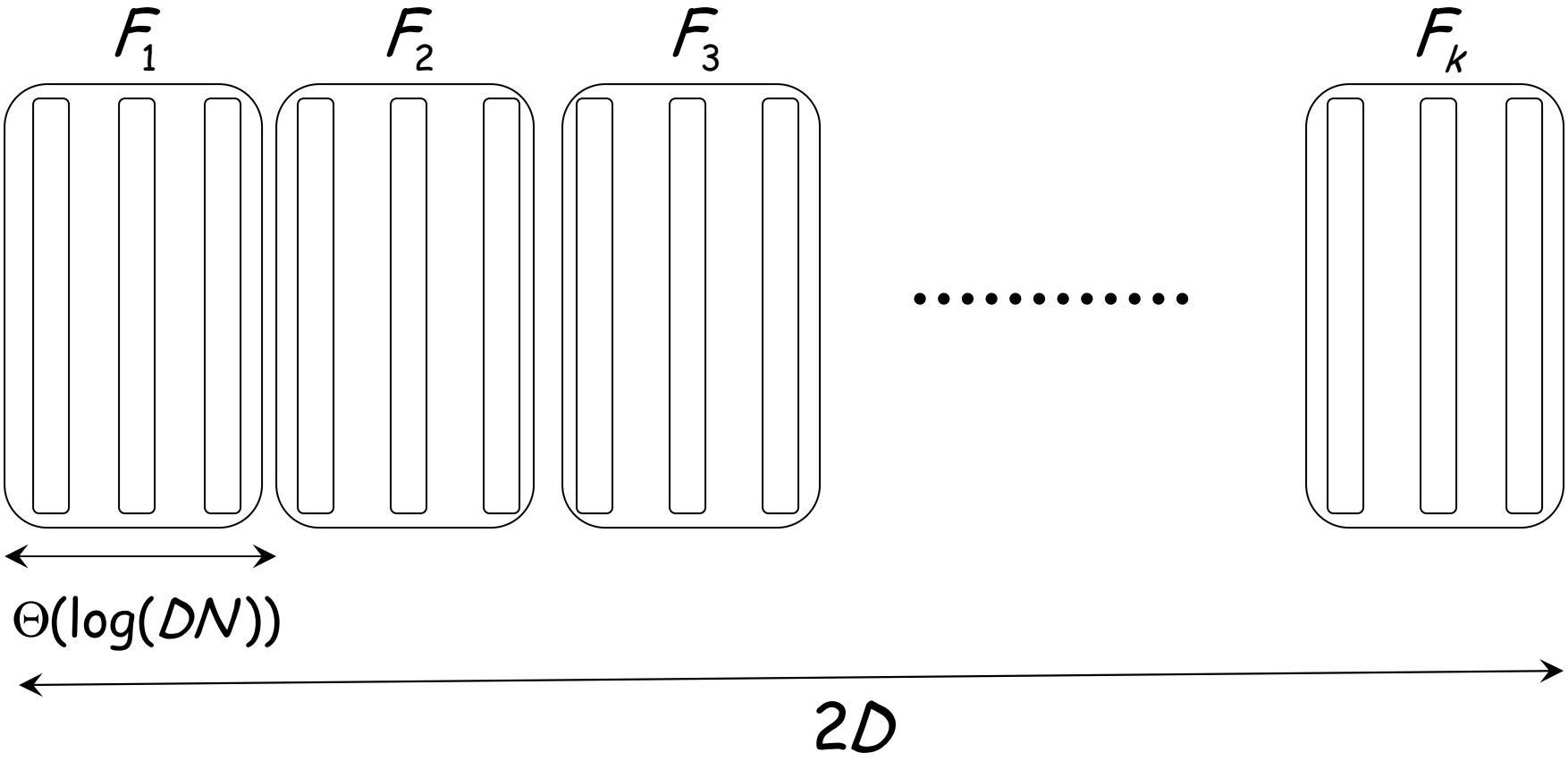
Expected one packet from each packet set



Expected Congestion 1

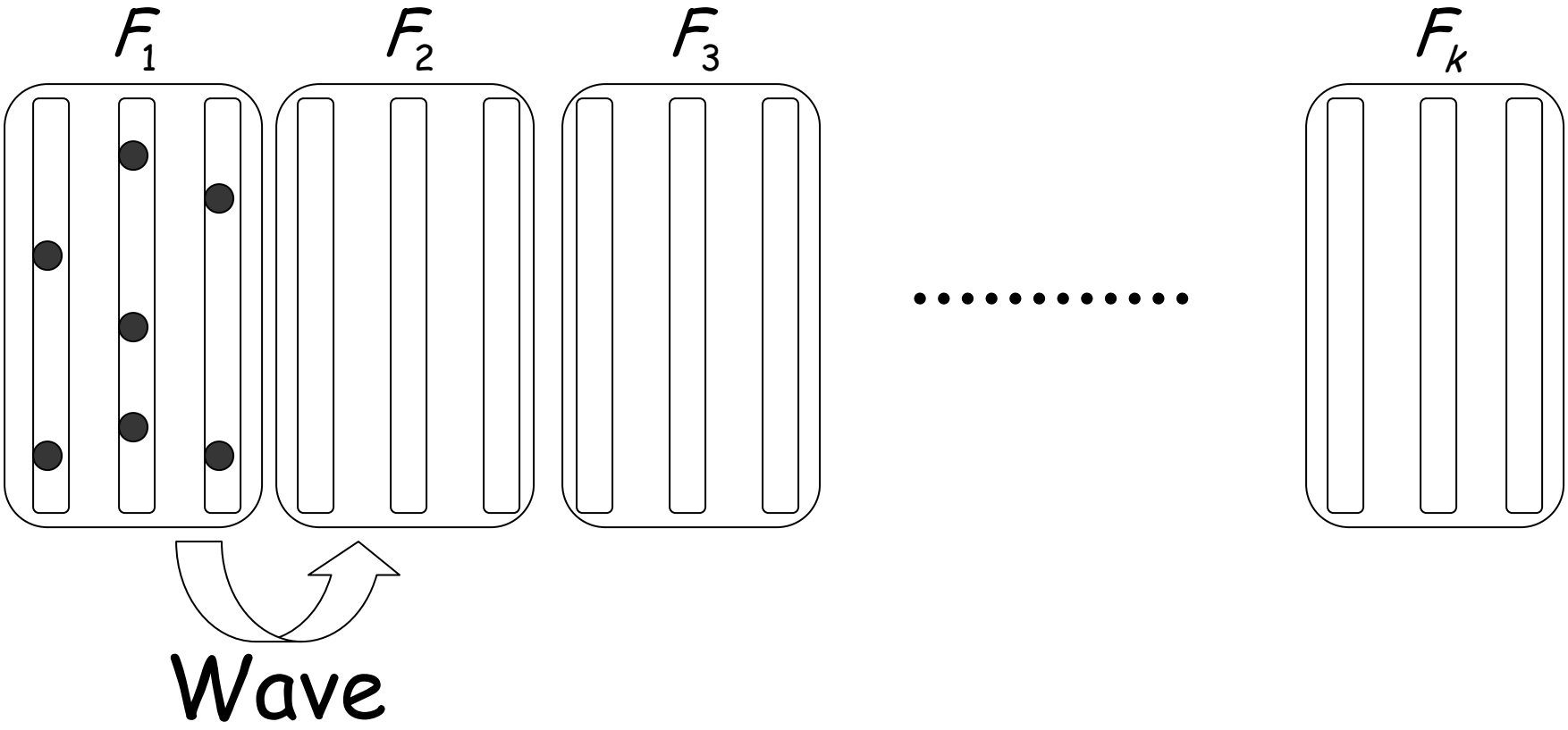
(Congestion $\log(DN)$ w.h.p.)

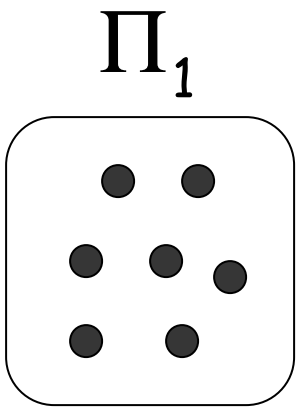
We partition the levels into **frames**



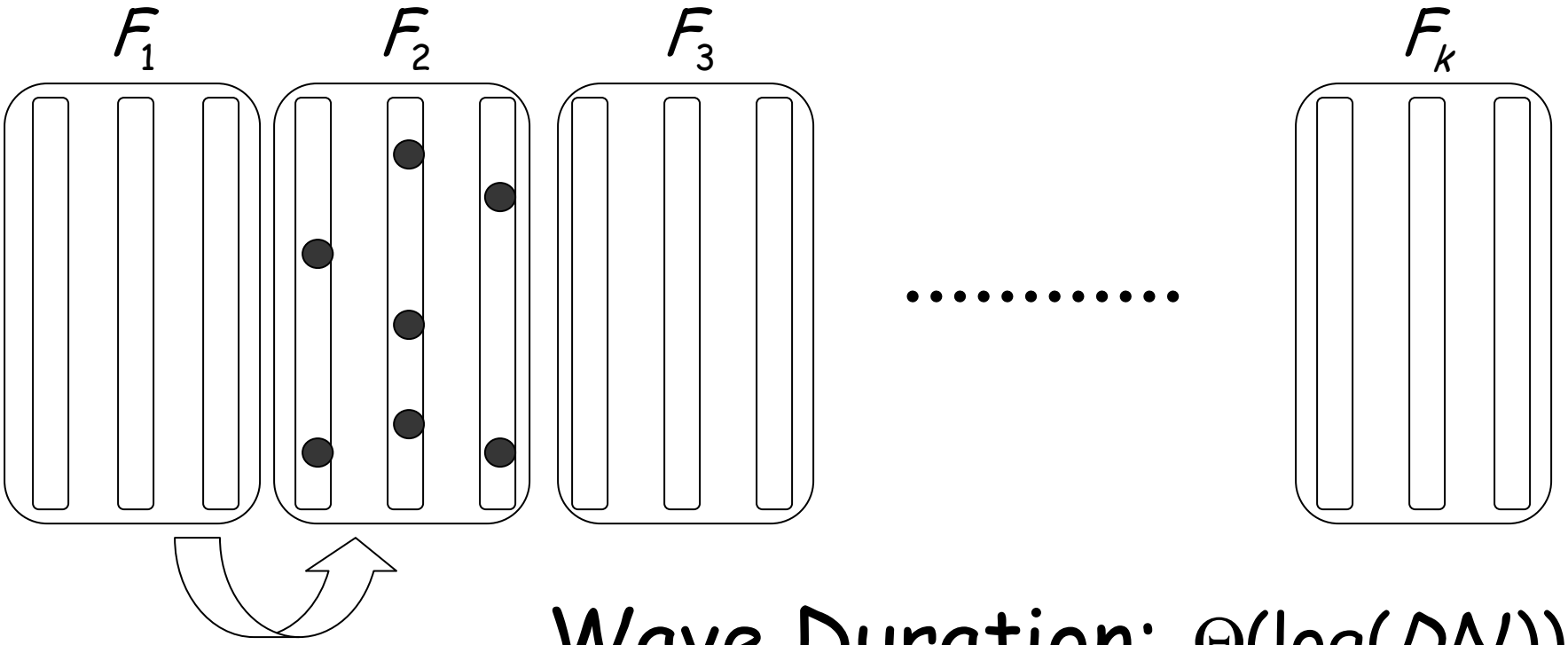
number of frames: $\Theta\left(\frac{D}{\log(DN)}\right)$

We send packets from frame to frame



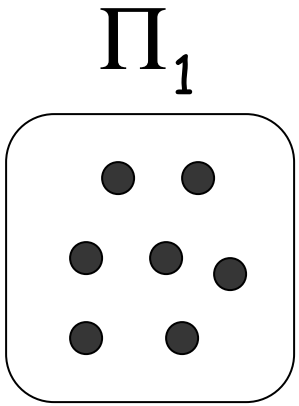


We send packets from frame to frame

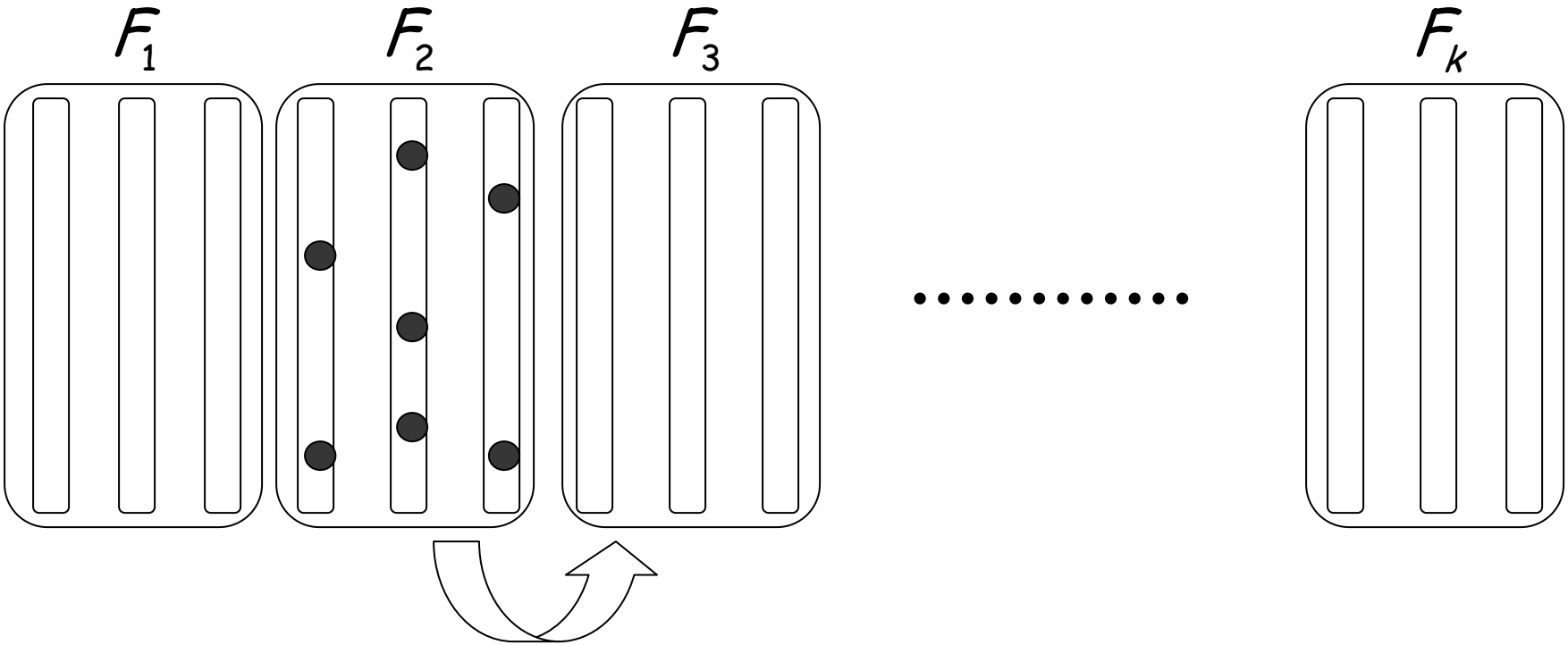


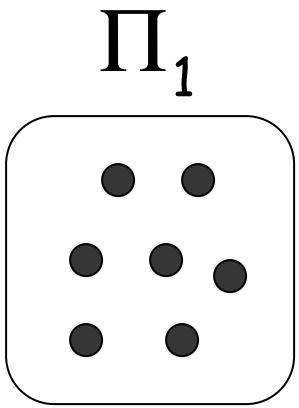
Wave

Wave Duration: $\Theta(\log(DN))$

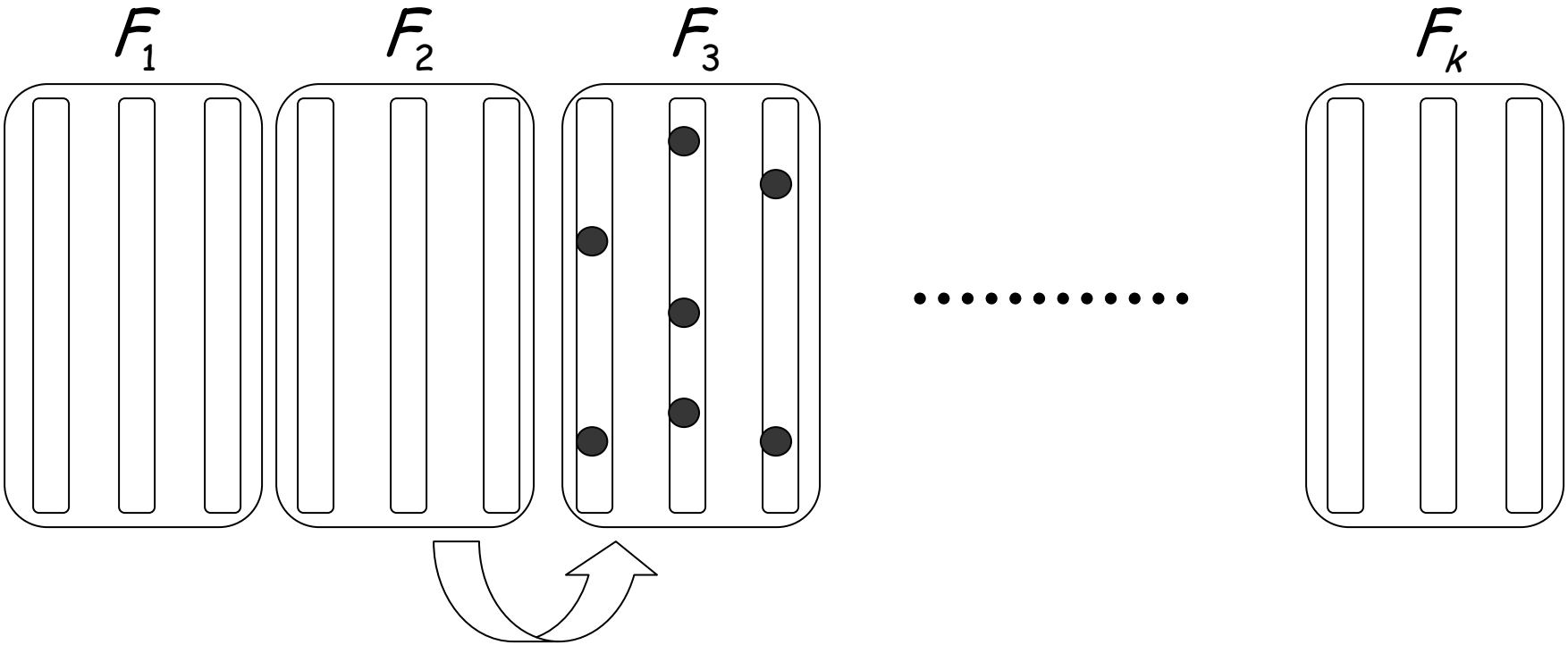


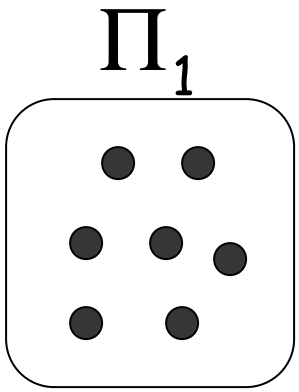
We send packets from frame to frame



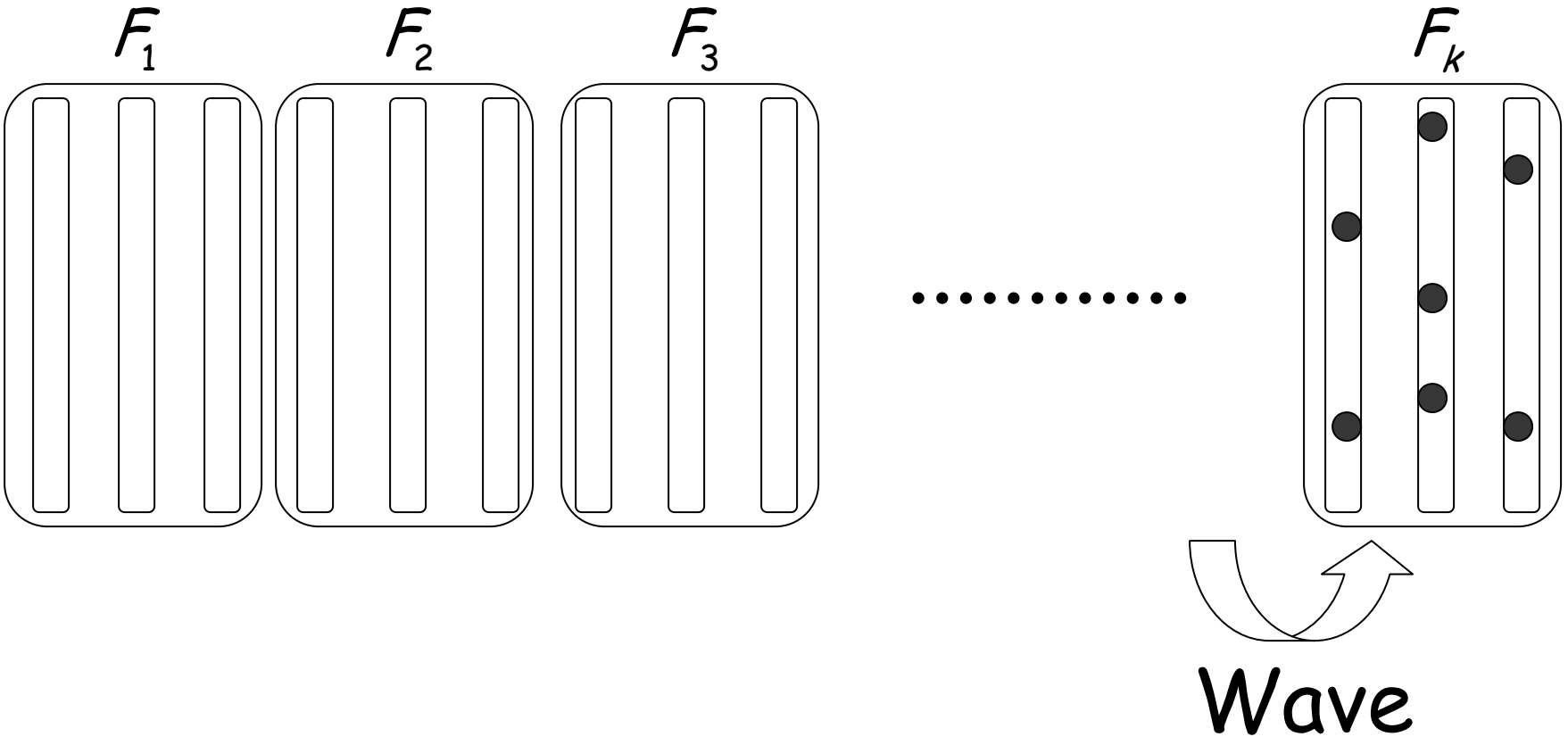


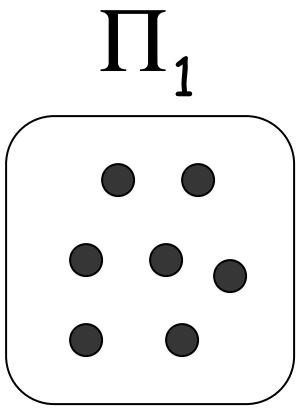
We send packets from frame to frame

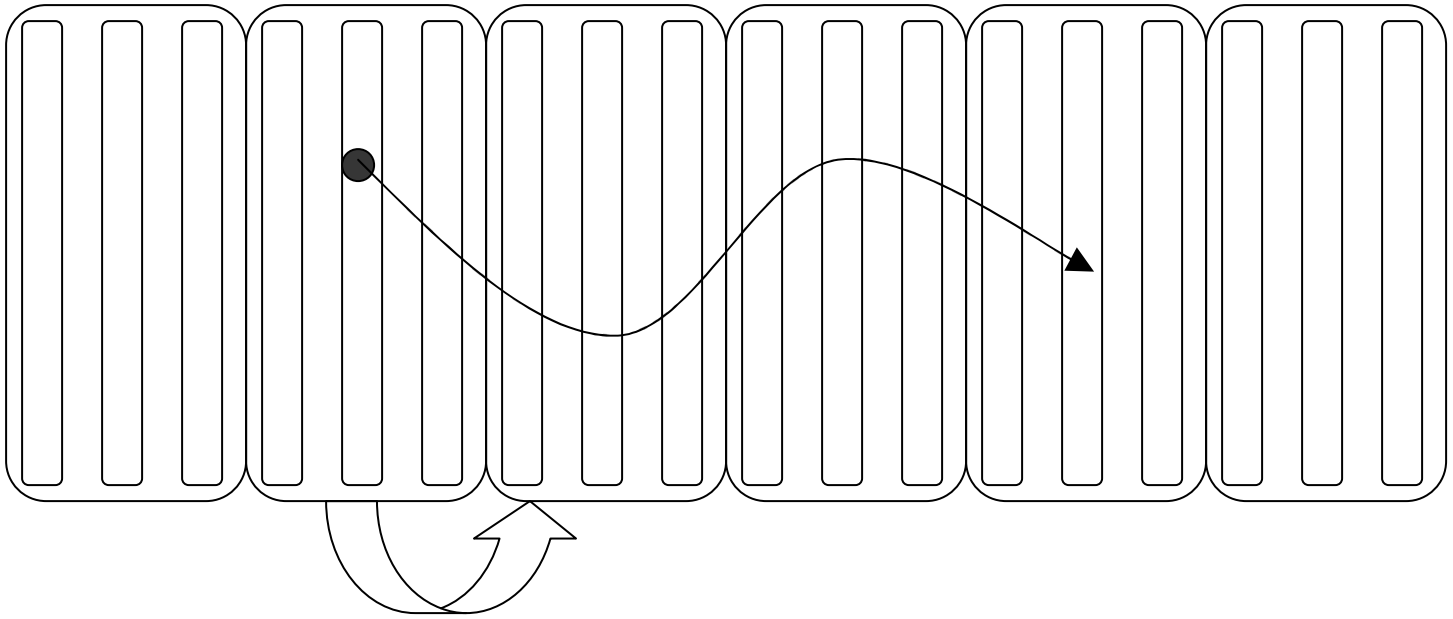




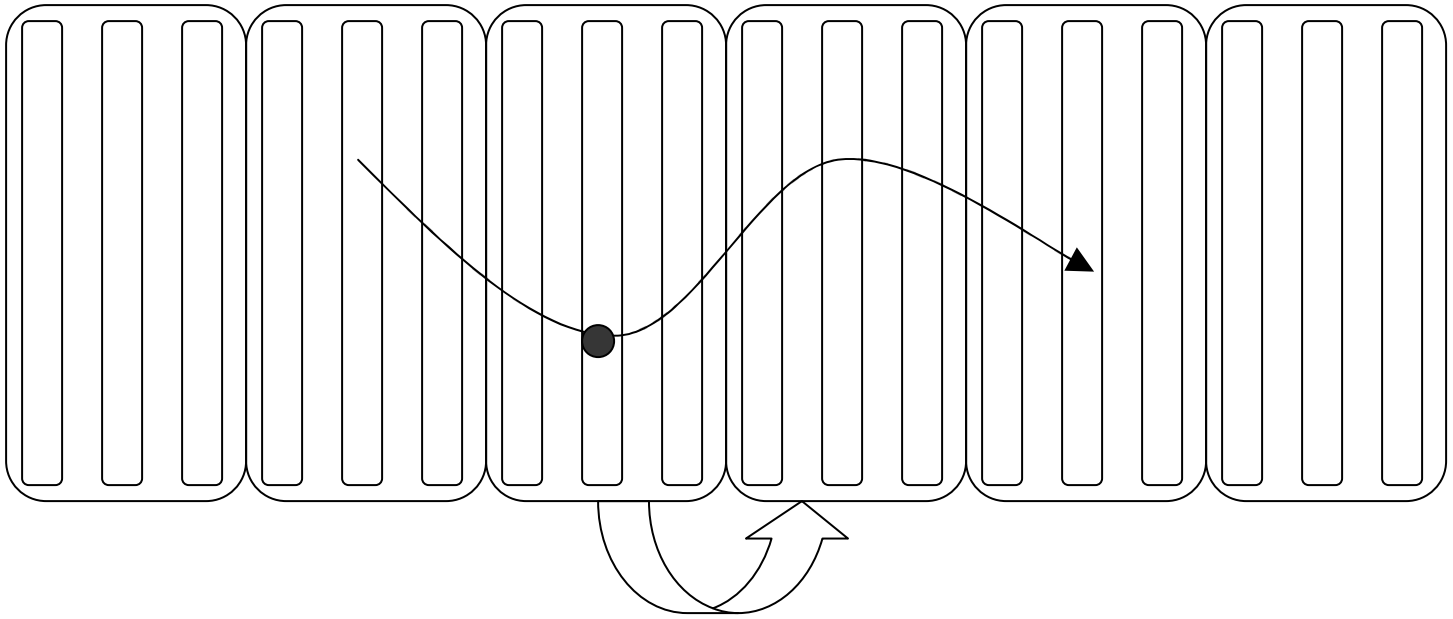
We send packets from frame to frame



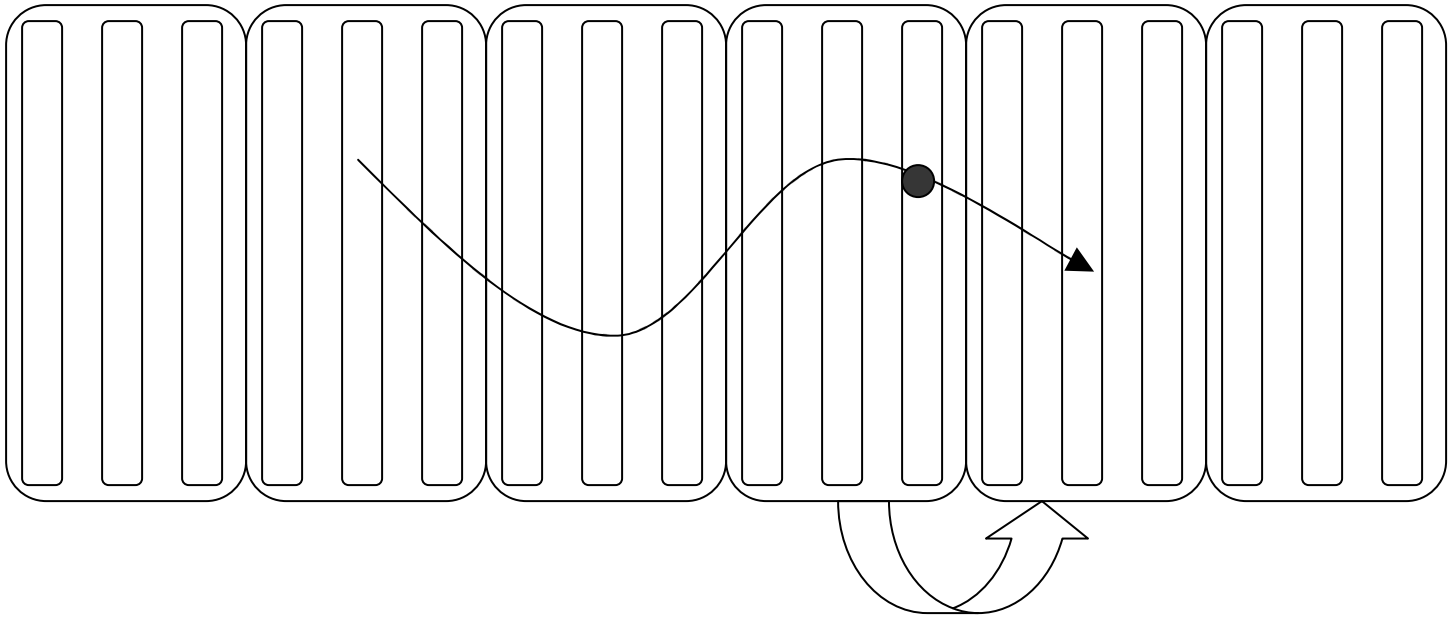




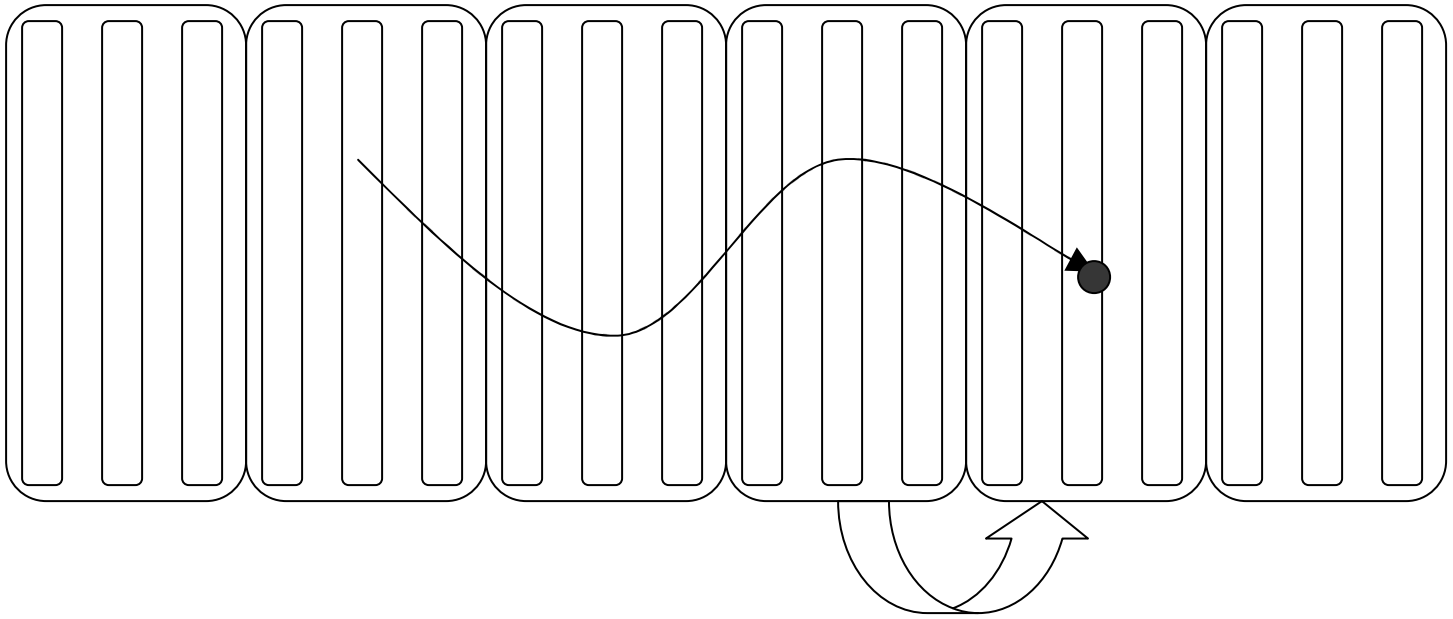
A packet follows its path from source to Destination along the wave



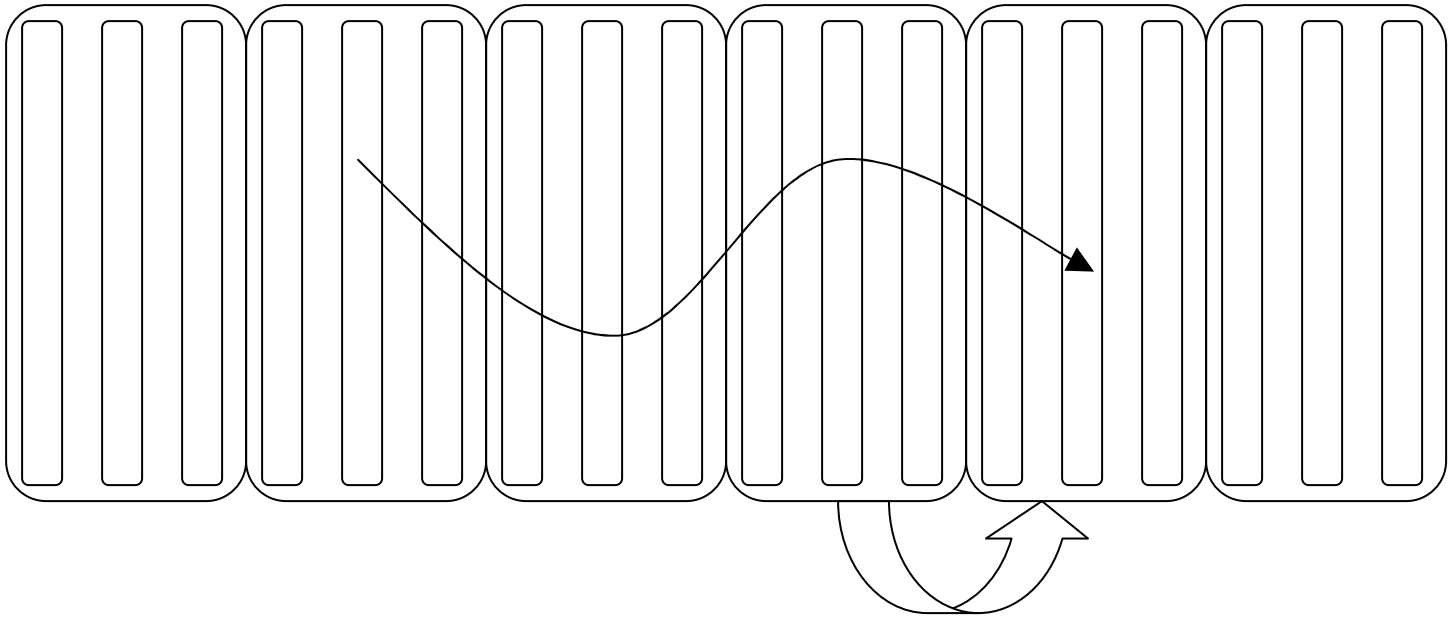
A packet follows its path from source to Destination along the wave



A packet follows its path from source to Destination along the wave

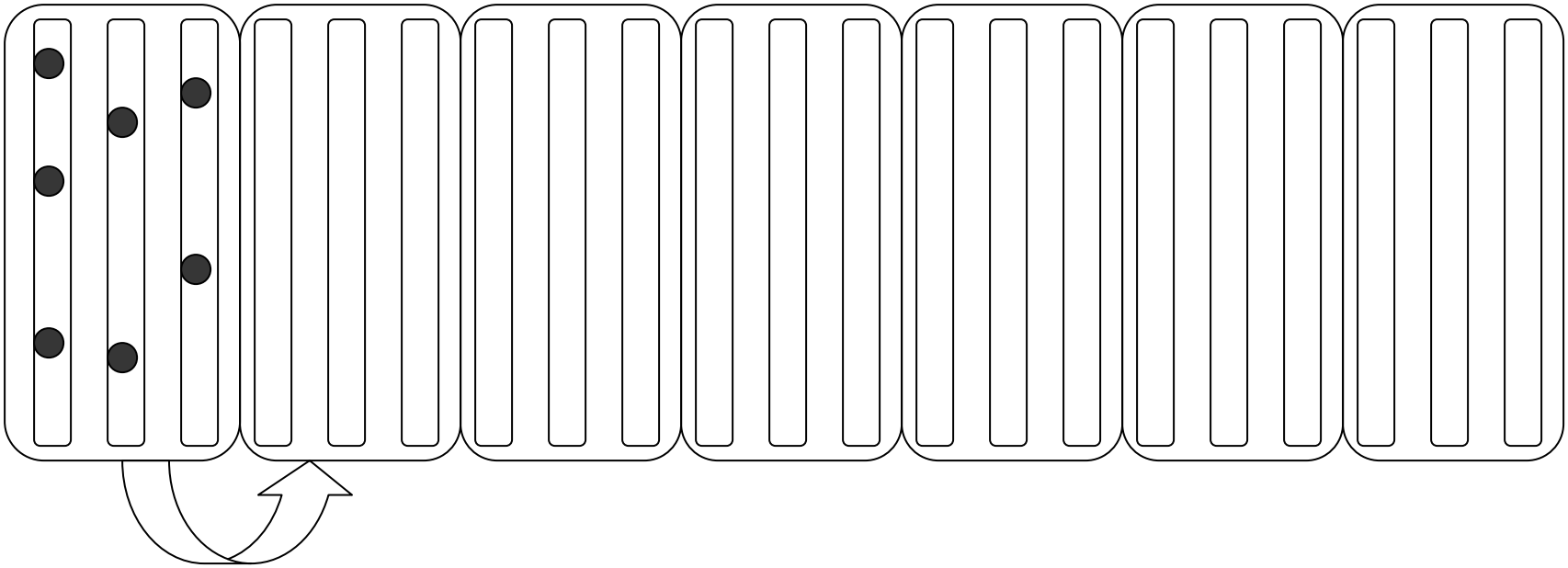


A packet follows its path from source to Destination along the wave

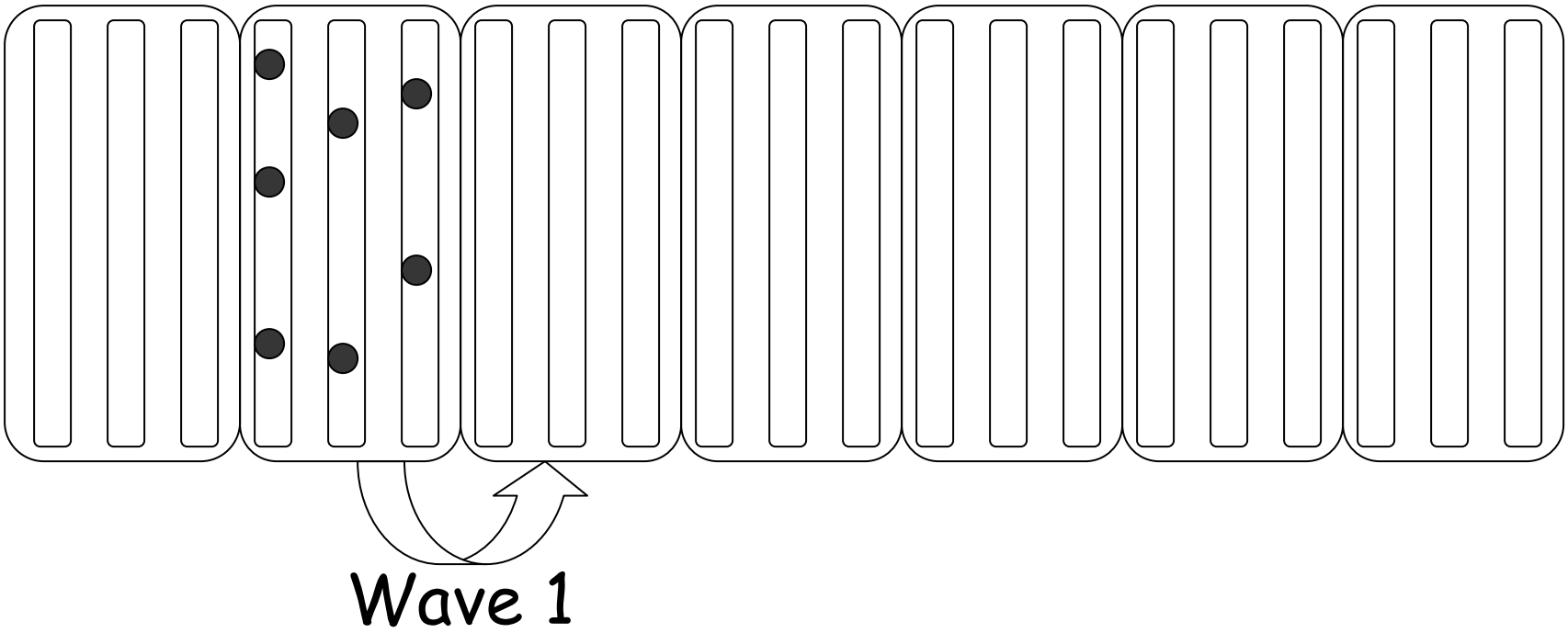


A packet follows its path from source to Destination along the wave

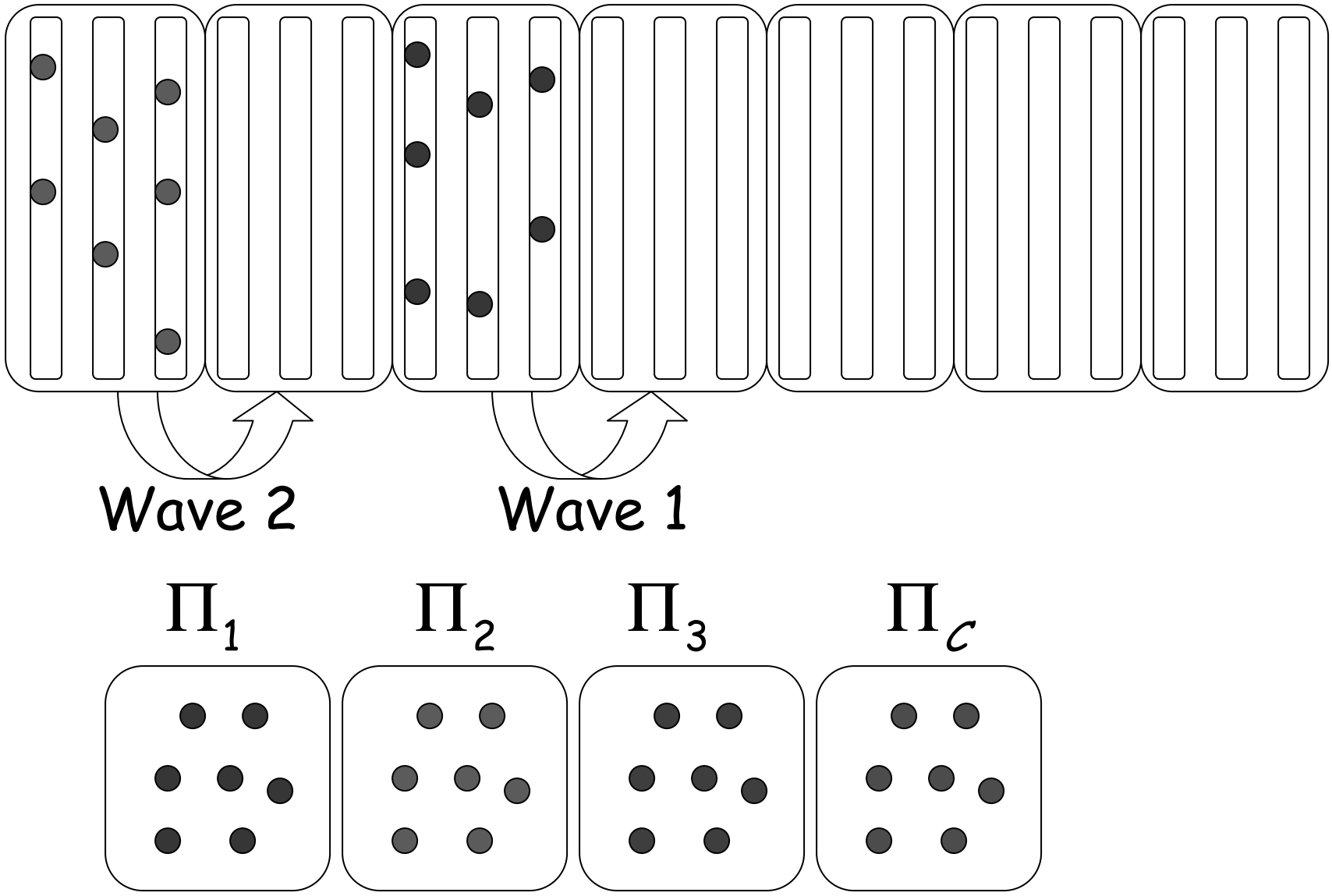
Sending packets of different packet sets simultaneously



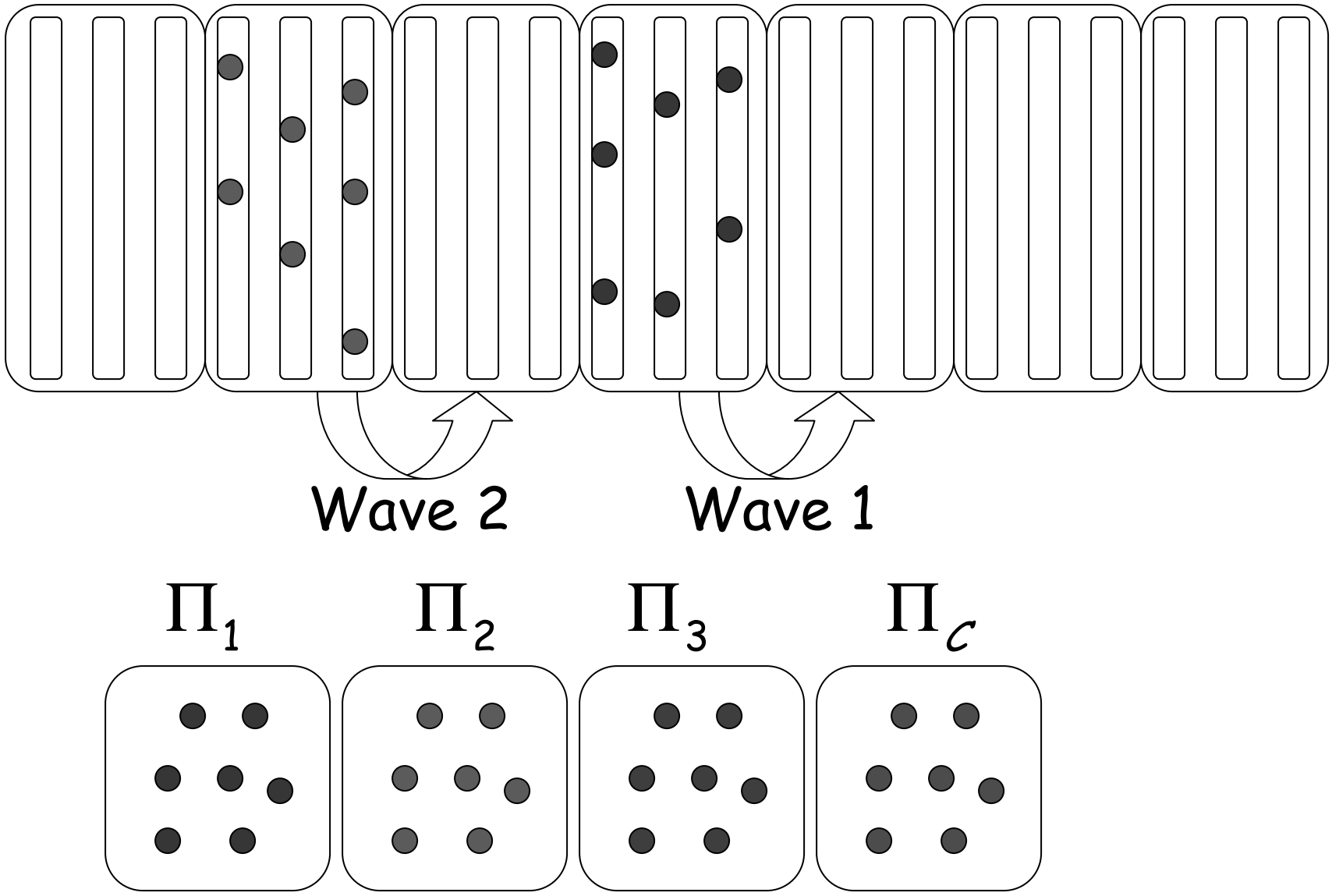
Sending packets of different packet sets simultaneously



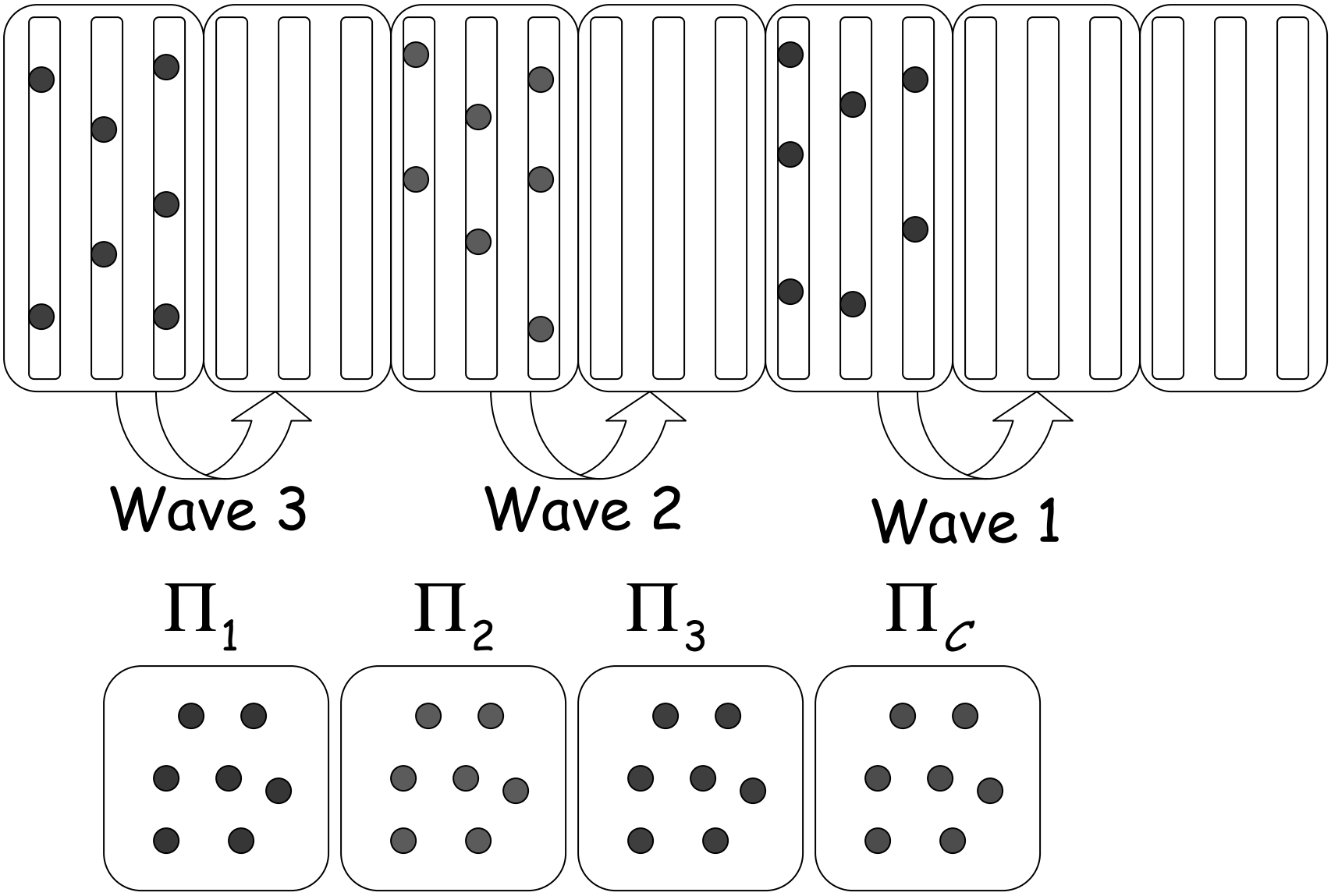
Sending packets of different packet sets simultaneously



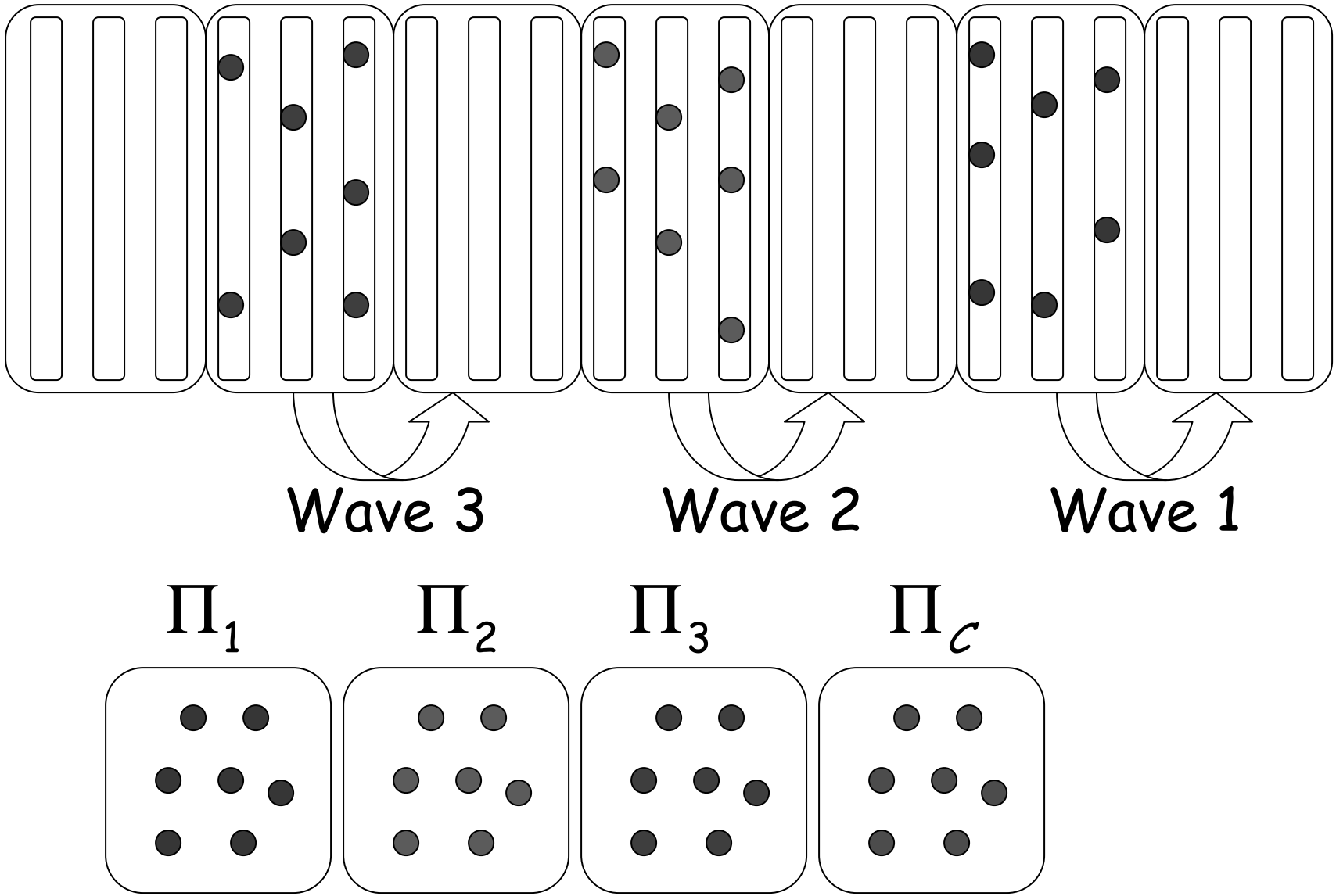
Sending packets of different packet sets simultaneously



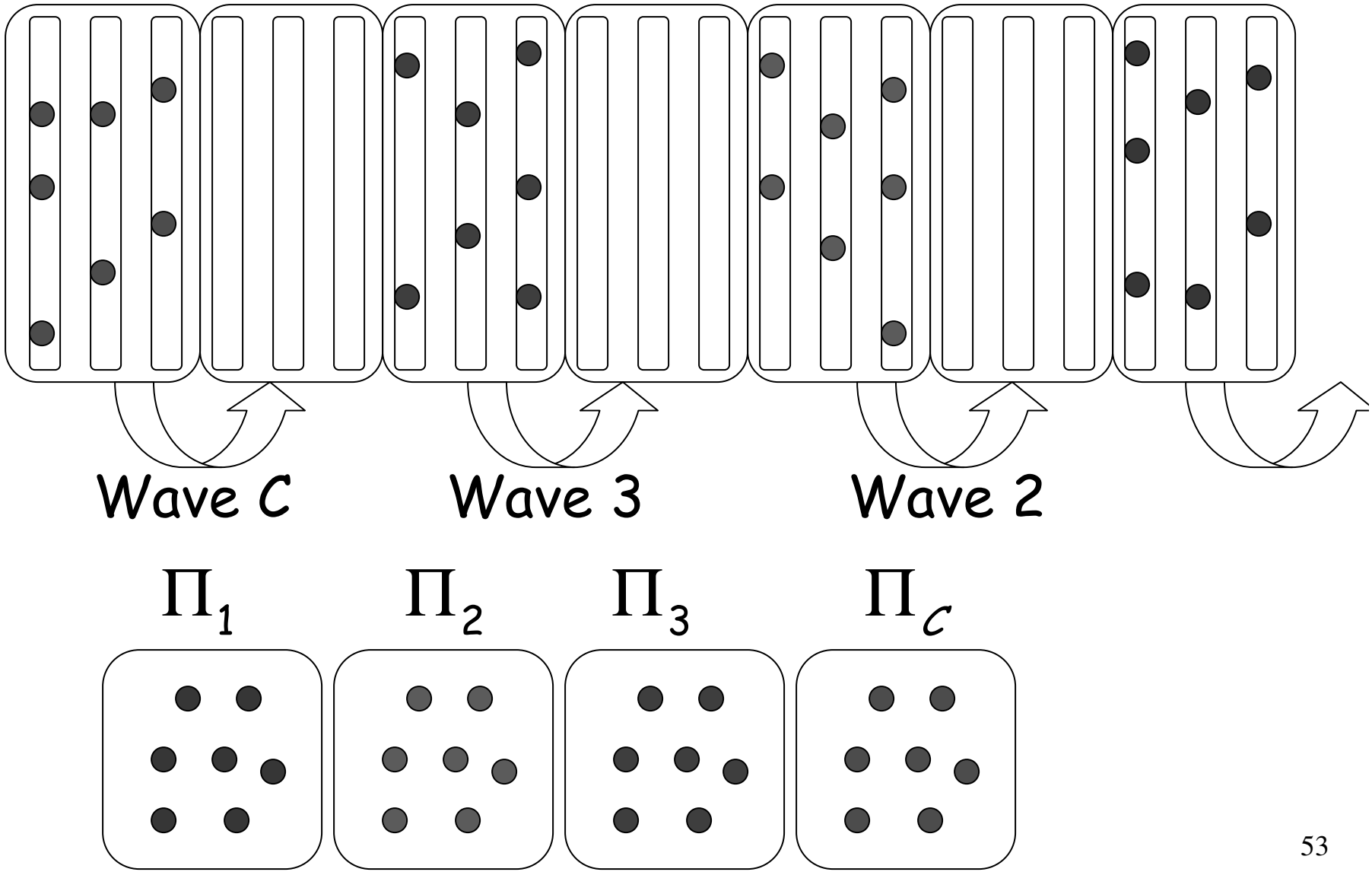
Sending packets of different packet sets simultaneously



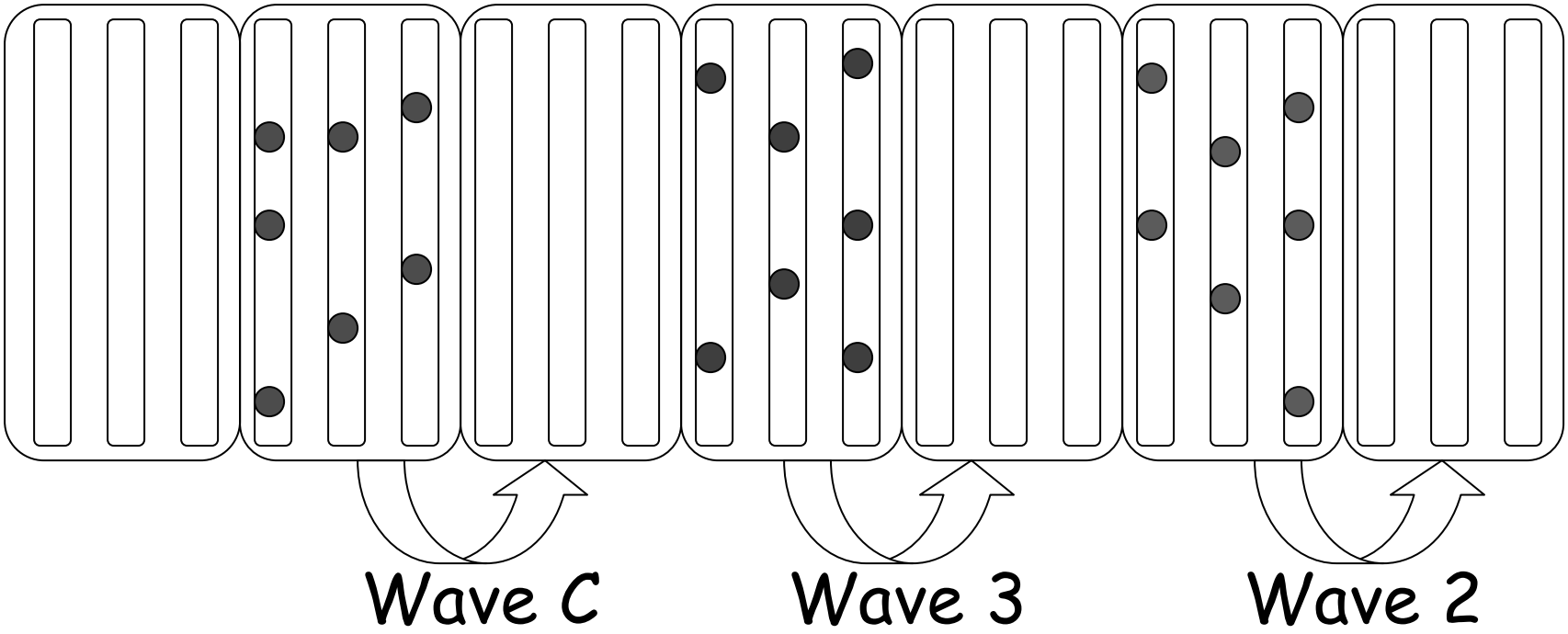
Sending packets of different packet sets simultaneously

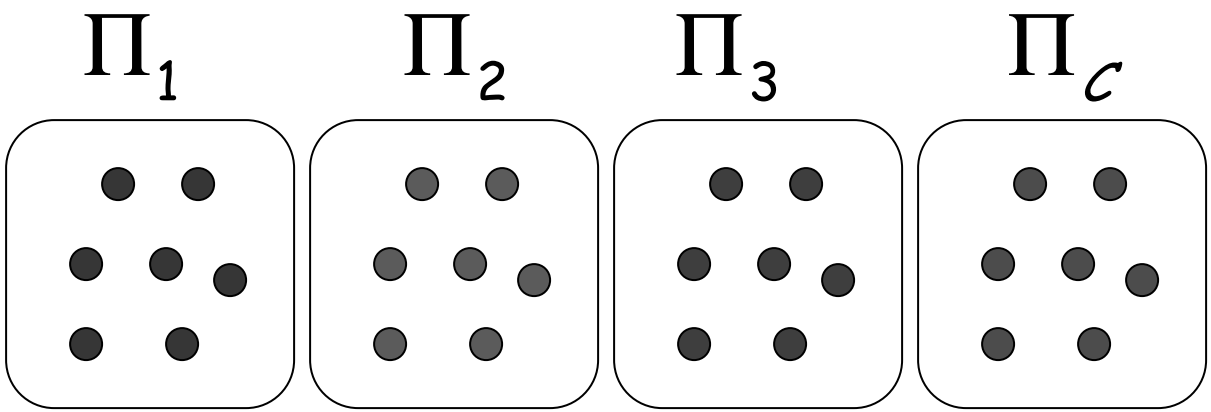


Sending packets of different packet sets simultaneously

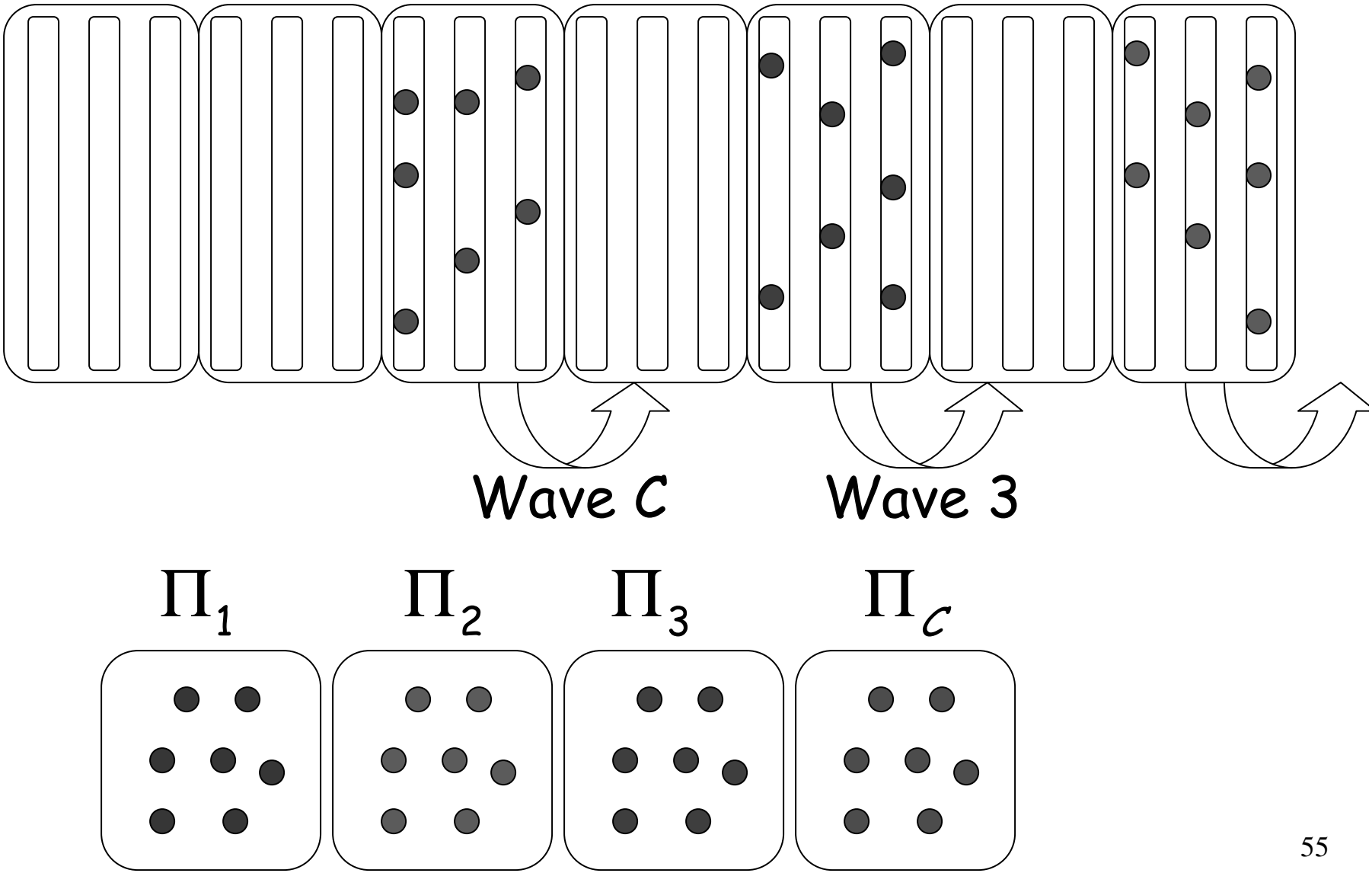


Sending packets of different packet sets simultaneously

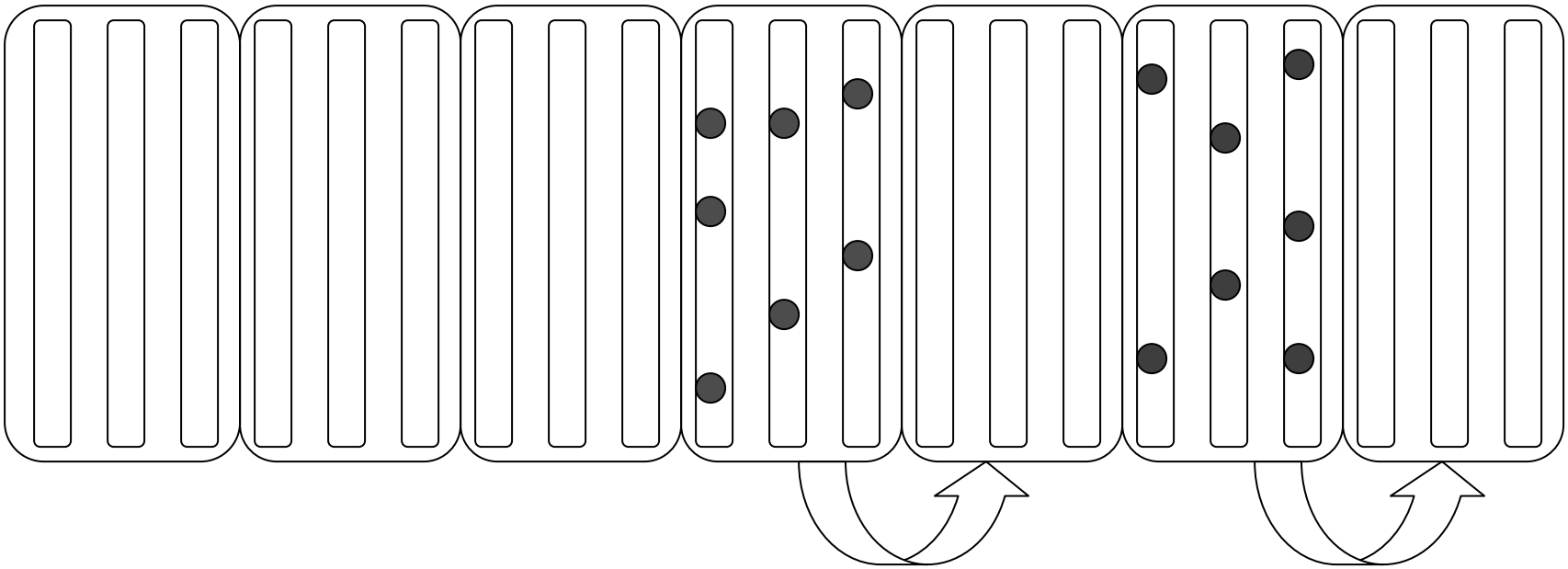


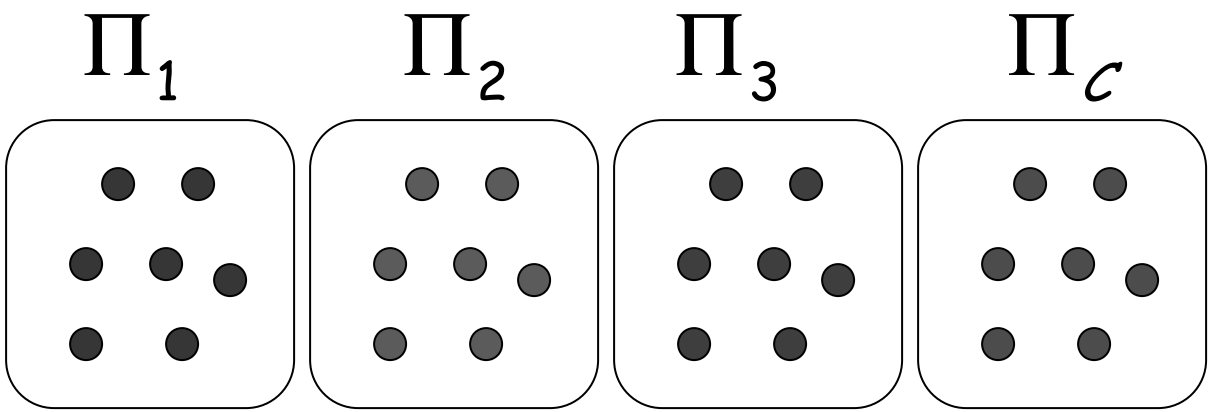


Sending packets of different packet sets simultaneously

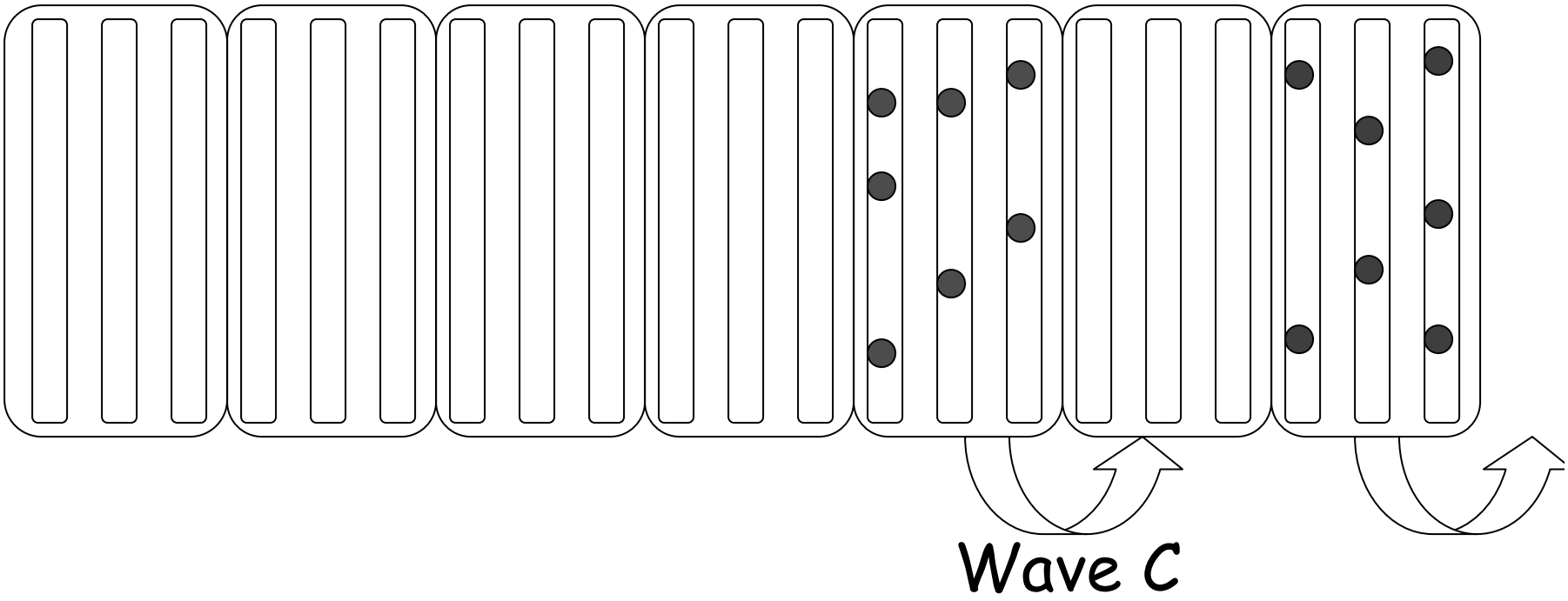


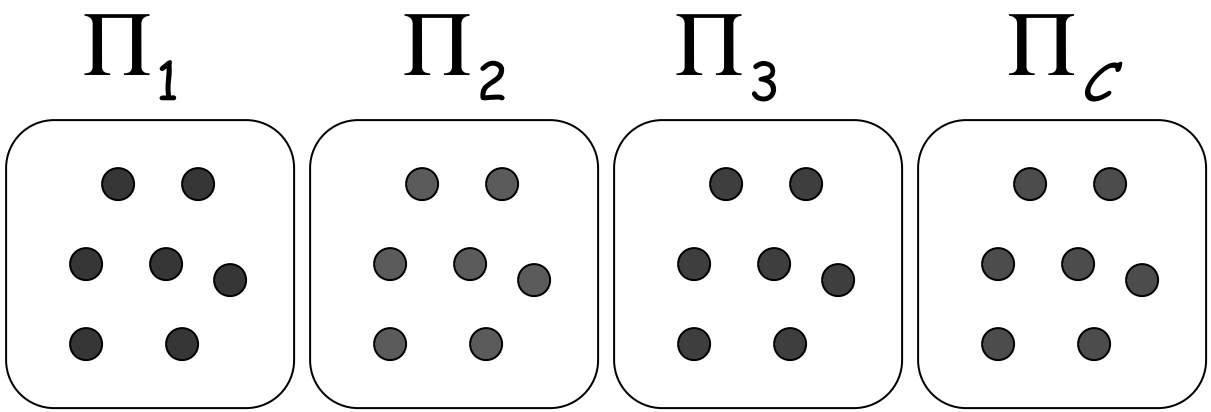
Sending packets of different packet sets simultaneously



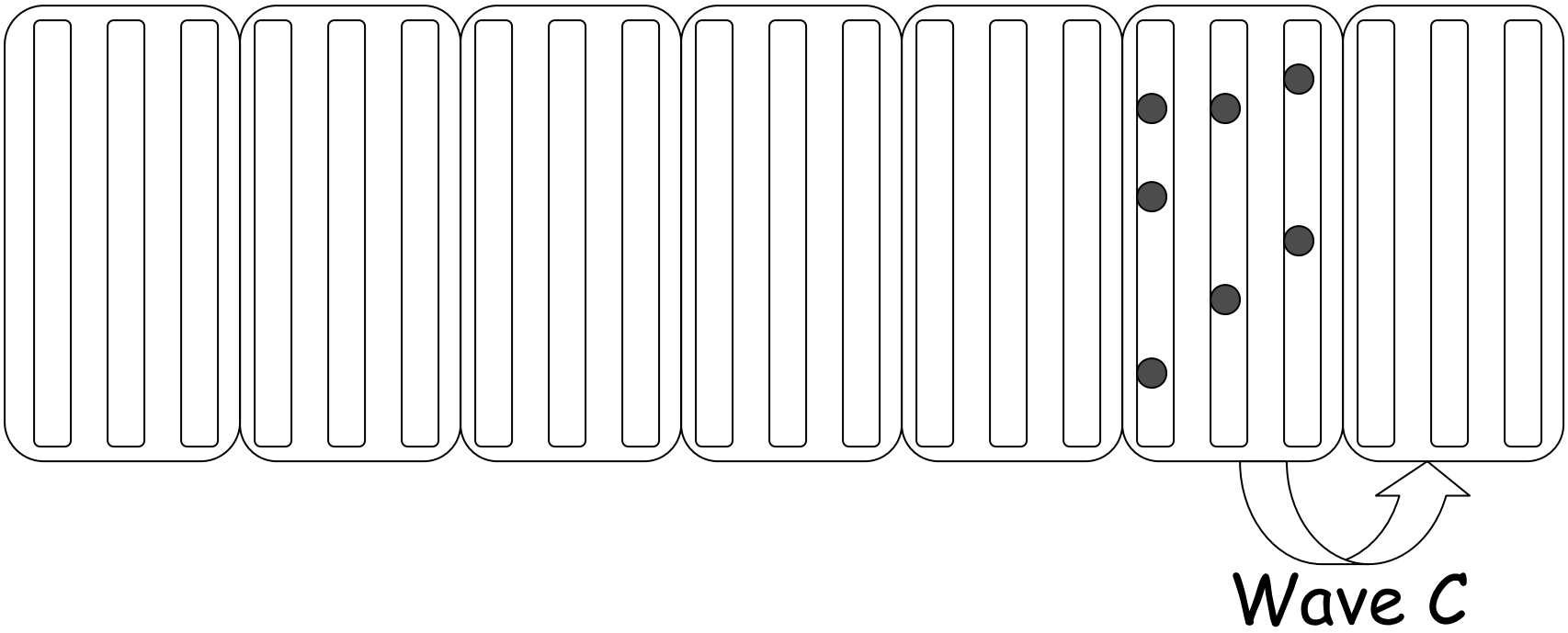


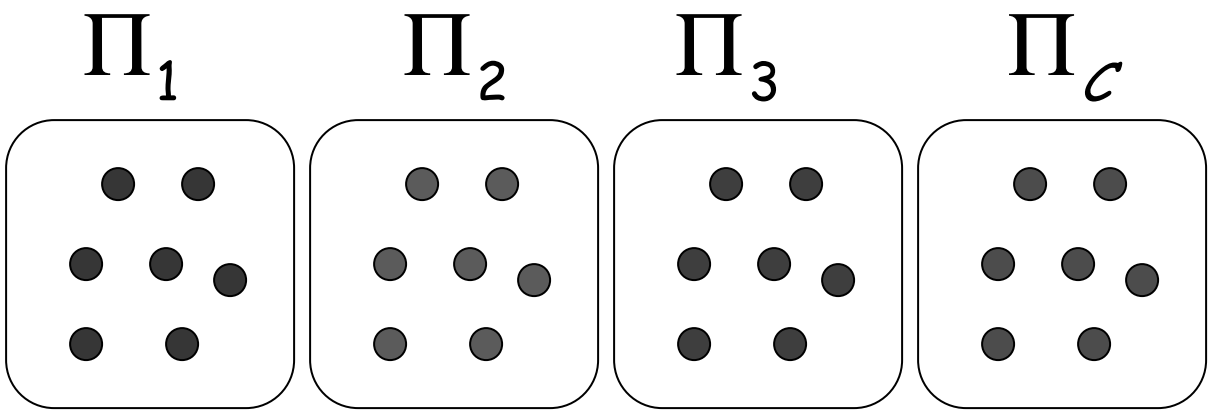
Sending packets of different packet sets simultaneously



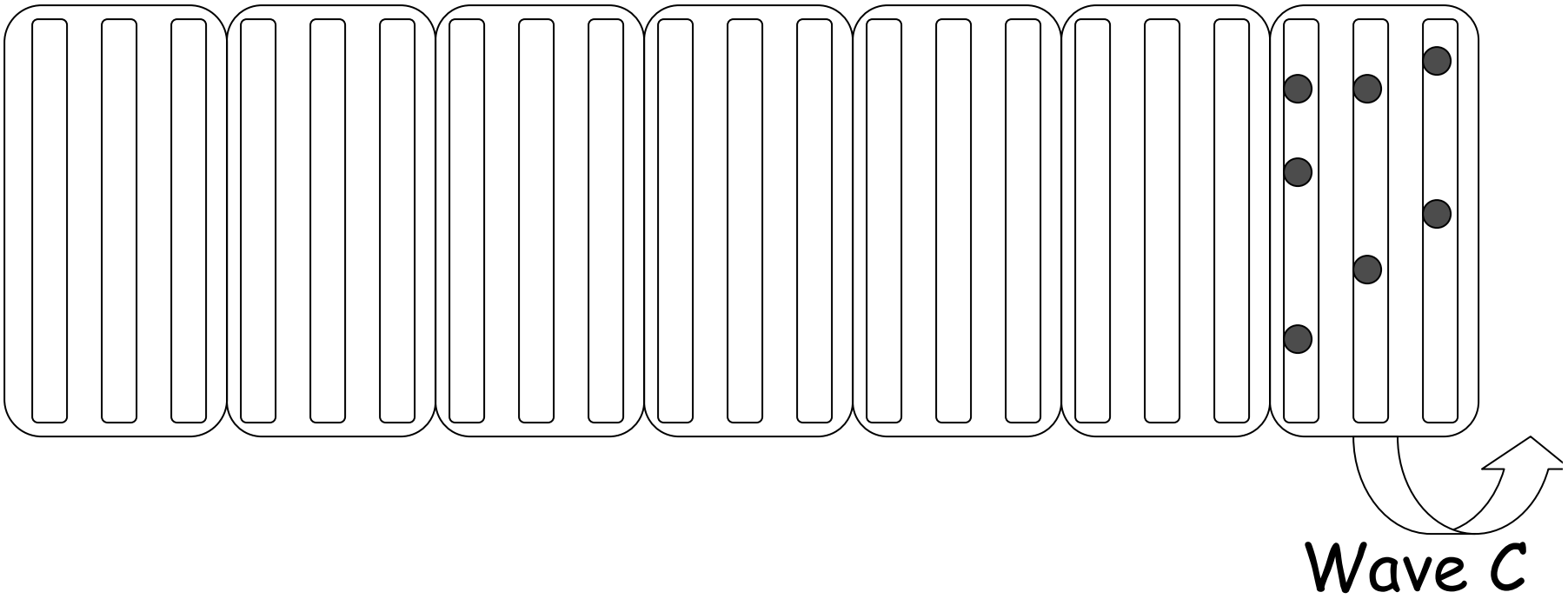


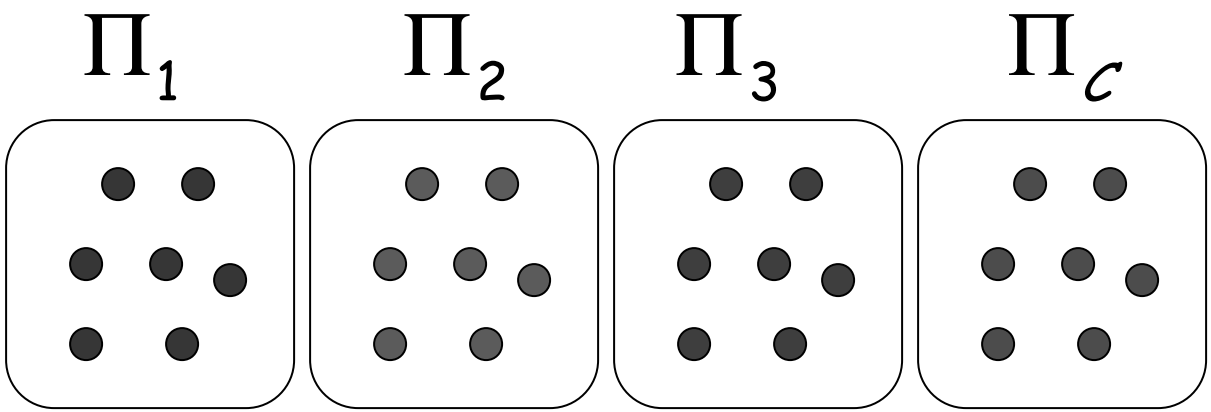
Sending packets of different packet sets simultaneously



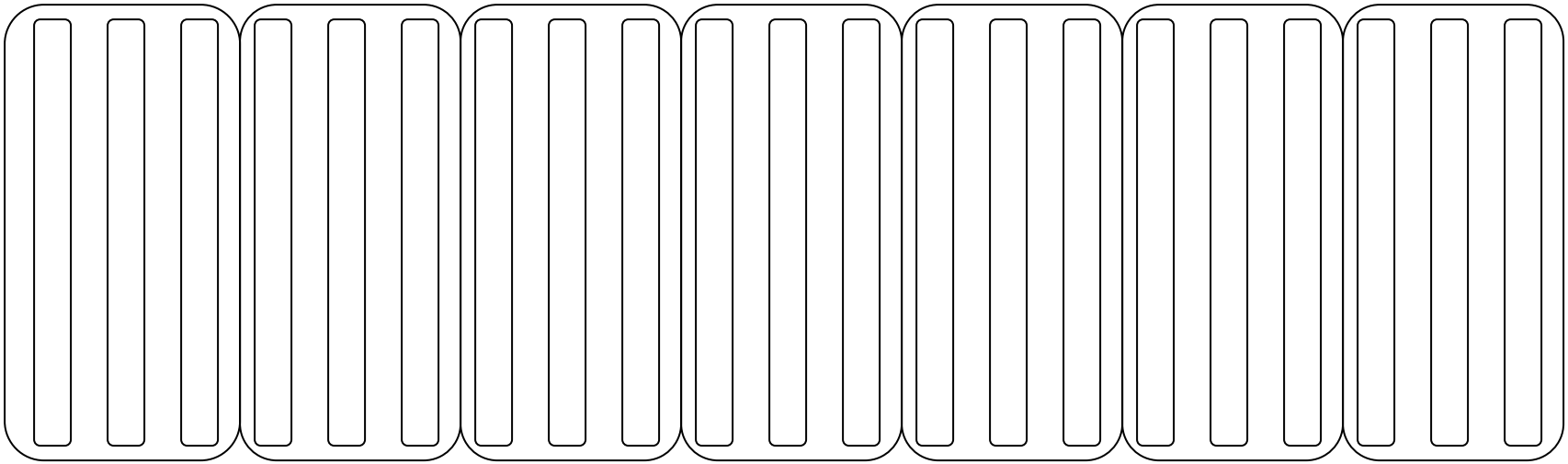


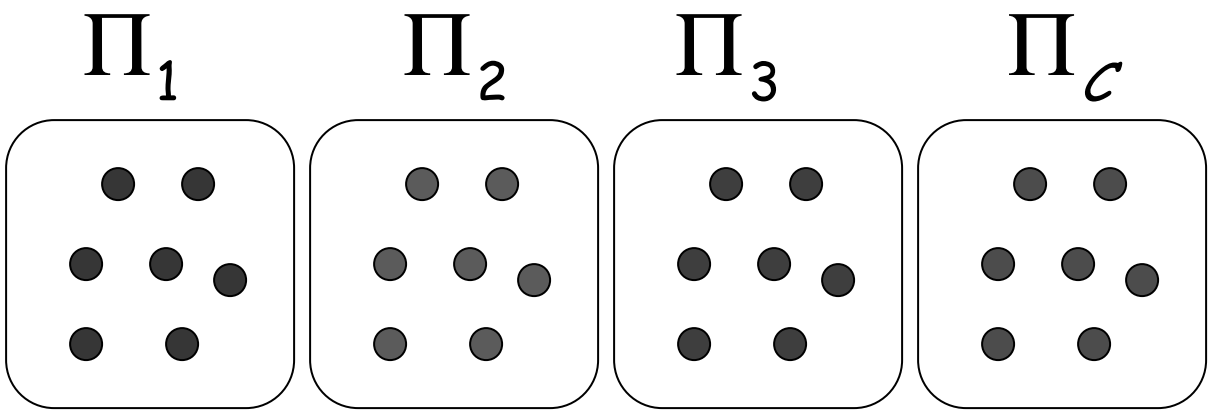
Sending packets of different packet sets simultaneously





All packets have been absorbed!





Routing Time = Time until last wave C leaves the network

Wave duration: $2C\log(DN)$ + #frames: $\frac{D}{\log(DN)}$ Wave duration: $\log(DN) = \Theta(C\log(DN) + D)$

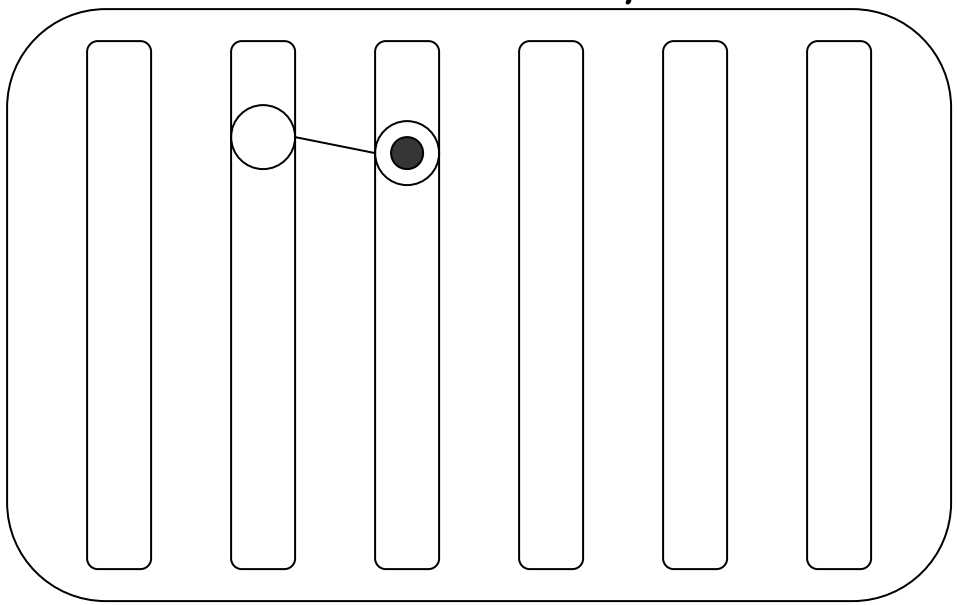
Time when wave C enters the network

Time that wave C needs to traverse the network

Oscilation

Simulates buffering

Frame F_i

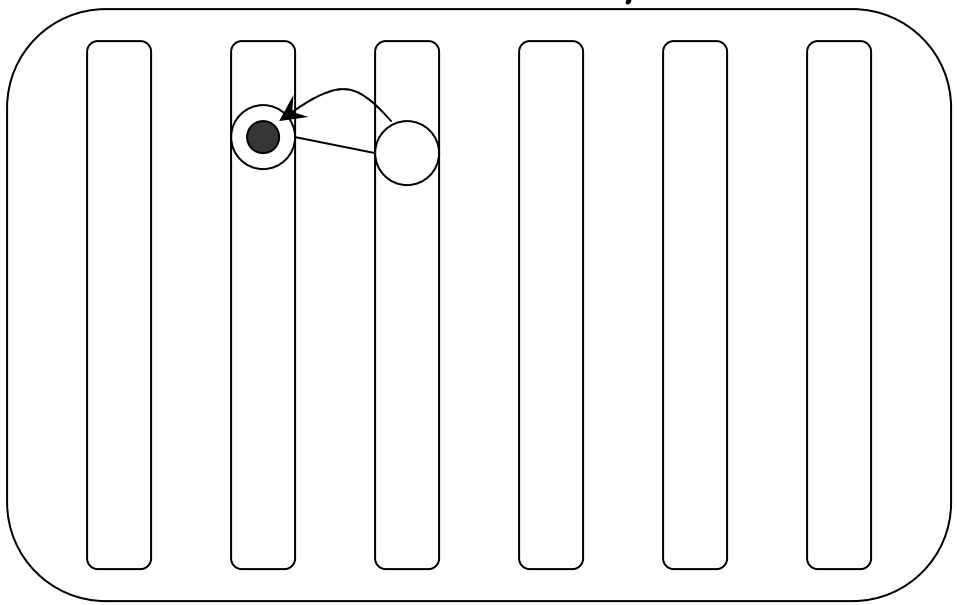


Time t

Oscilation

Simulates buffering

Frame F_i

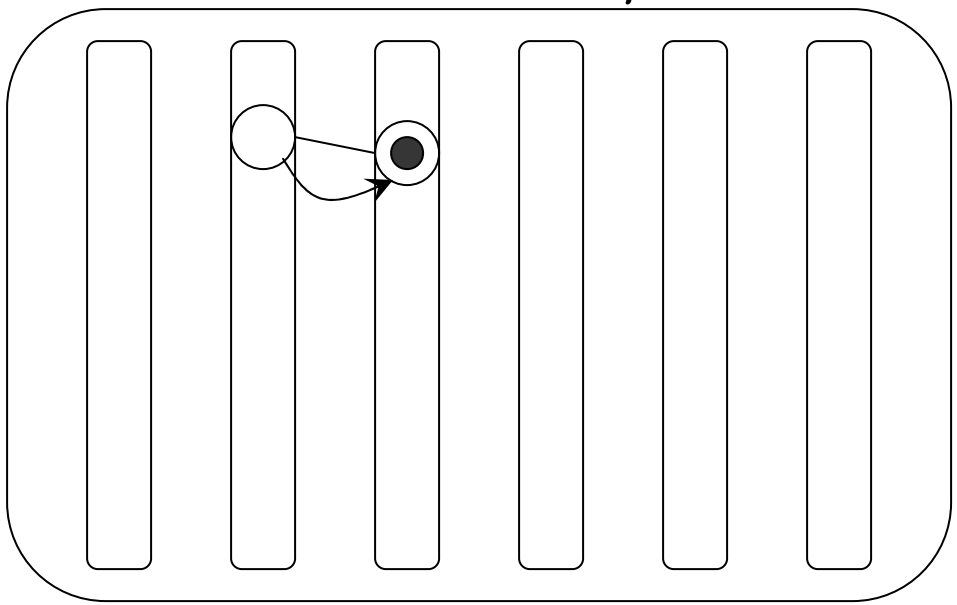


Time $t+1$

Oscilation

Simulates buffering

Frame F_i

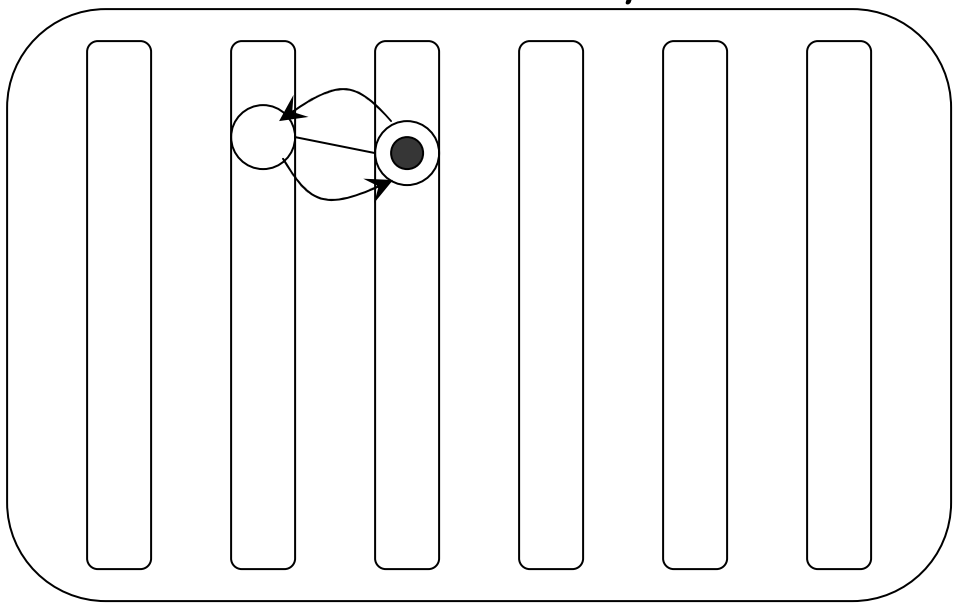


Time $t+2$

Oscilation

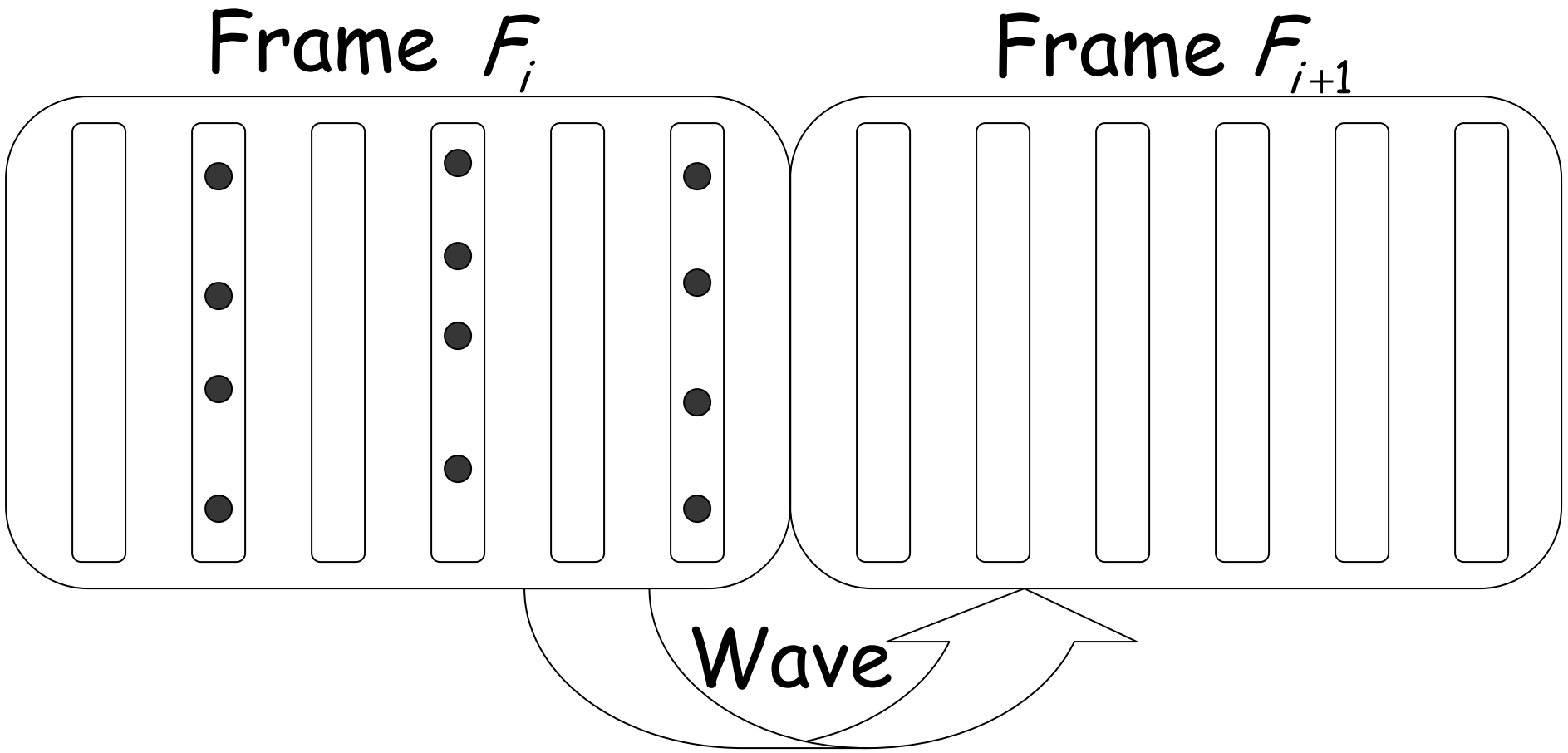
Simulates buffering

Frame F_i

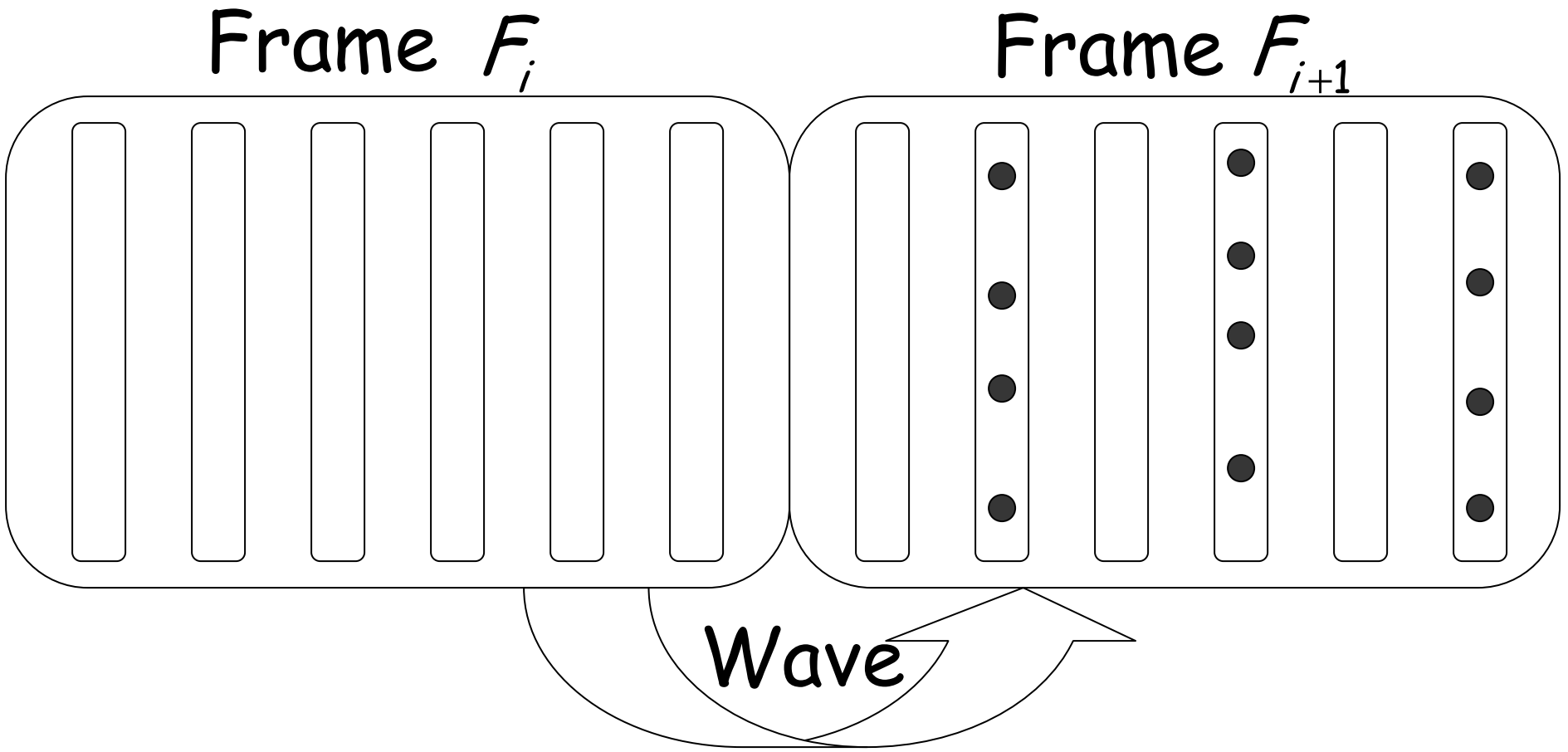


Time $t+3, t+4, \ldots$

Packet propagation during a wave

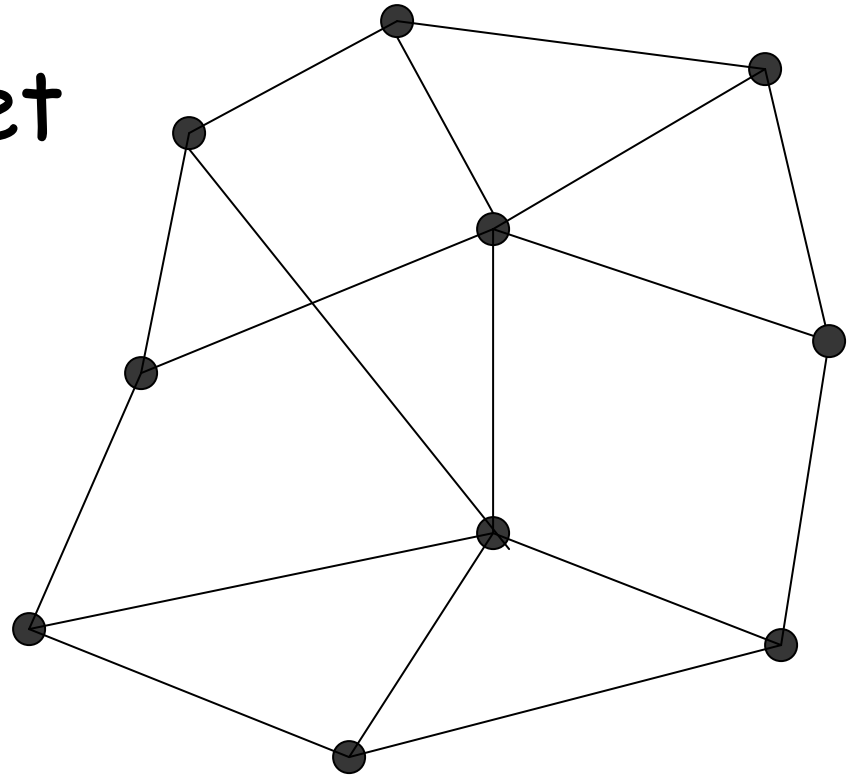


Packet propagation during a wave



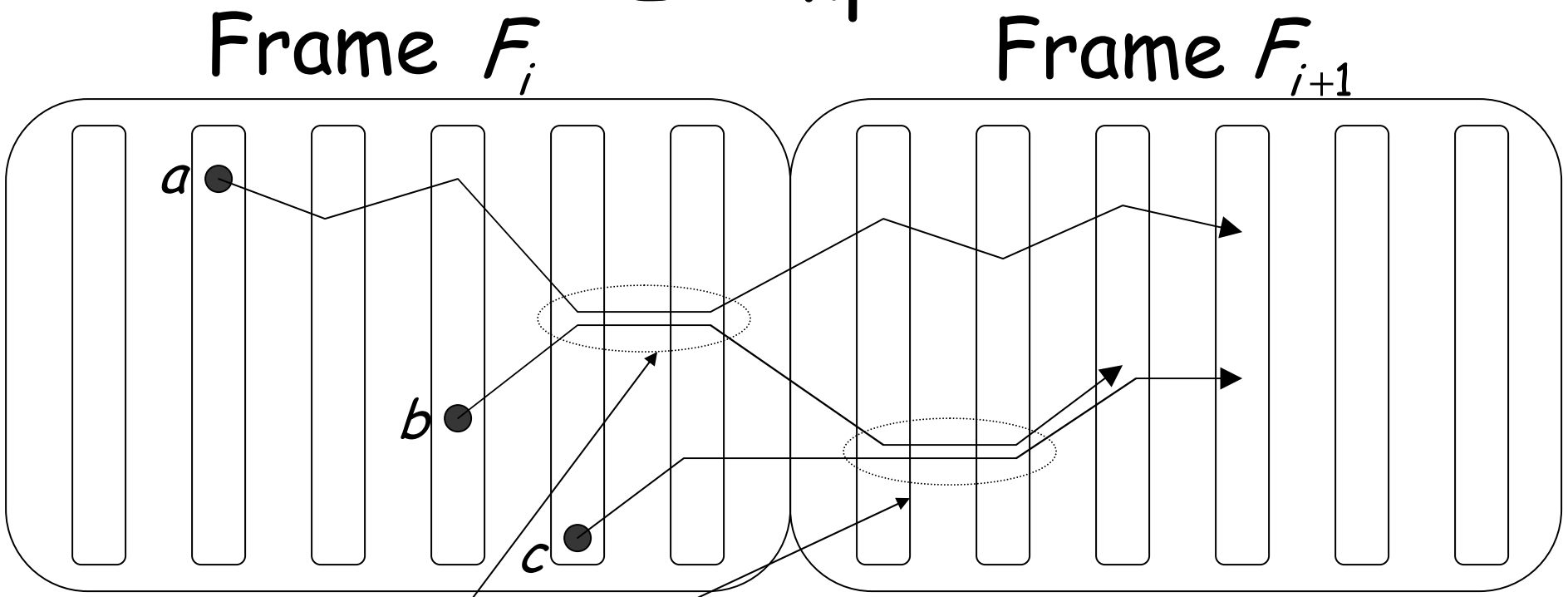
Conflict Graph

Each node is a packet



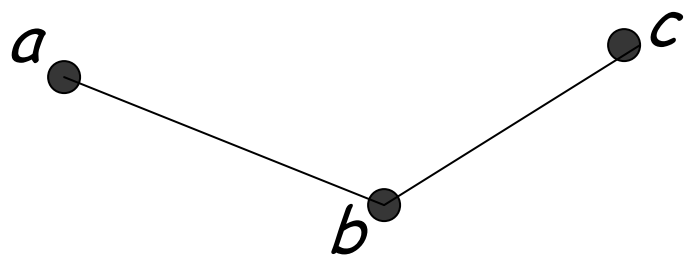
Two packets are adjacent
if their paths use a common edge in the frames F_i and F_{i+1}

Example



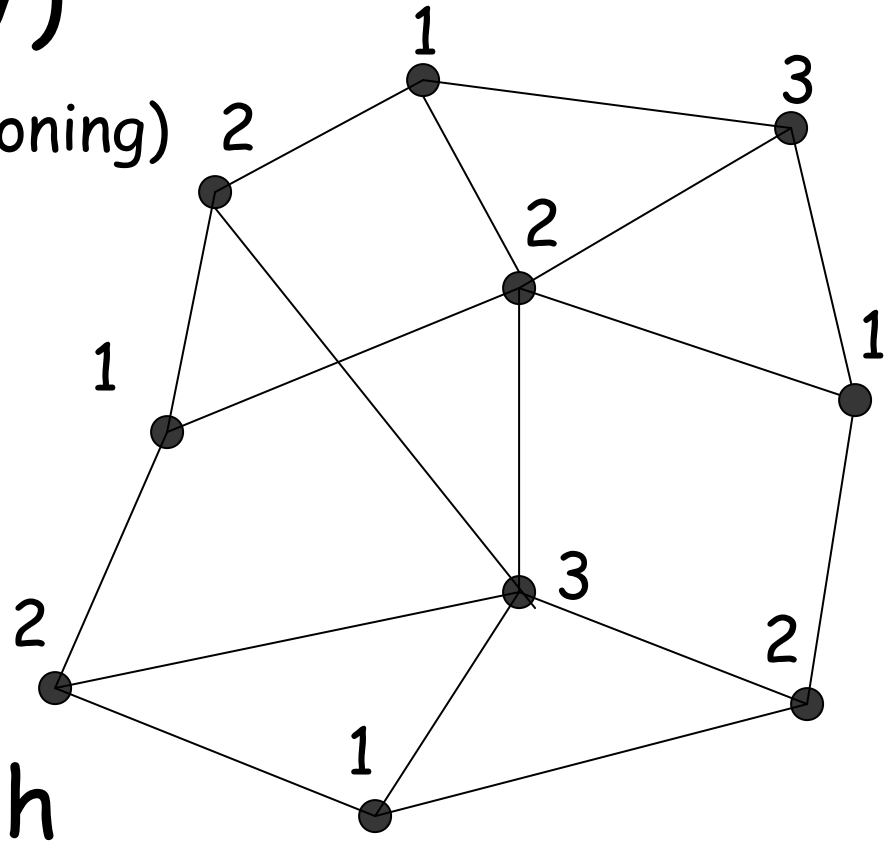
Share edge

Conflict graph



The degree of any node is bounded by $\log(DN)$

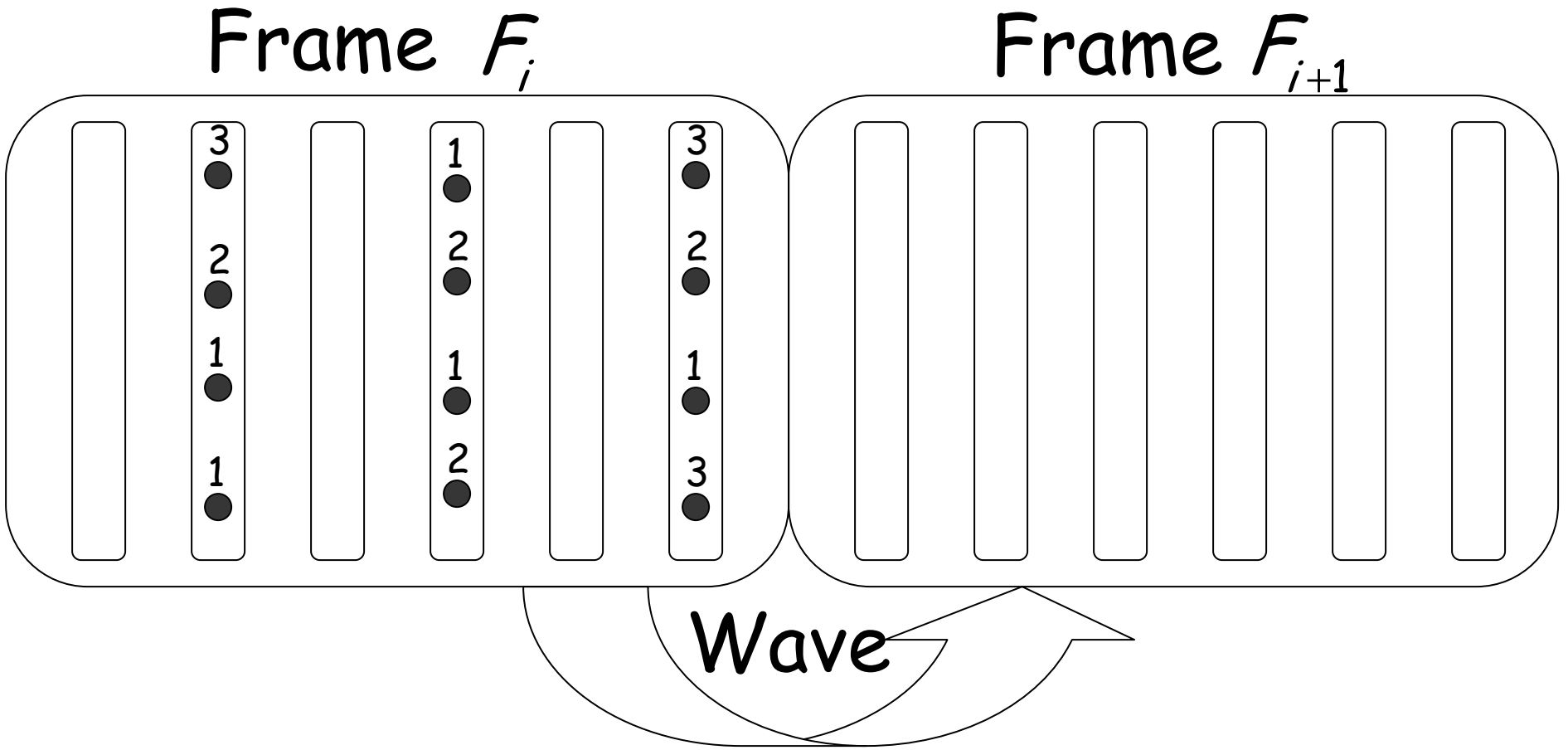
(A consequence of packet partitioning)



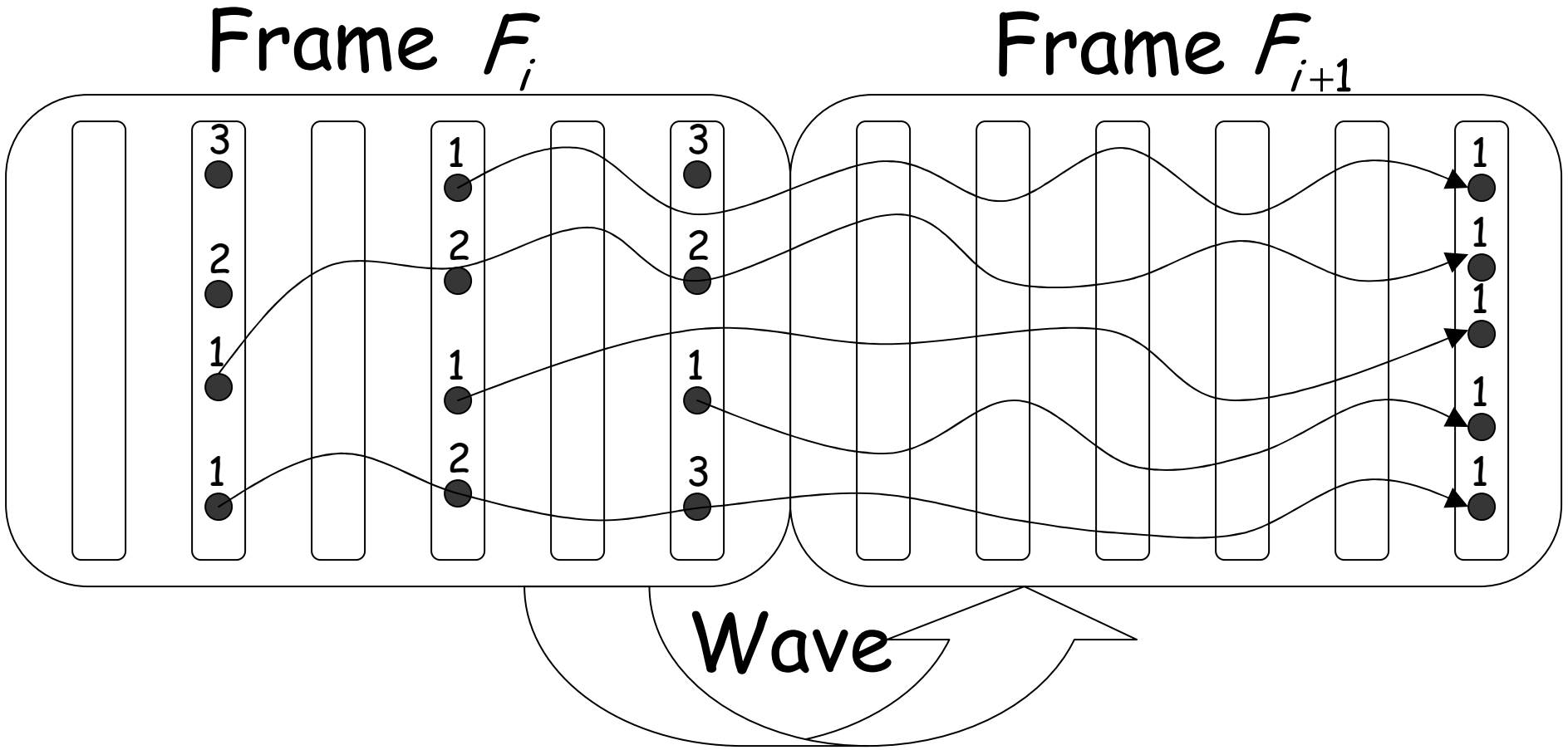
Thus the conflict graph can be colored with

$$\chi = \log(DN) + 1 \quad \text{colors}$$

We send packets of each color seperately



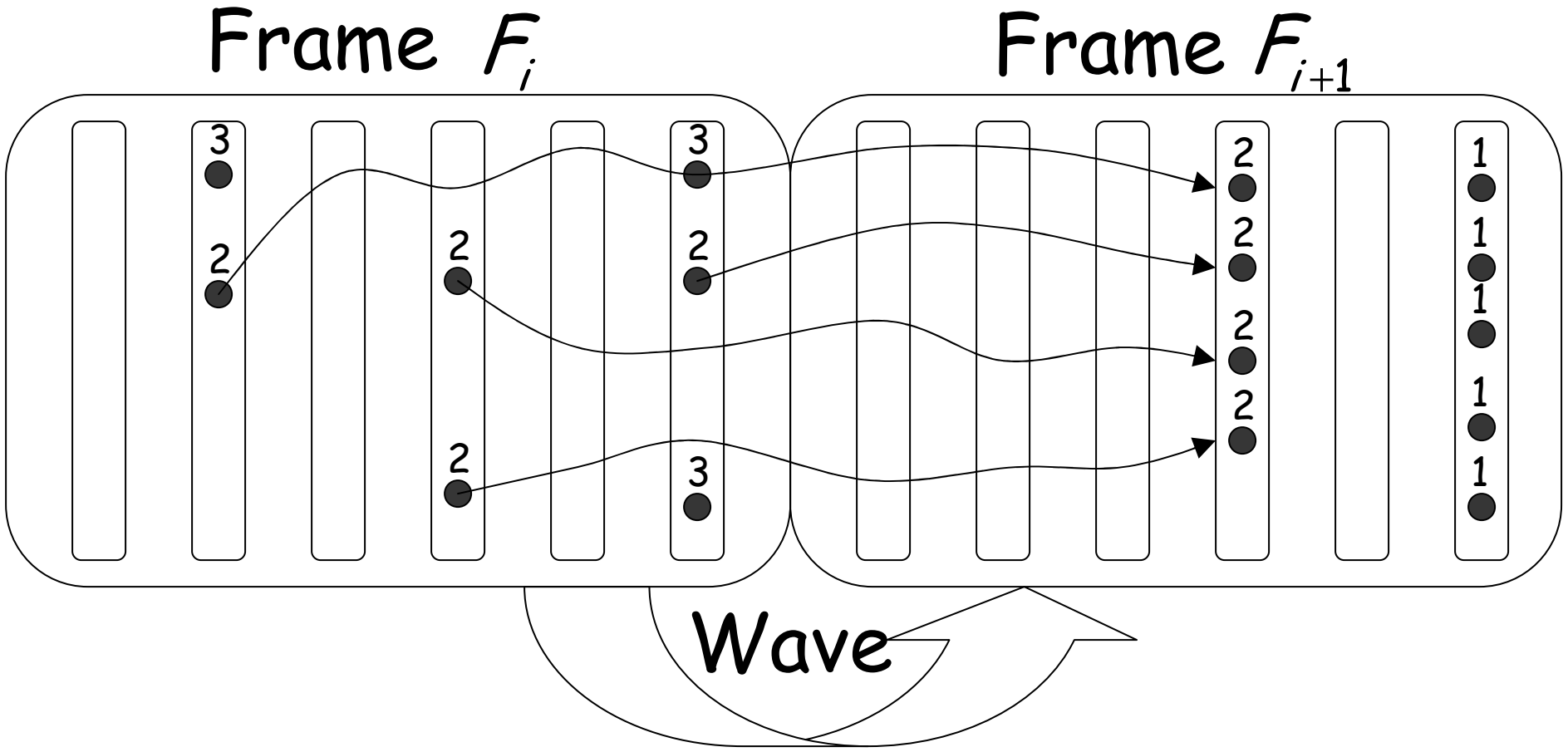
First send packets of color 1



Packet paths don't conflict

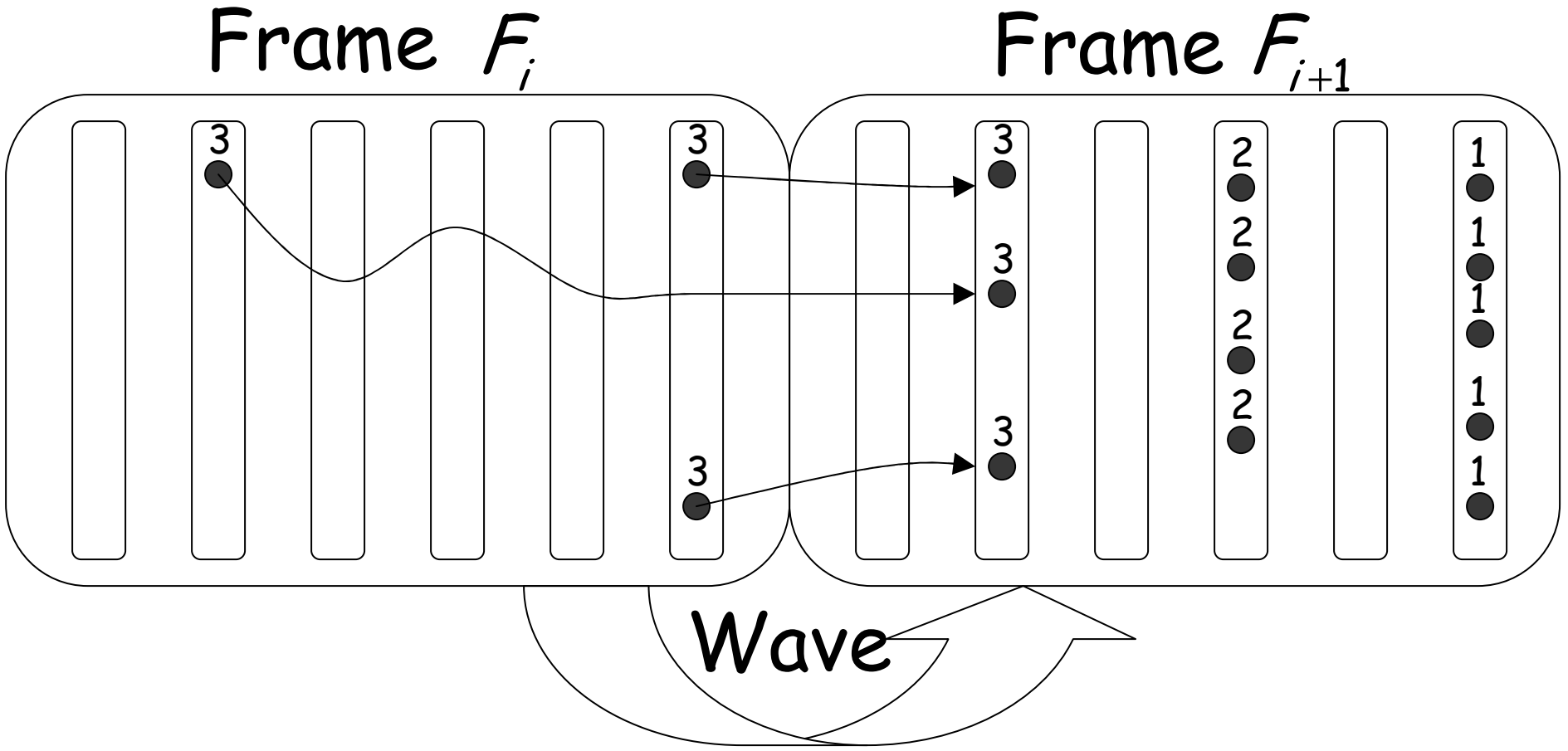
Time needed: $\Theta(\log(DN))$

Similarly, send packets of color 2



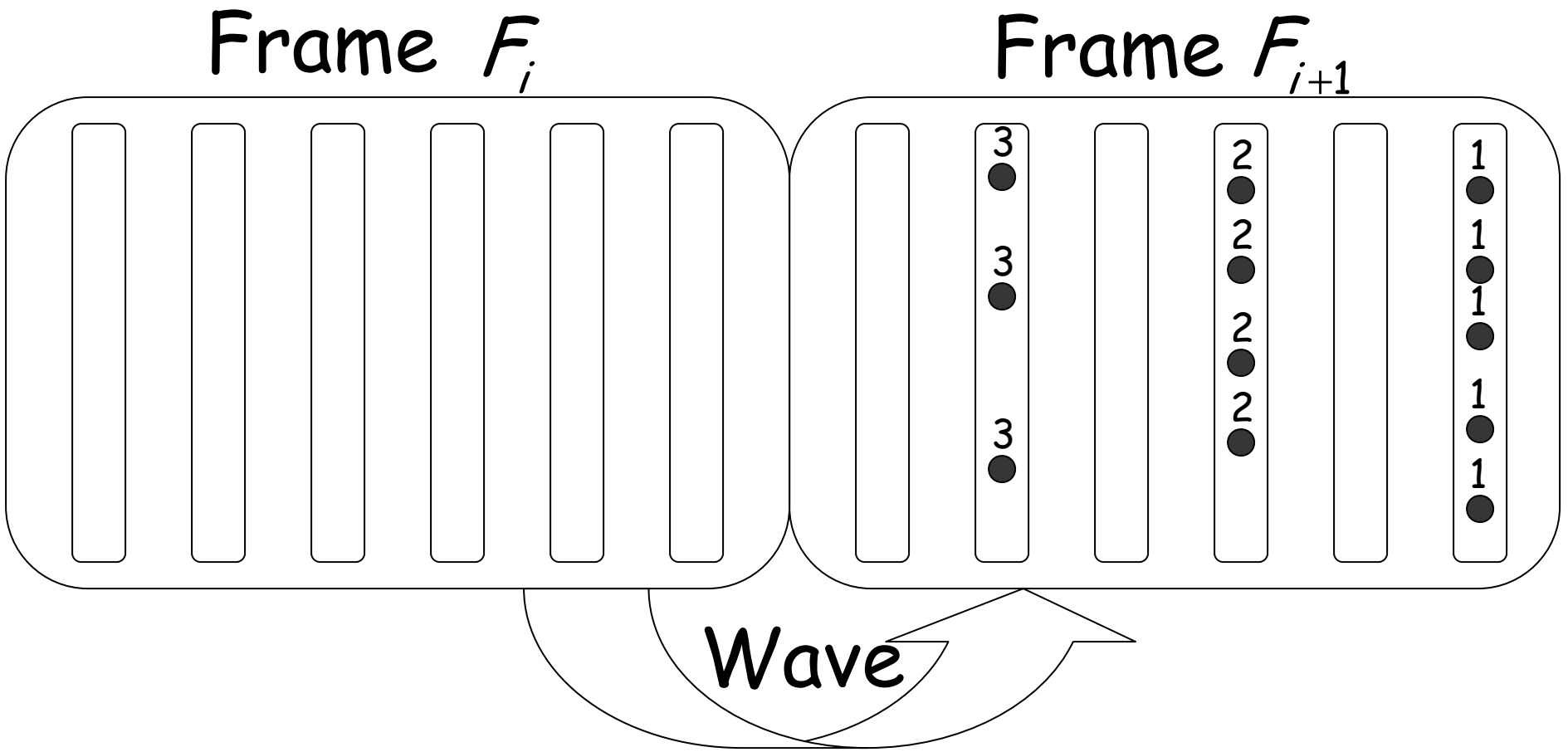
Packet paths don't conflict

Similarly, send packets of color 3



Packet paths don't conflict

All packets have been delivered



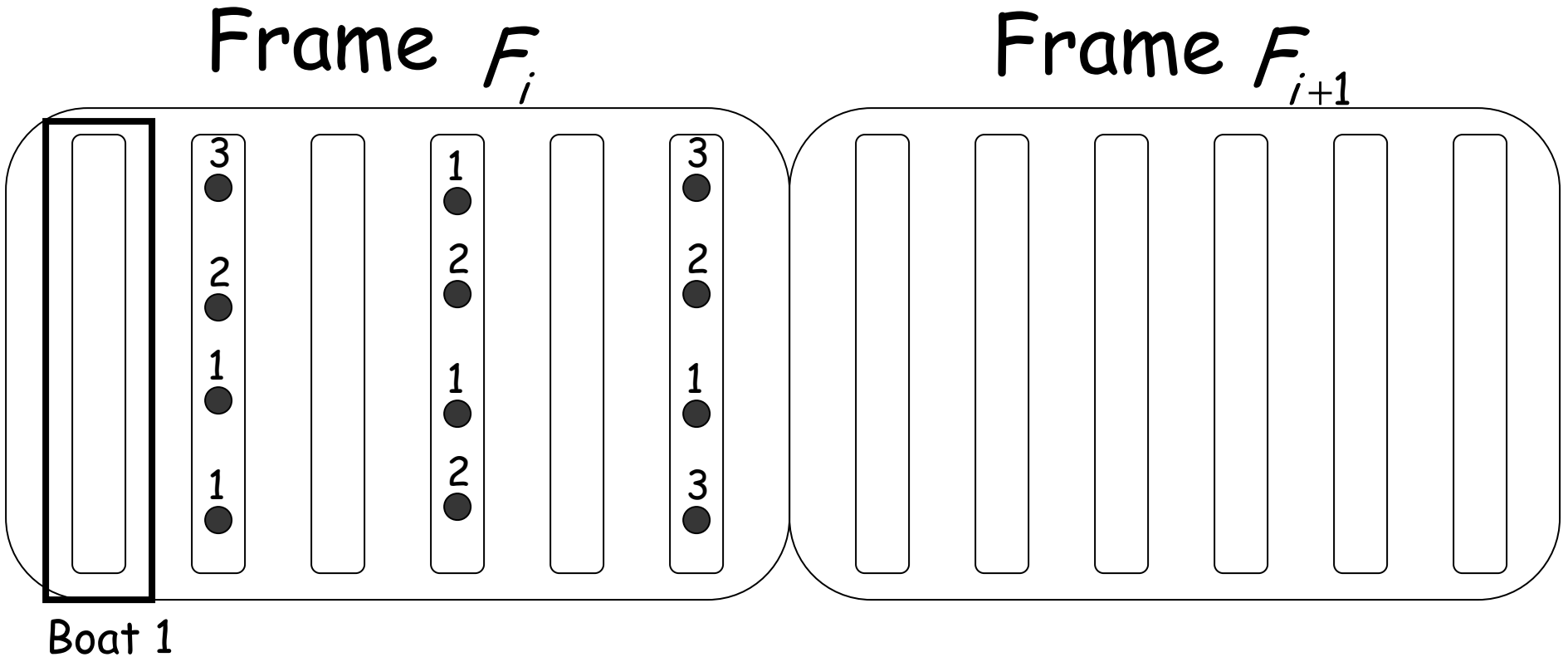
Wave time: Colors X 2 Frame size

$$\log(DN) \times 2\log(DN) = \Theta(\log^2(DN))$$

We can speed up the process
by pipelining different colors:

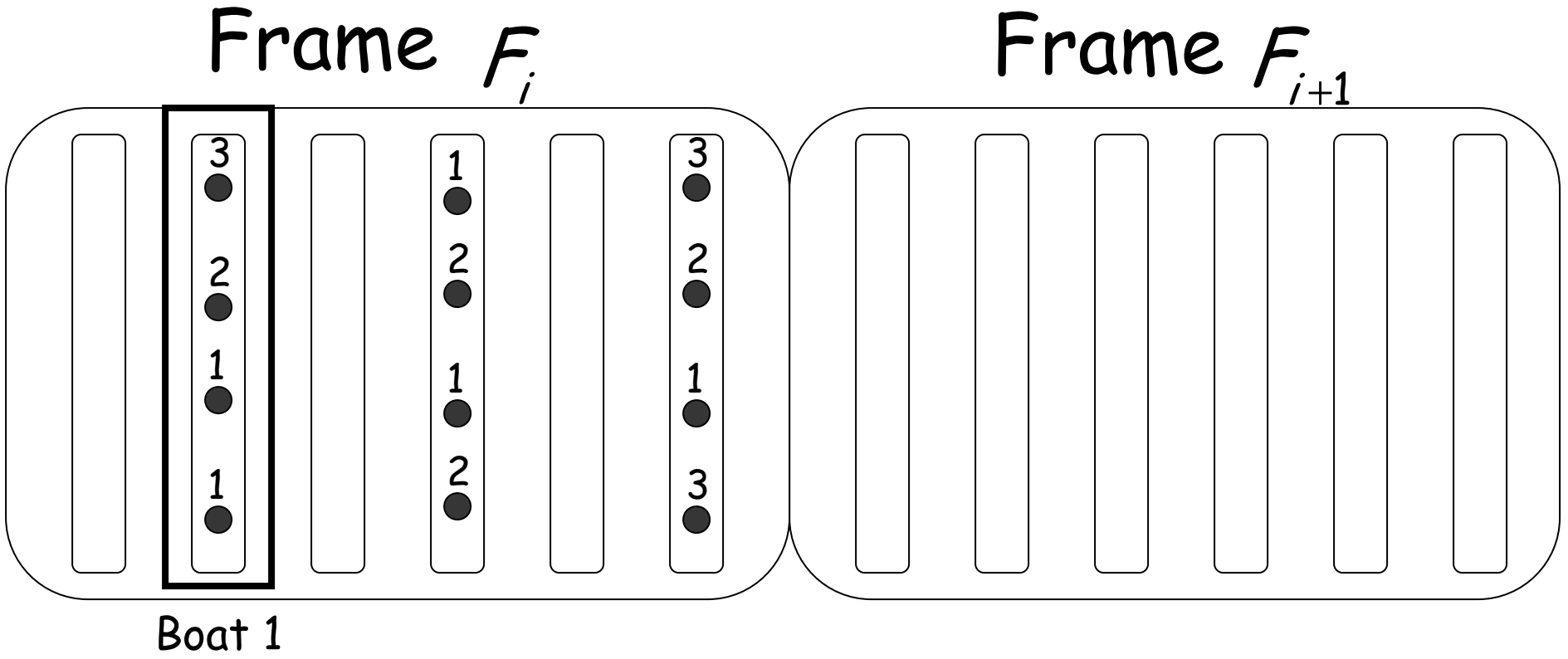
$$\Theta(\log(DN))$$

Pipelining using Boats



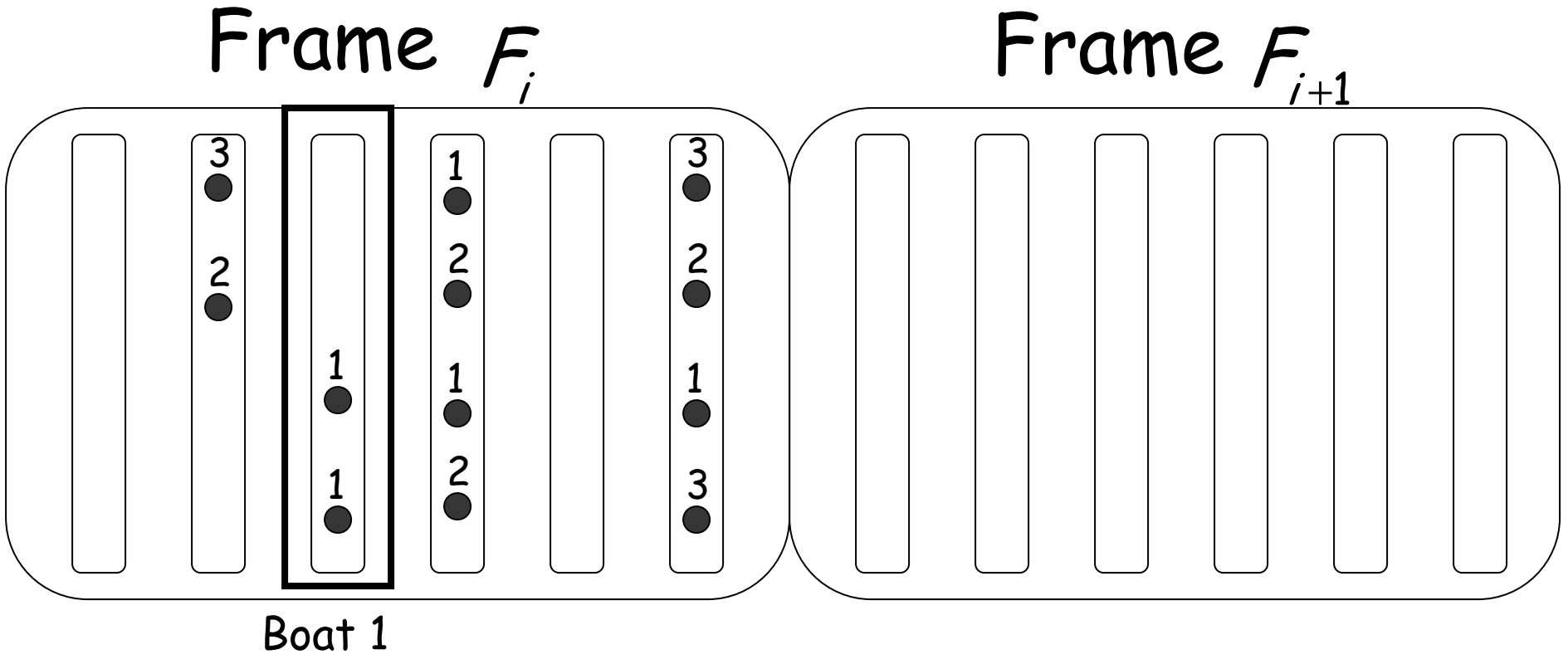
Packets of color i follow boat i

Pipelining using Boats



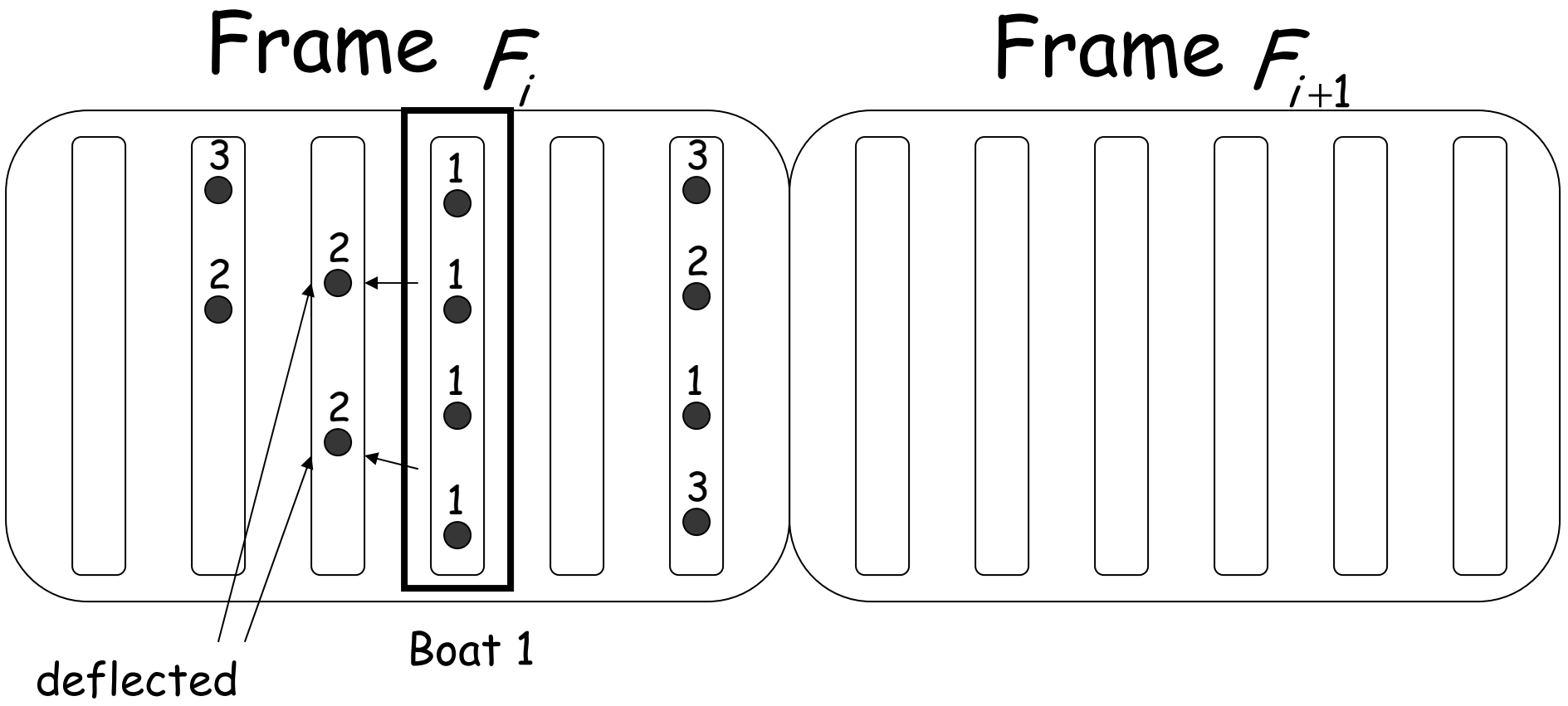
Packets of color i follow boat i

Pipelining using Boats



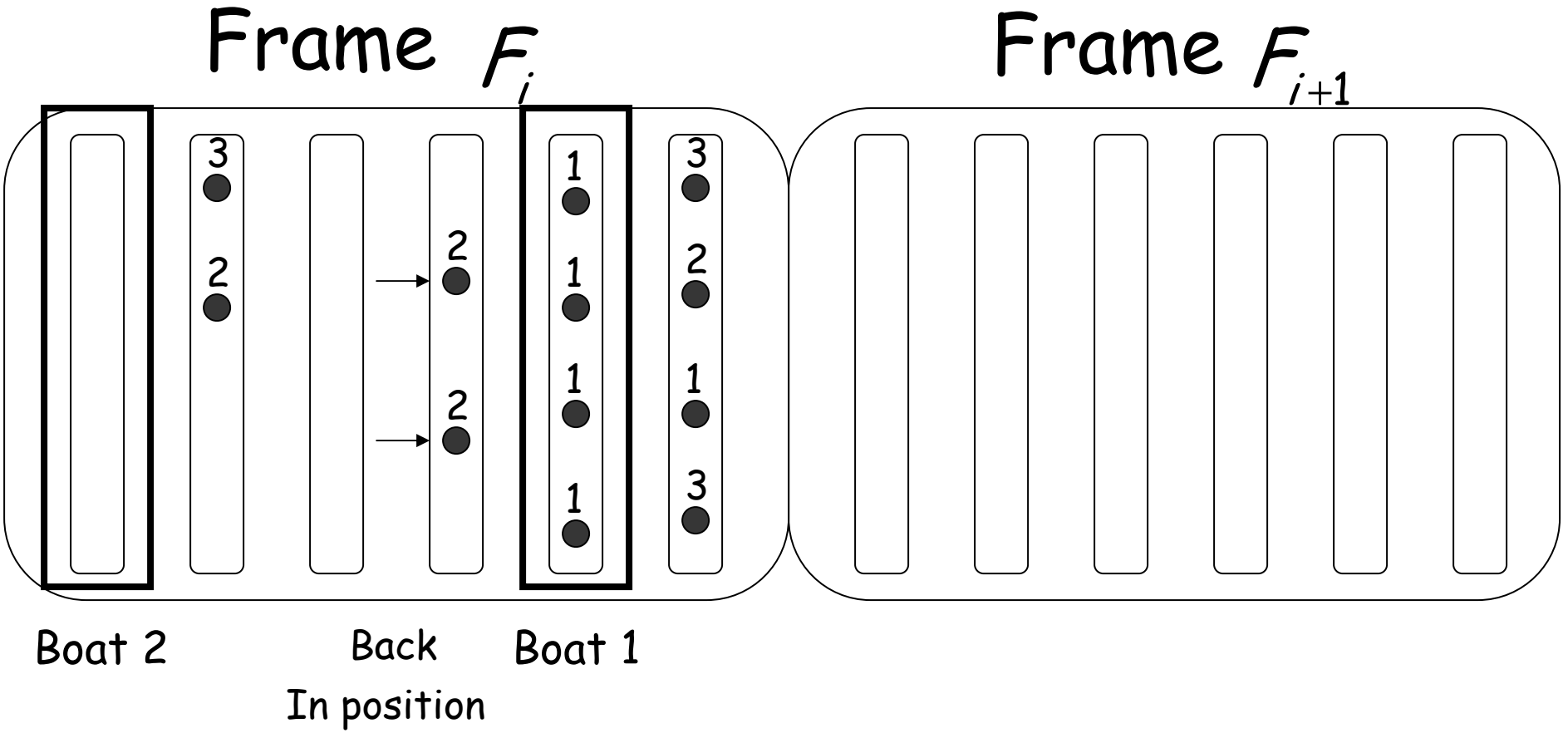
Packets of color i follow boat i

Pipelining using Boats



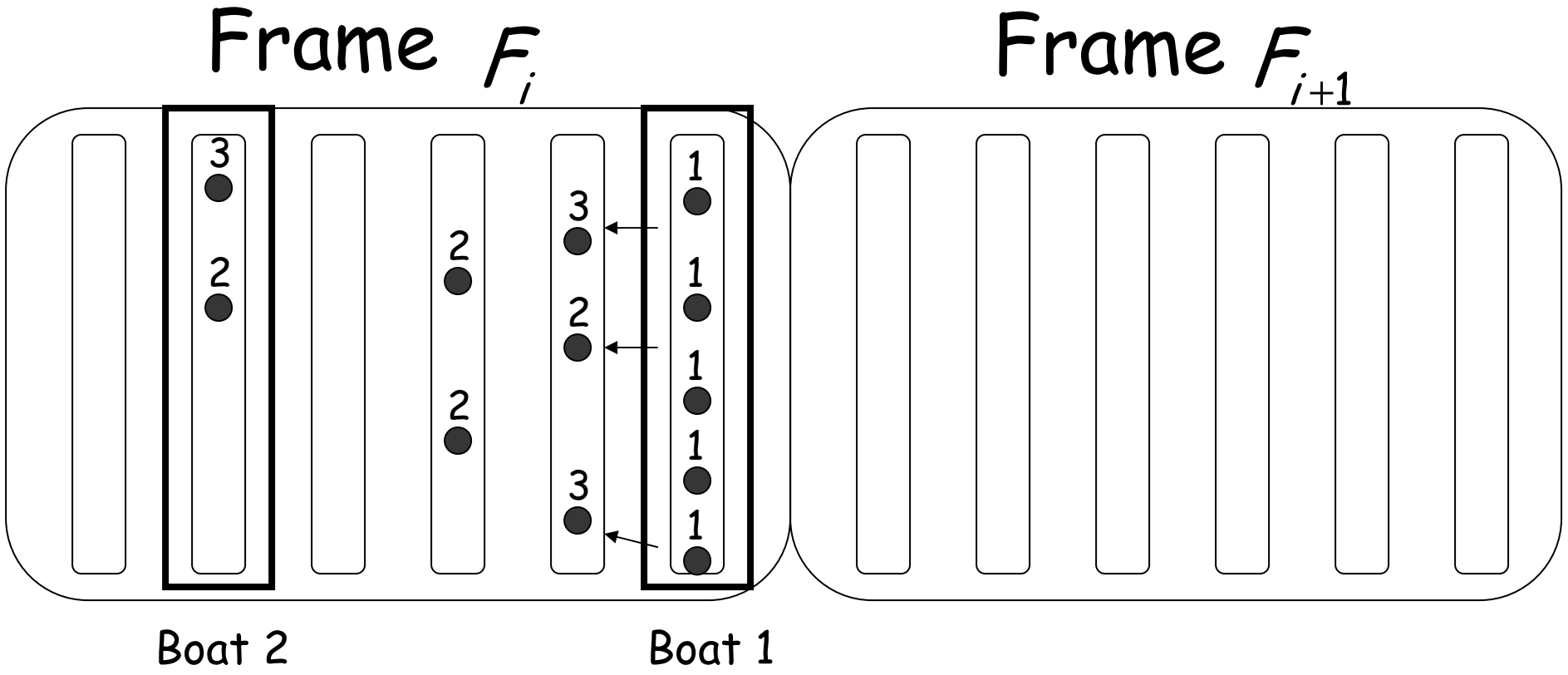
Packets of color i follow boat i

Pipelining using Boats



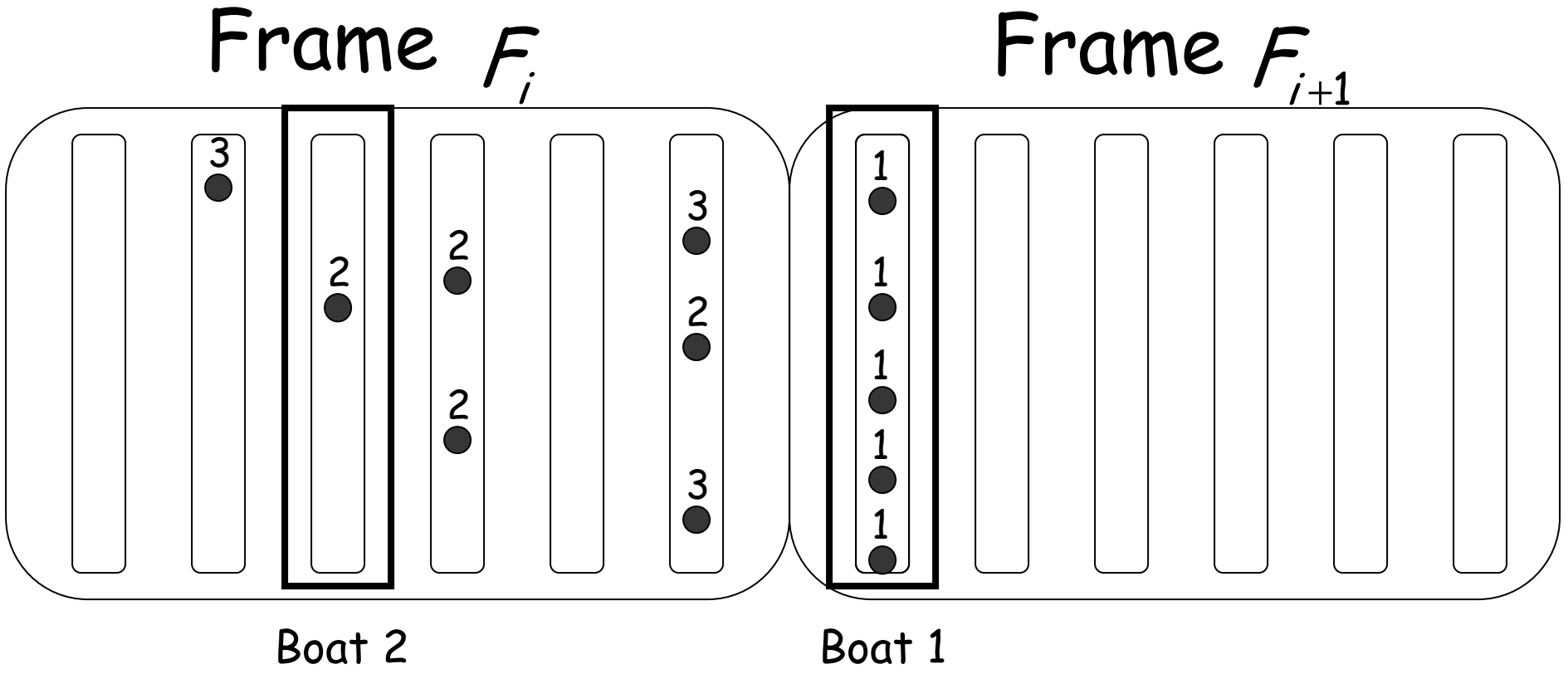
Packets of color i follow boat i

Pipelining using Boats



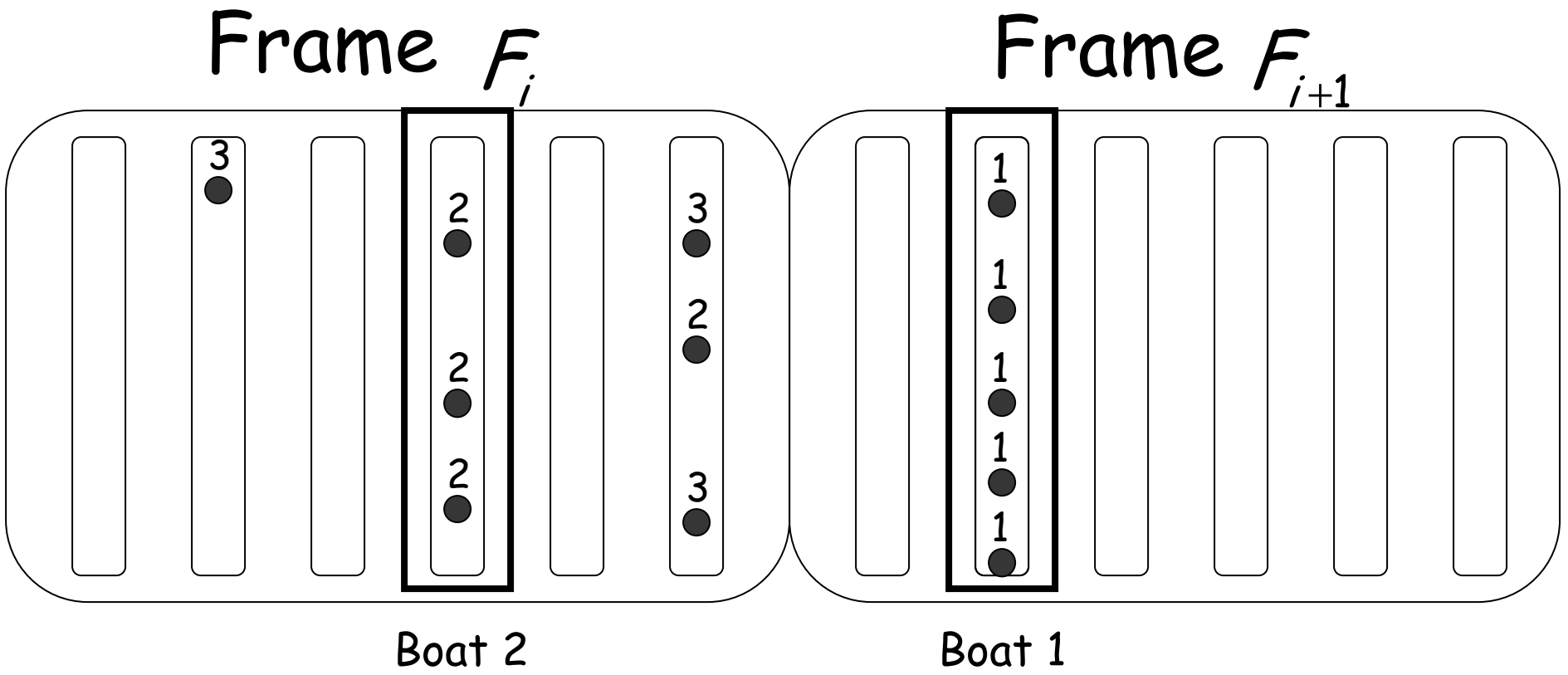
Packets of color i follow boat i

Pipelining using Boats



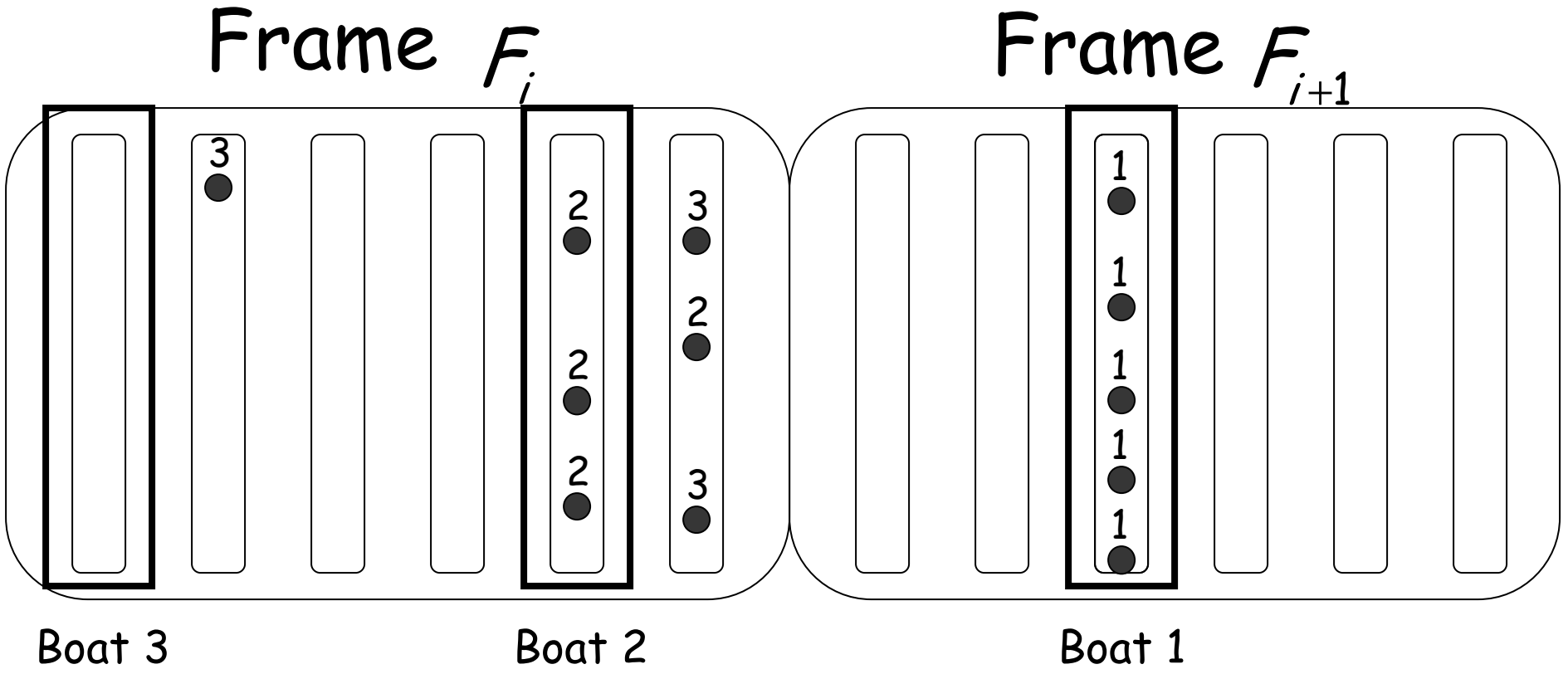
Packets of color i follow boat i

Pipelining using Boats



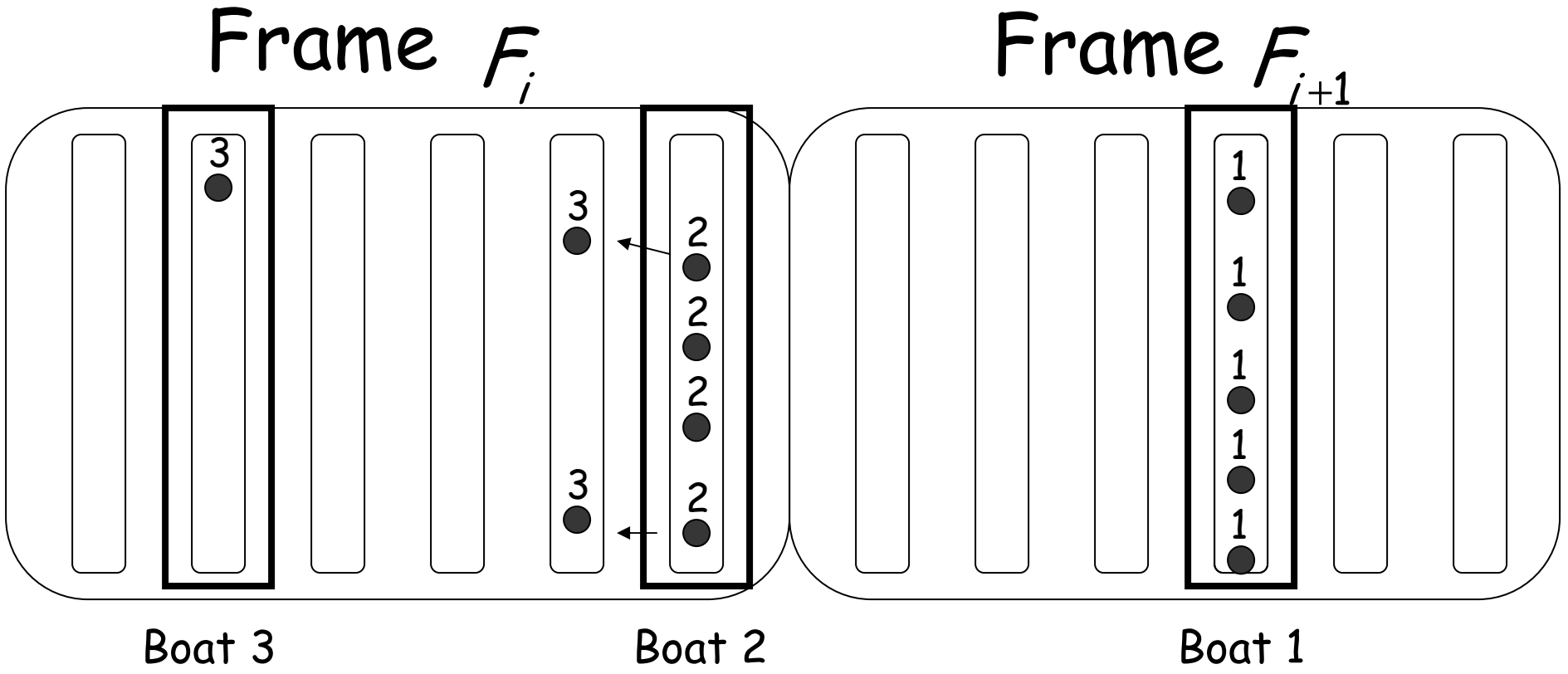
Packets of color i follow boat i

Pipelining using Boats



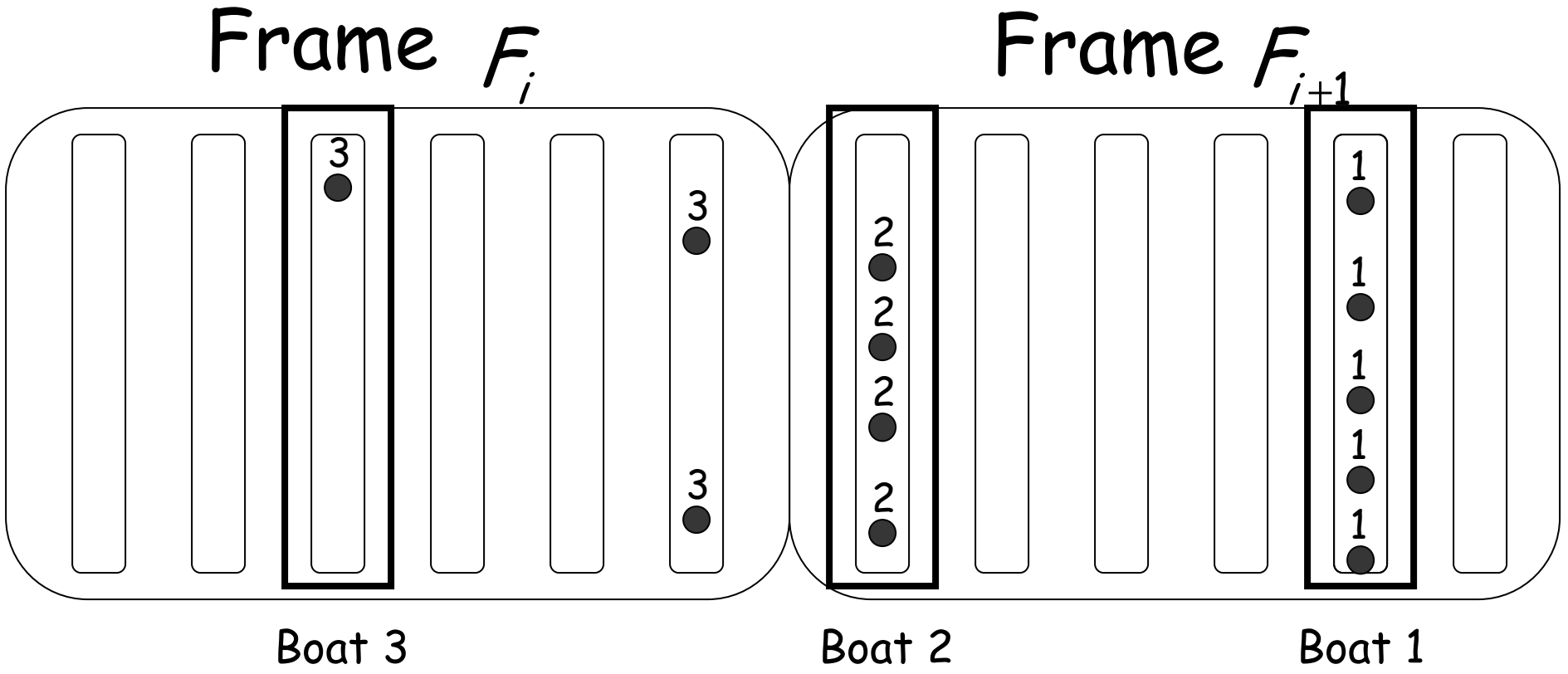
Packets of color i follow boat i

Pipelining using Boats



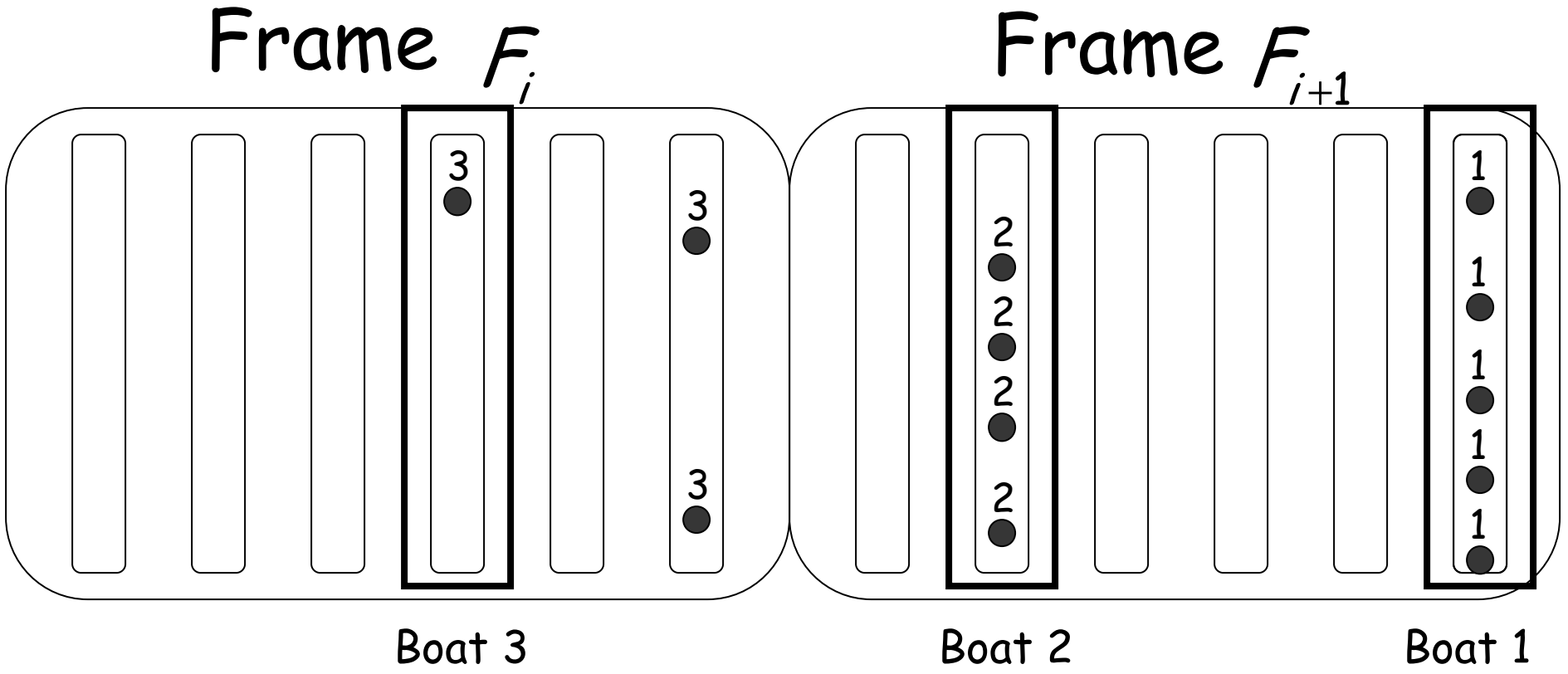
Packets of color i follow boat i

Pipelining using Boats



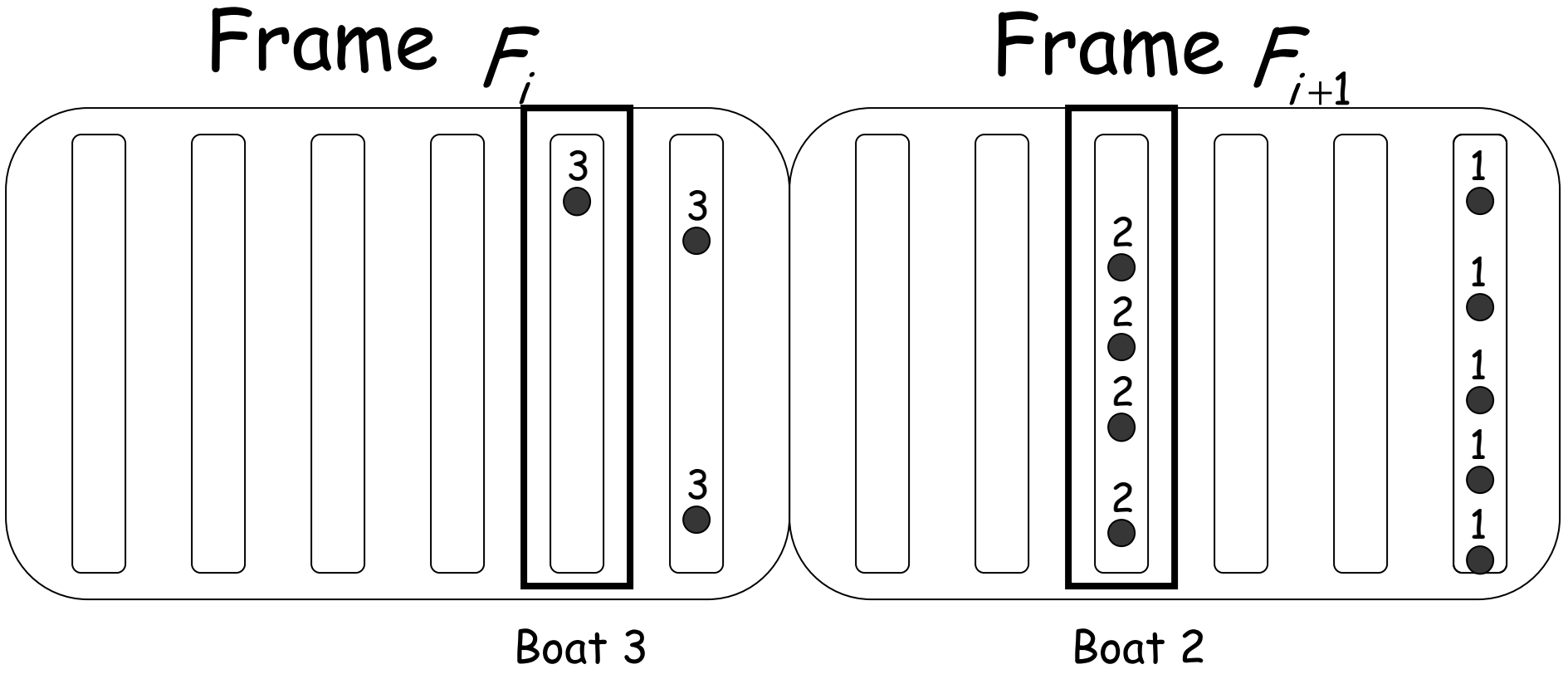
Packets of color i follow boat i

Pipelining using Boats



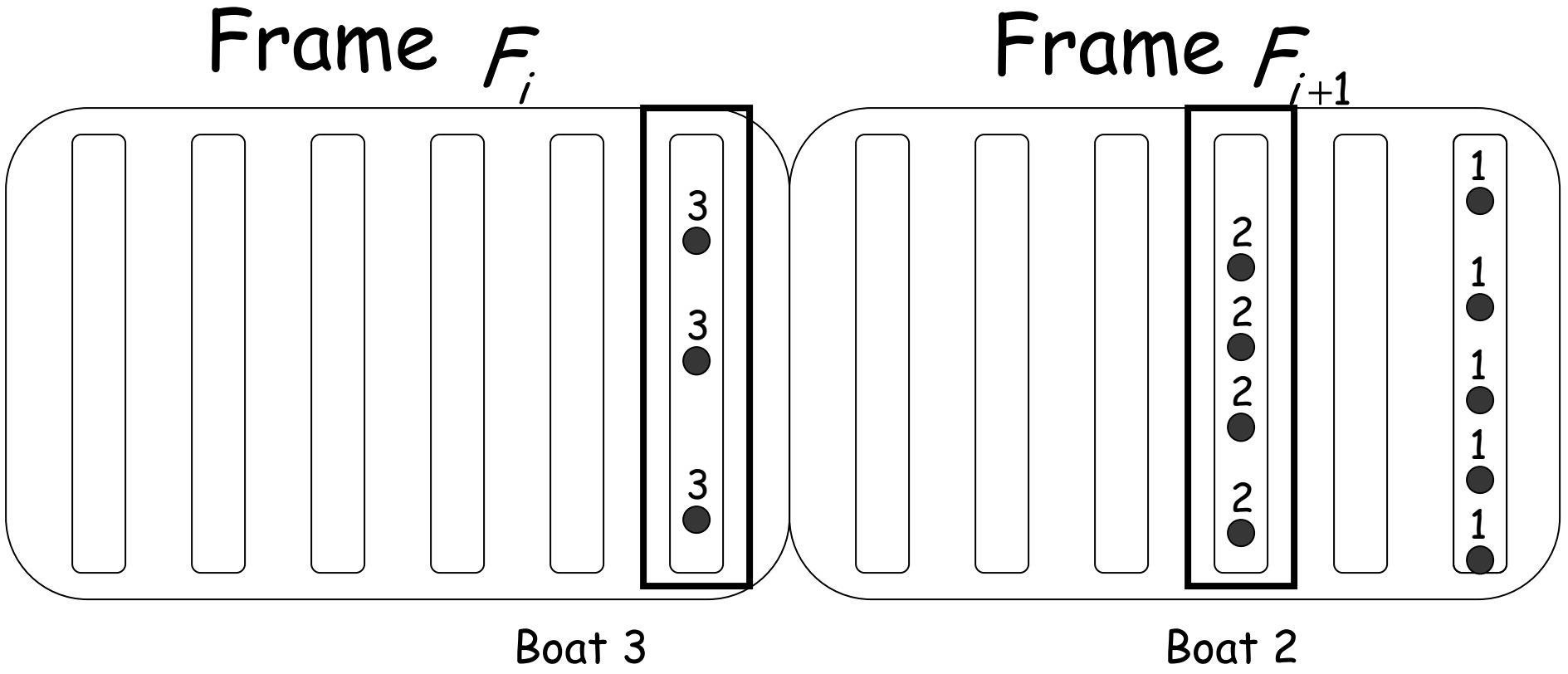
Packets of color i follow boat i

Pipelining using Boats



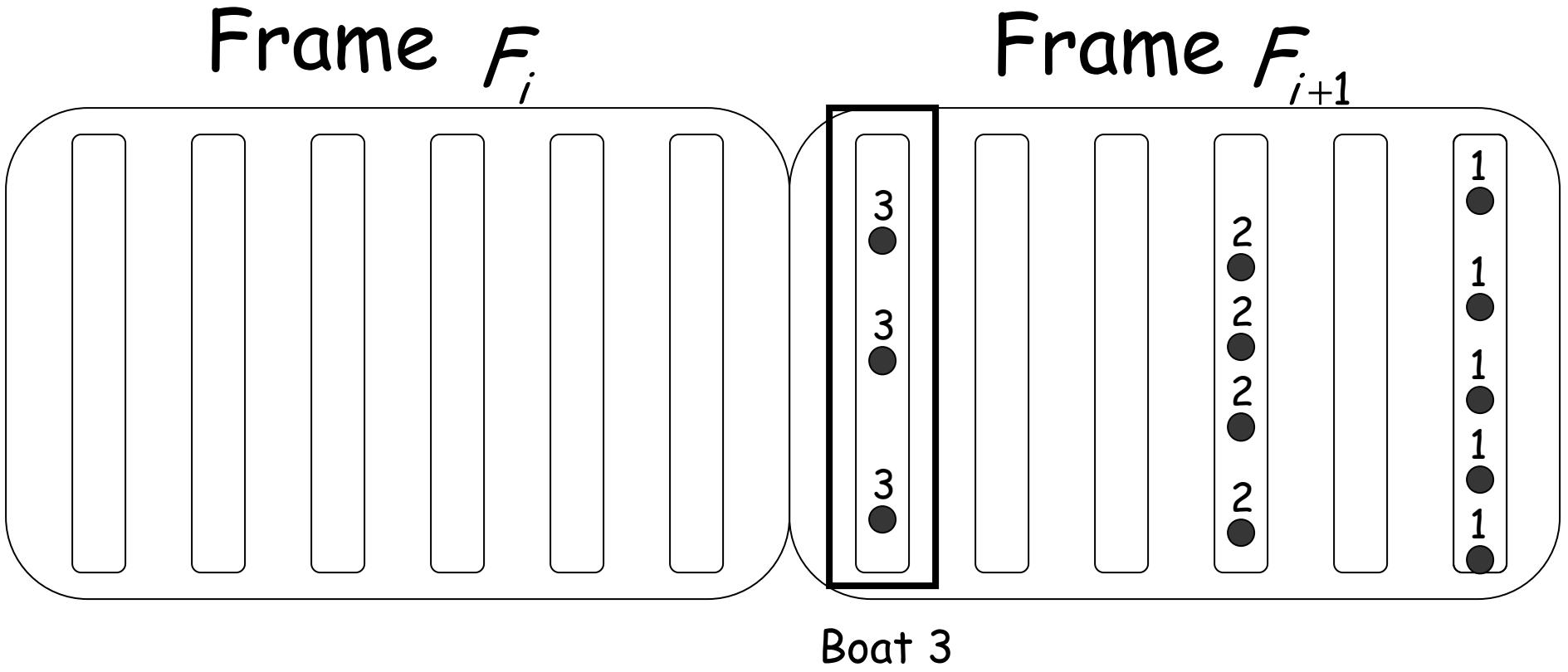
Packets of color i follow boat i

Pipelining using Boats



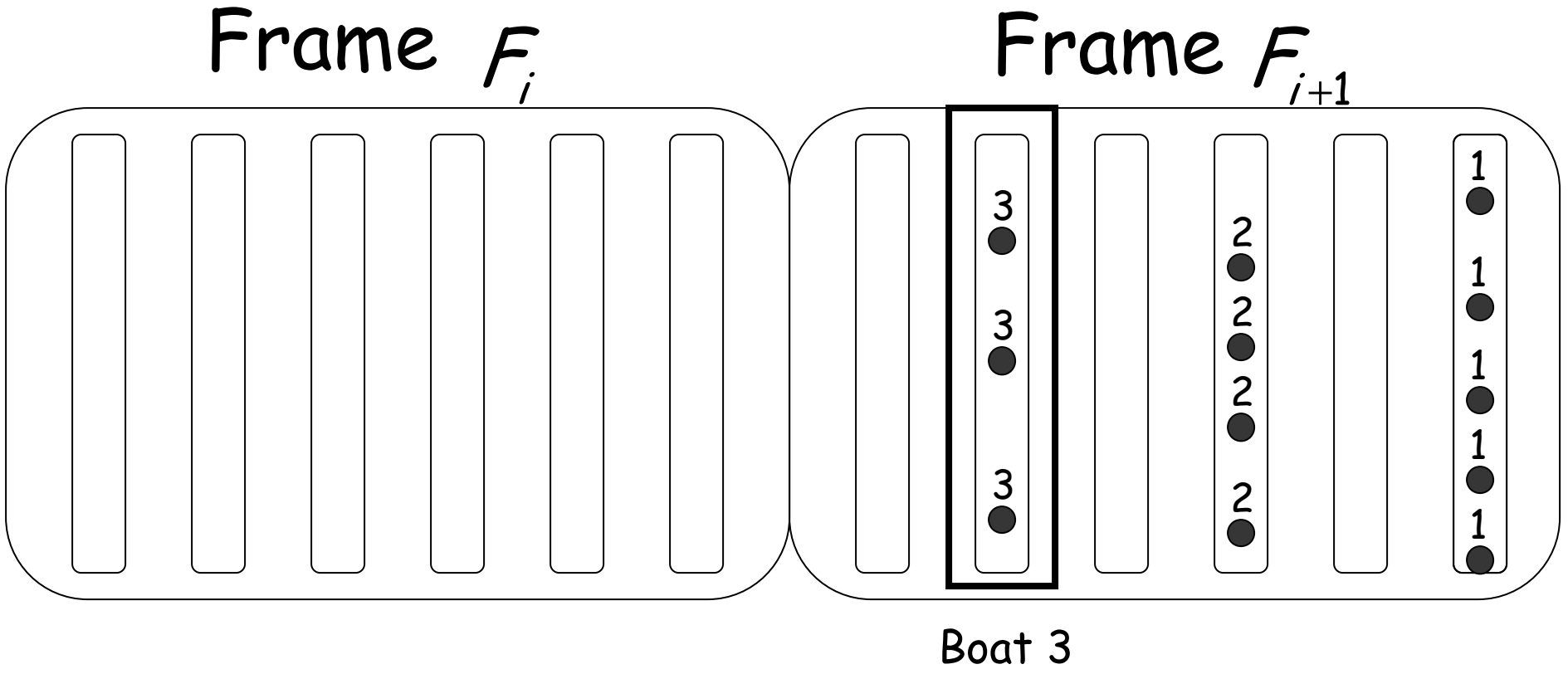
Packets of color i follow boat i

Pipelining using Boats



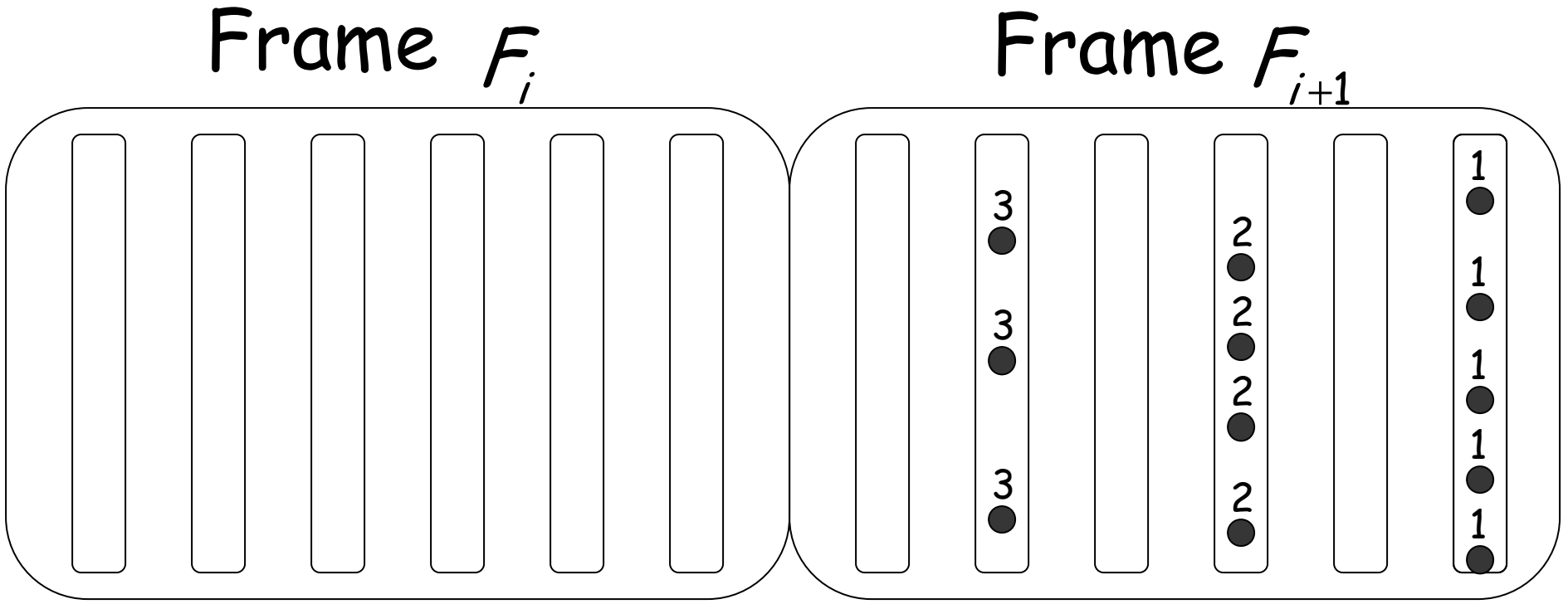
Packets of color i follow boat i

Pipelining using Boats



Packets of color i follow boat i

Pipelining using Boats



Packets of color i follow boat i

Wave time: Time until last boat reaches target level

Number of colors

Frame size

$$4\chi + 2\log(DN)$$
$$= 4\log(DN) + \log(DN)$$
$$= \Theta(\log(DN))$$

Talk Outline

- Introduction
- Centralized Algorithm
- Distributed Algorithm
- Conclusion

In the distributed version,
we assume that every node knows
parameters C, D, N

Nodes do not know the packet paths,
except for the packets in them.

The distributed algorithm
is the same with the centralized,
except for one thing:

The conflict graph is colored in a distributed manner

Distributed coloring – Basic Idea

During a wave:

1. each packet chooses a random color Between 0 and 2log(DN)

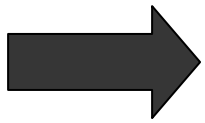
2. each packet assumes the color is correct and follows the respective boat

3. For packets that conflict, the process repeats

This process repeats a logarithmic number of times, thus it gives an extra logarithmic factor in the performance

Talk Outline

- Introduction
- Centralized Algorithm
- Distributed Algorithm
- Conclusion



Conclusion

- Centralized algorithm $O(C \cdot \log(DN) + D)$

  Uses conflict graph coloring

- Distributed Algorithm

  Worse by a logarithmic factor

  Implements distributed coloring