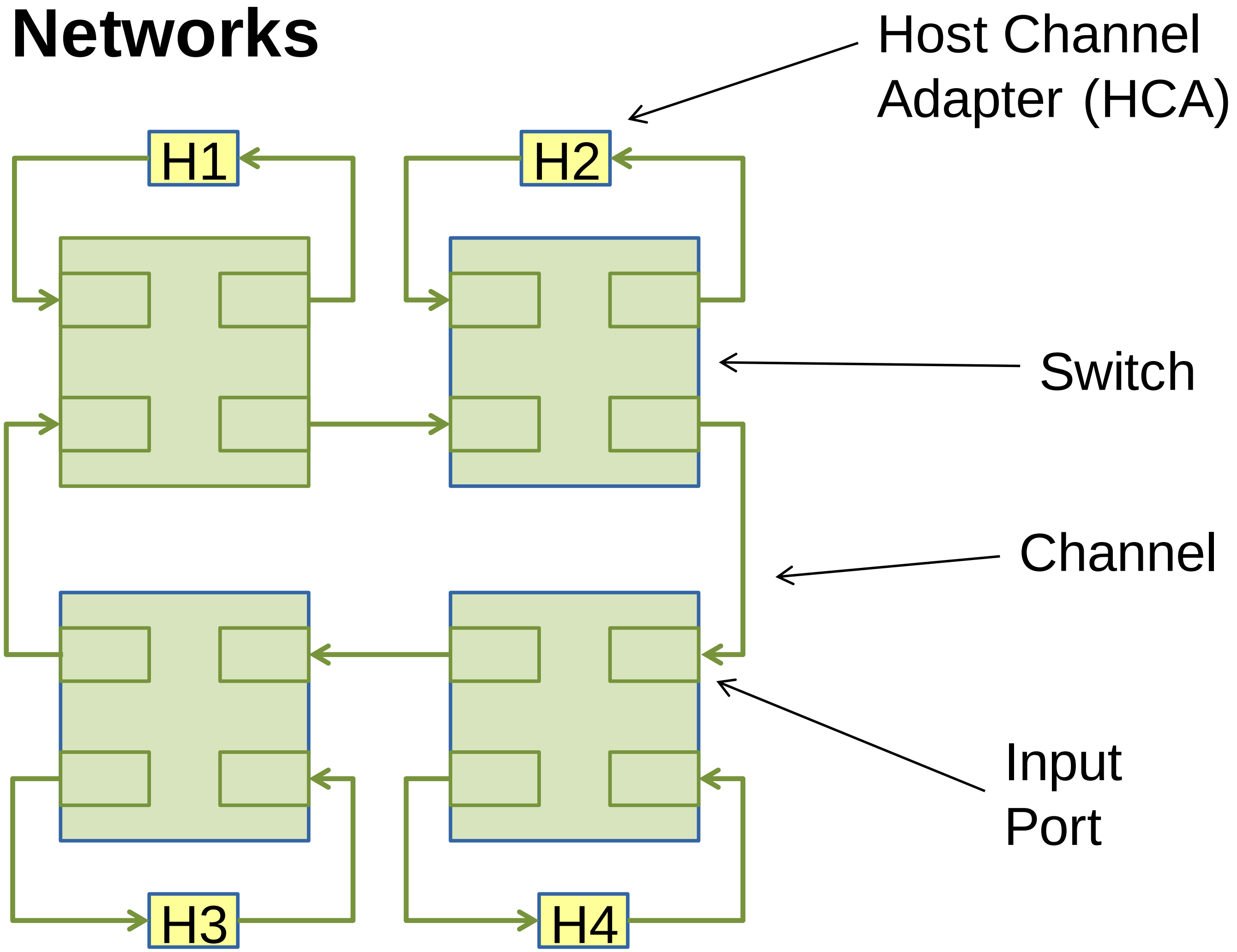
Timo Schneider, Otto Bibartiu, Torsten Hoefler

Ensuring Deadlock-Freedom in Low-Diameter InfiniBand Networks

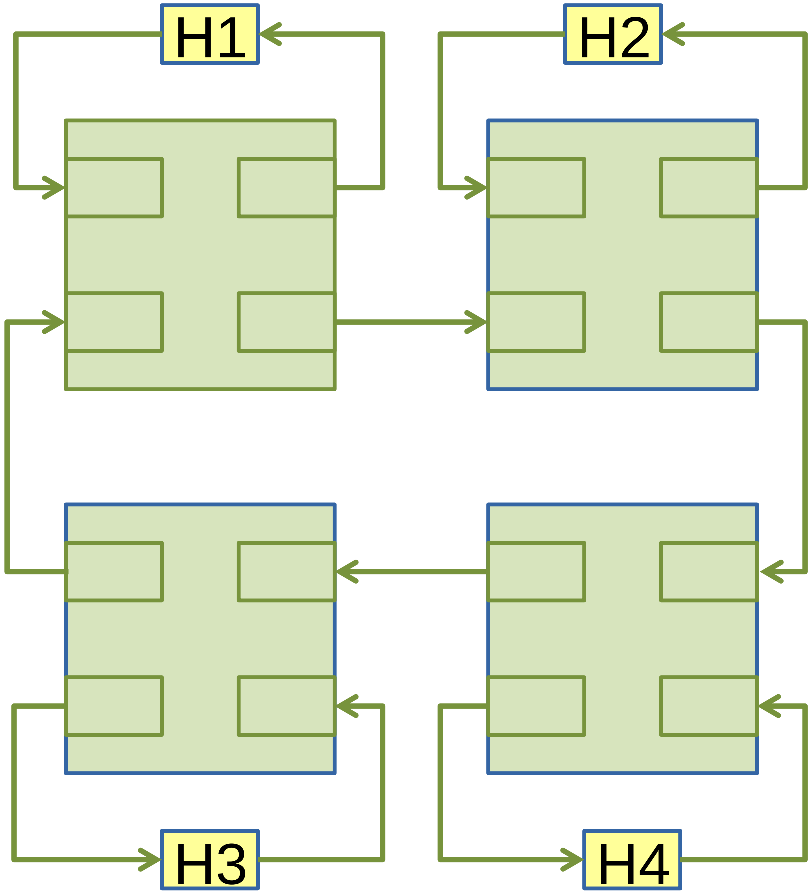
Systems@ETH zürich

InfiniBand Networks

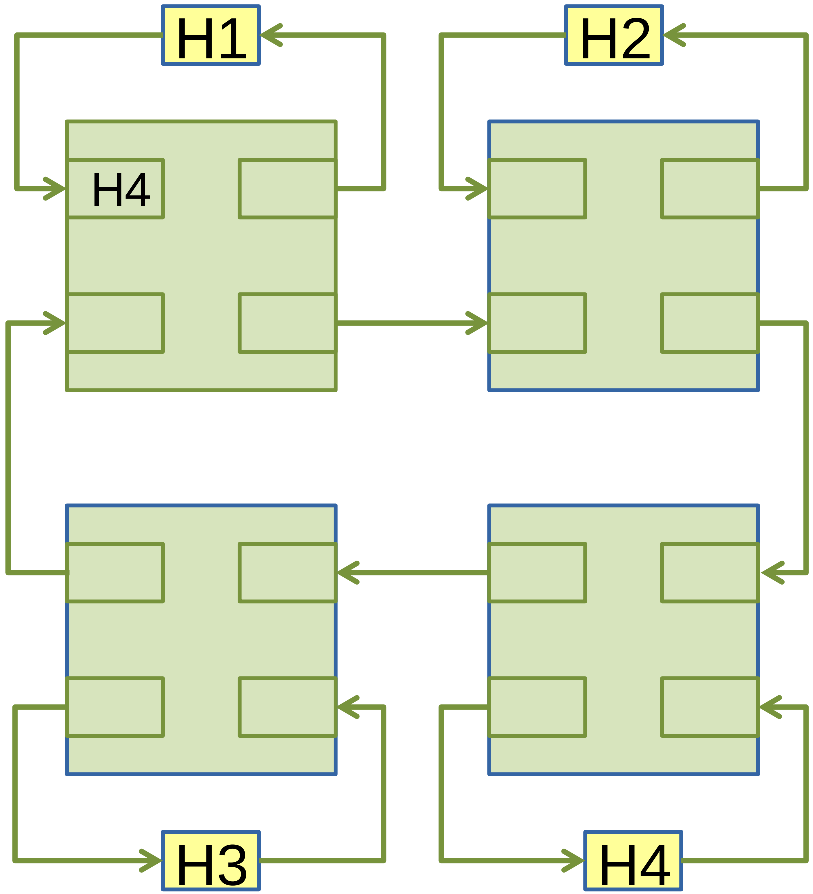


Switches and HCAs, connected via unidirectional channels.
We model this as a graph ( )

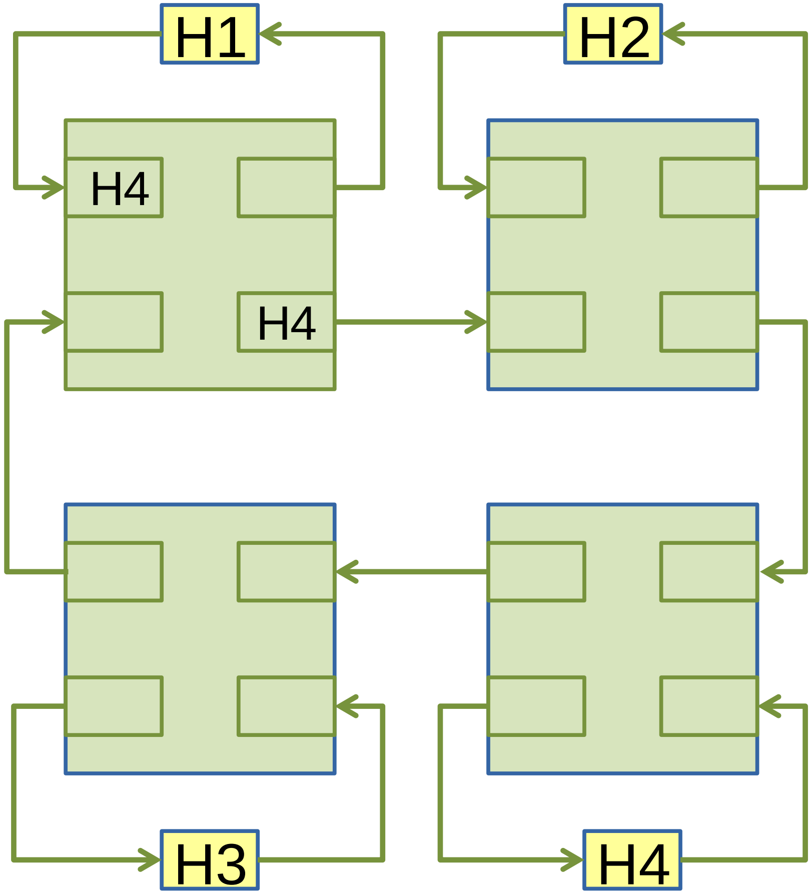
InfiniBand Networks



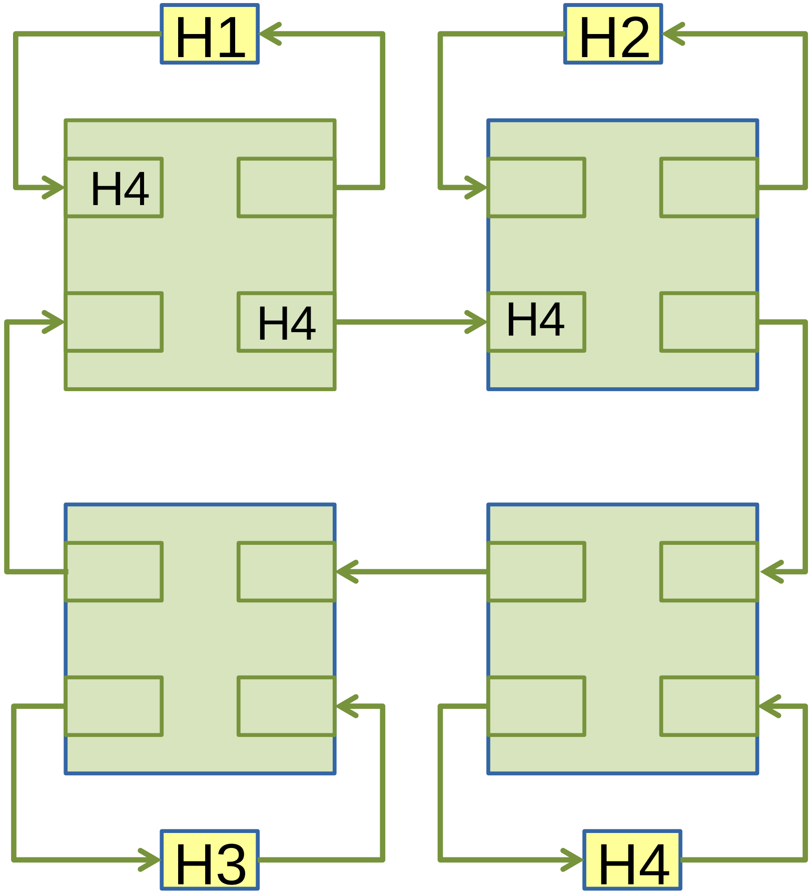
InfiniBand Networks



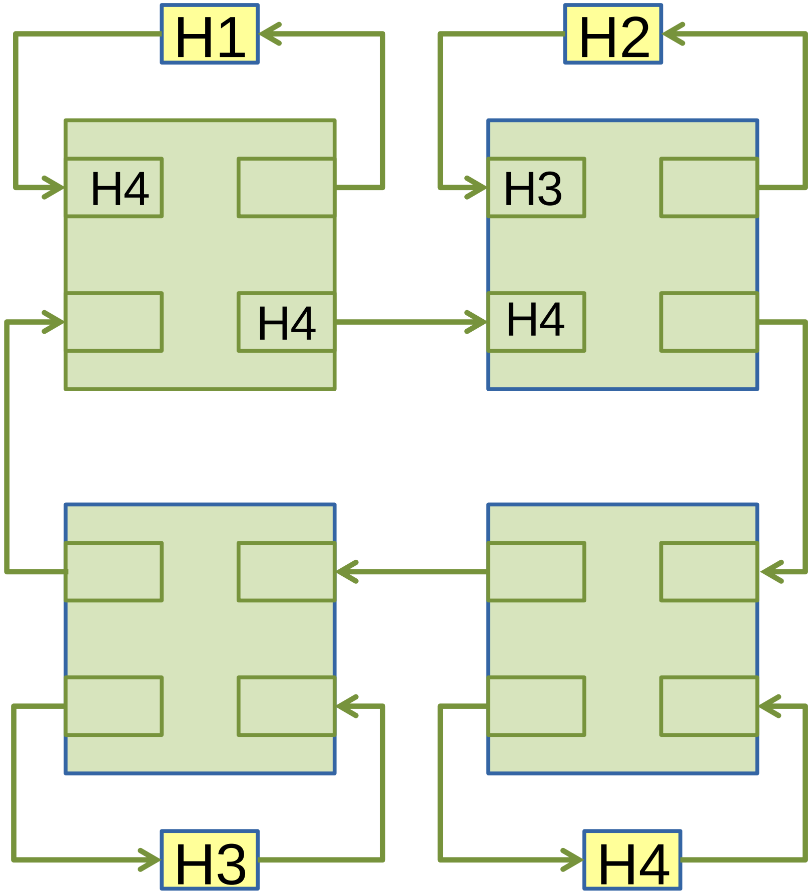
InfiniBand Networks



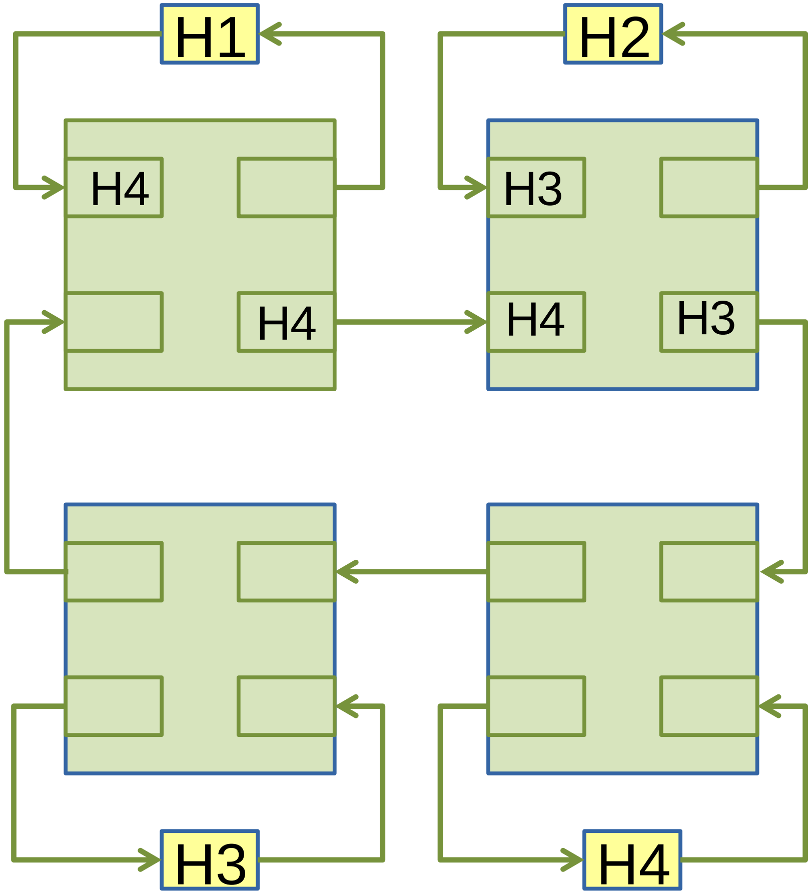
InfiniBand Networks



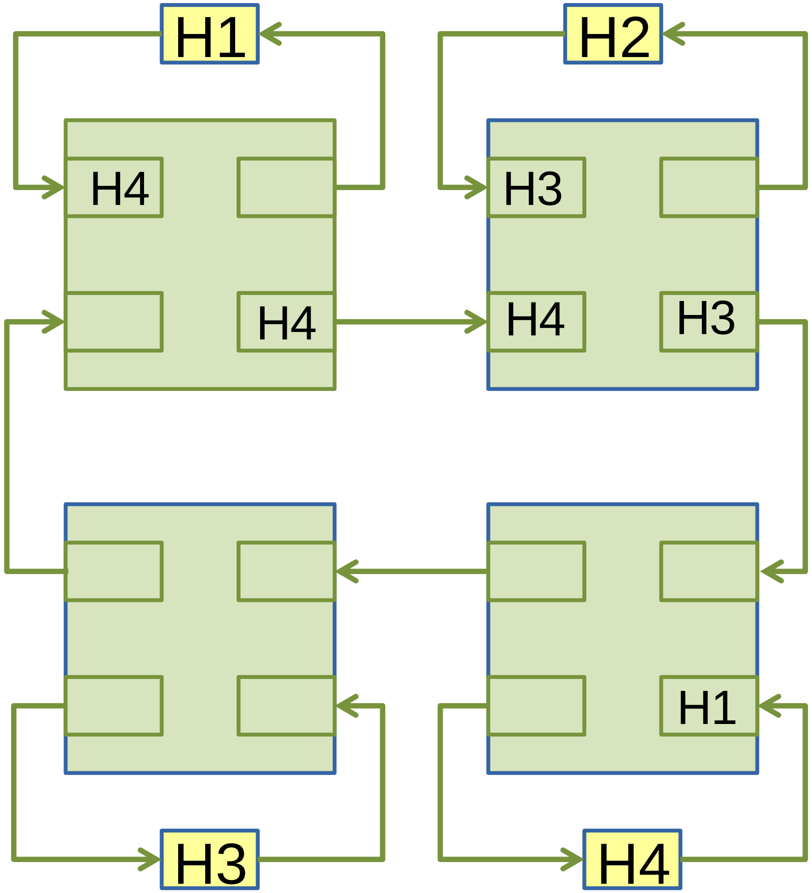
InfiniBand Networks



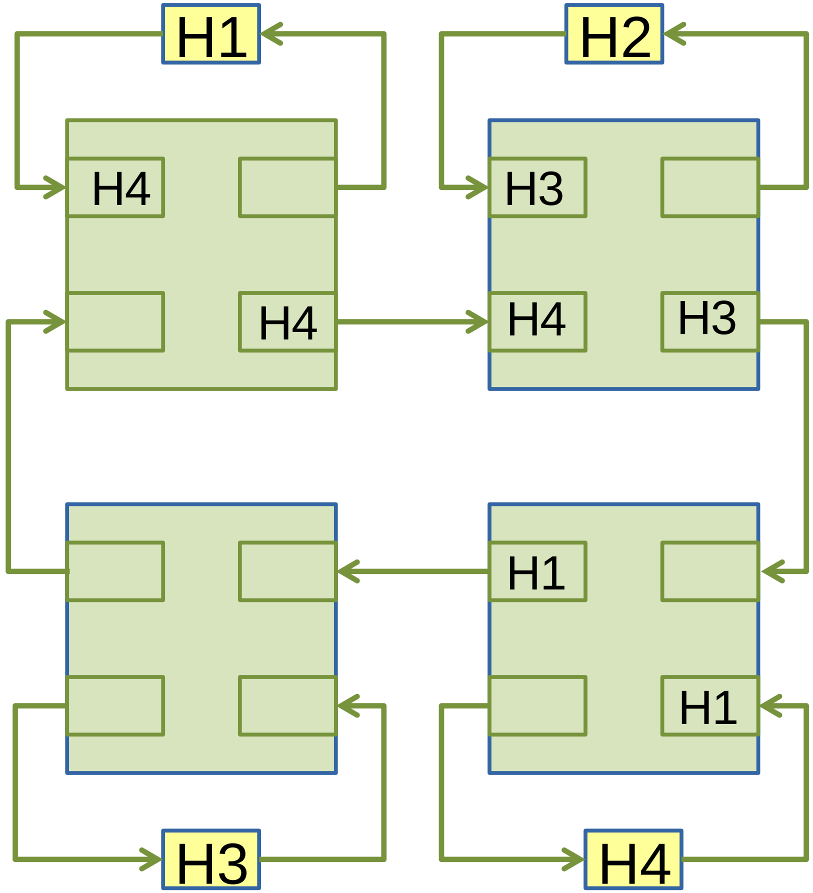
InfiniBand Networks



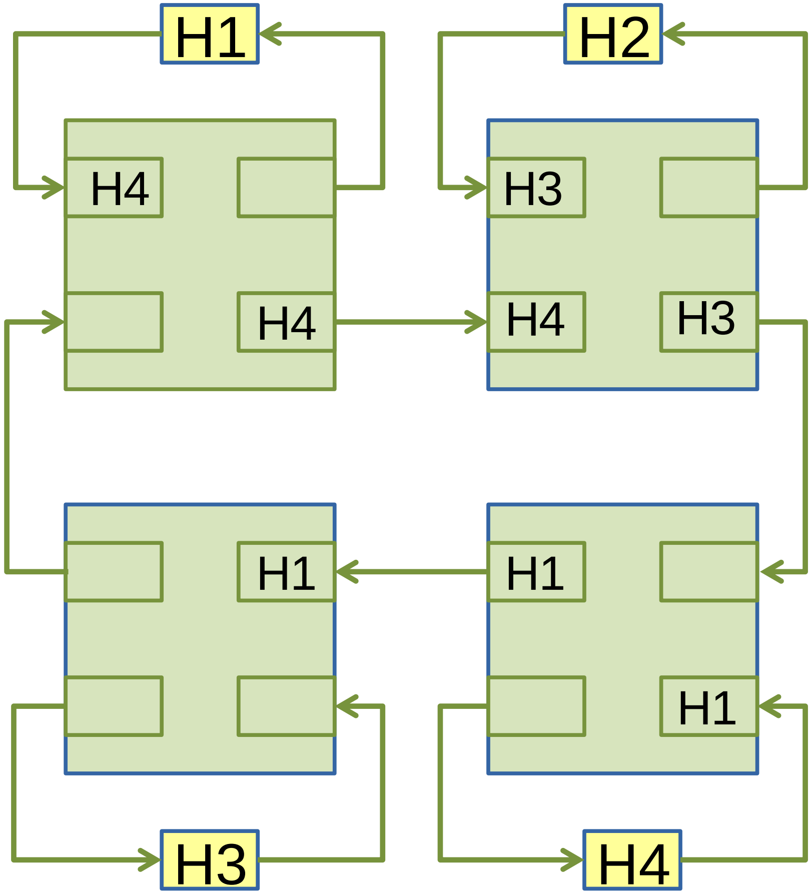
InfiniBand Networks



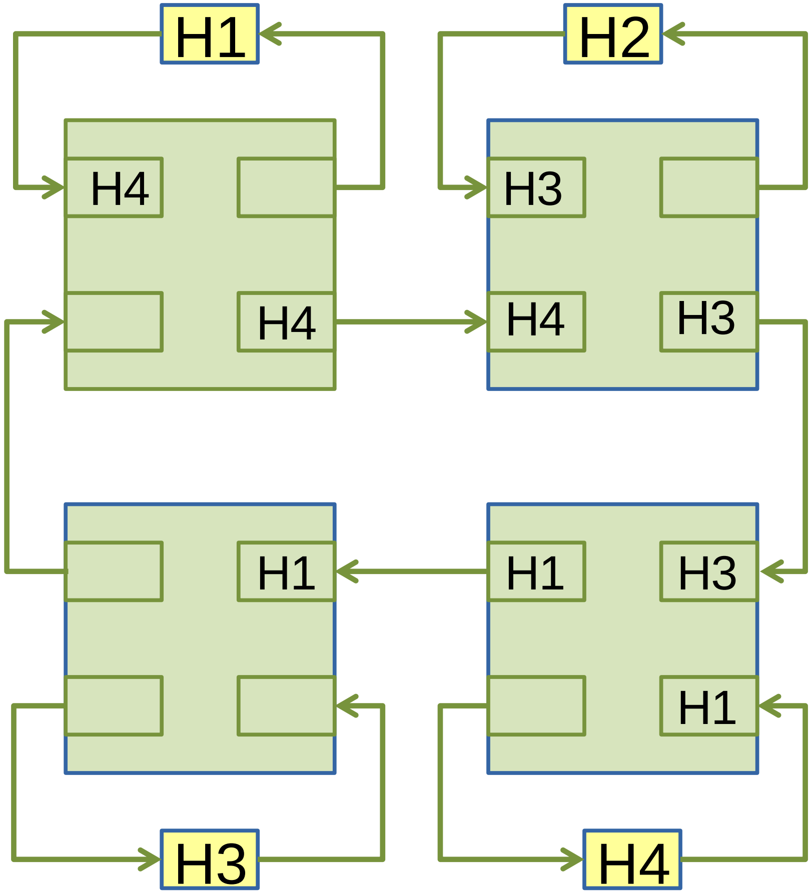
InfiniBand Networks



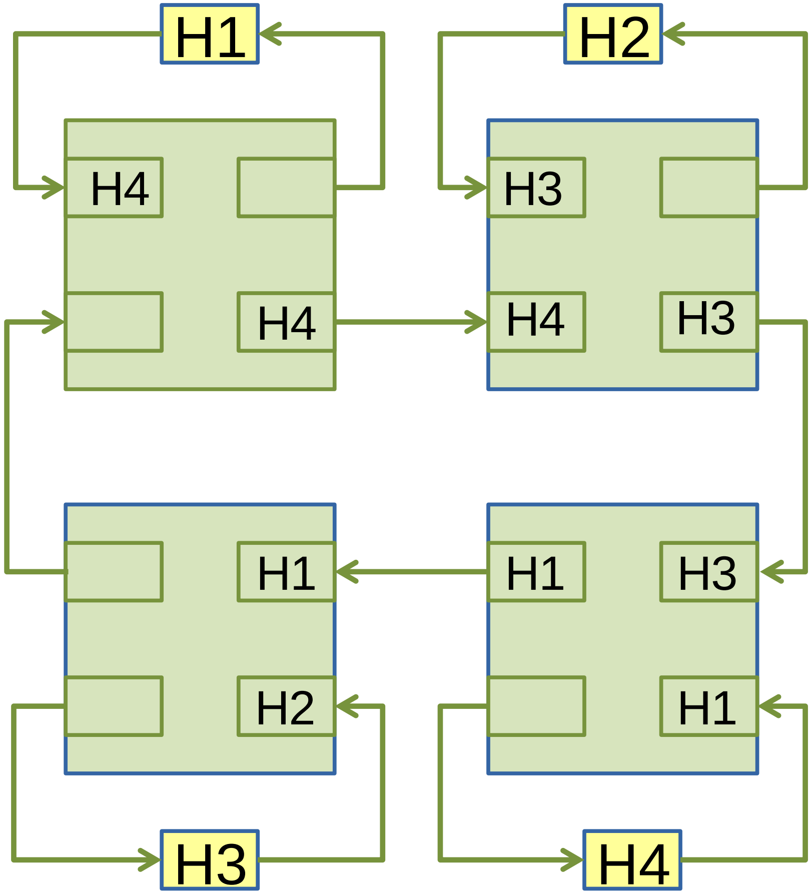
InfiniBand Networks



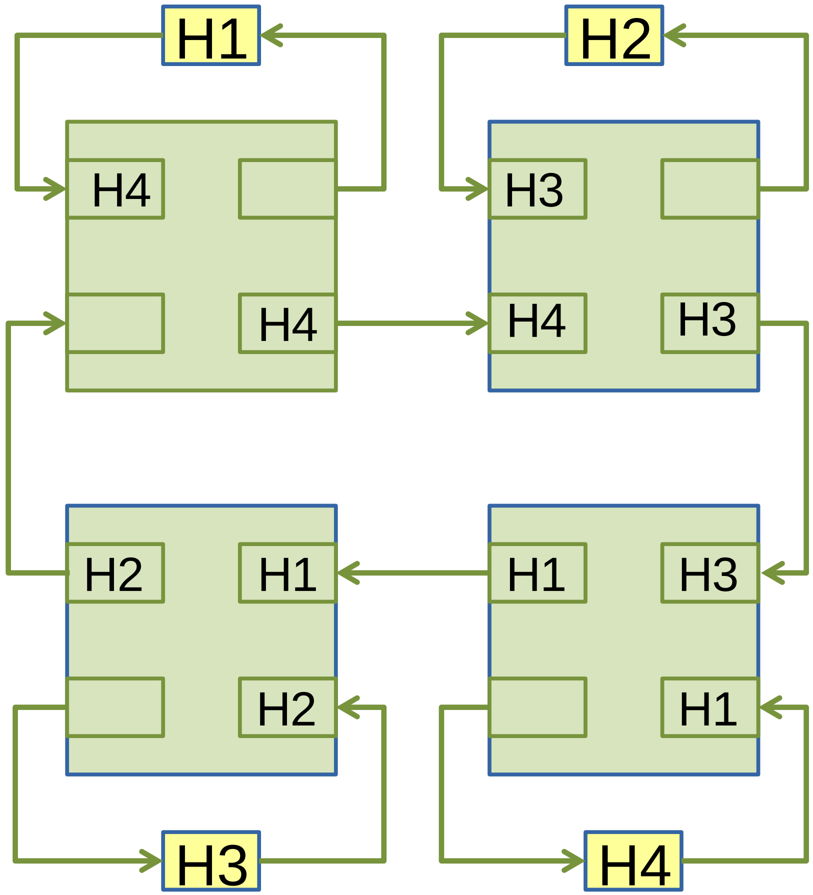
InfiniBand Networks



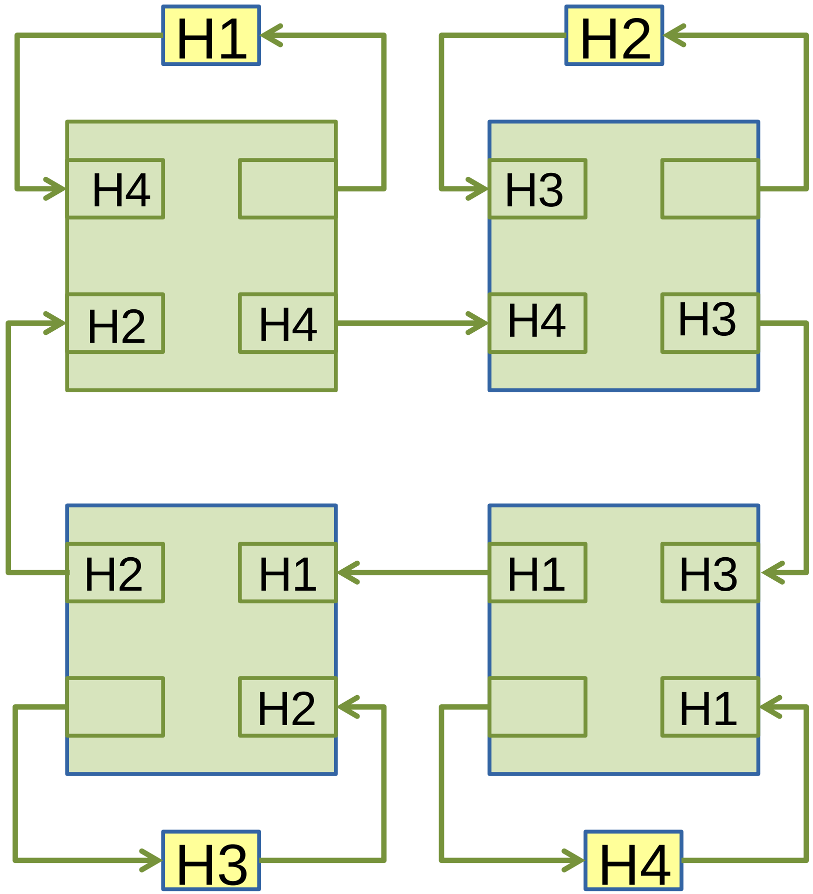
InfiniBand Networks



InfiniBand Networks

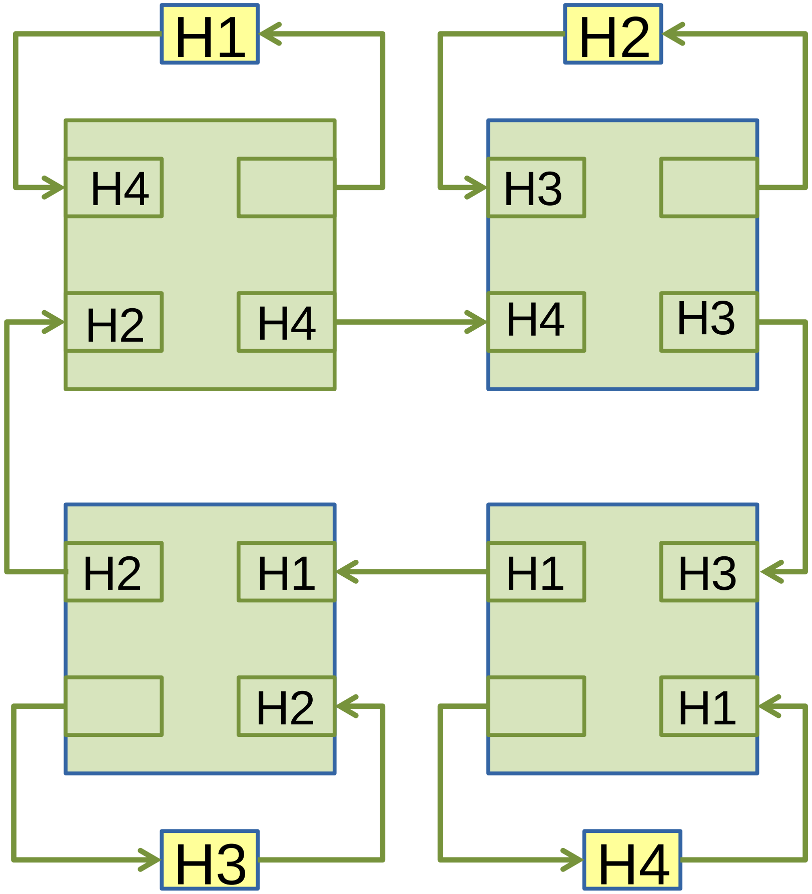


InfiniBand Networks

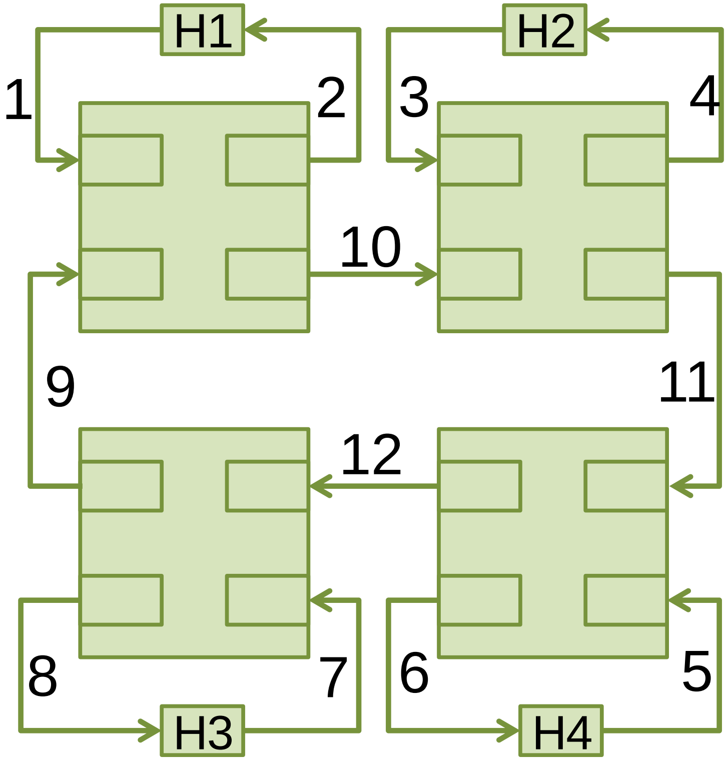


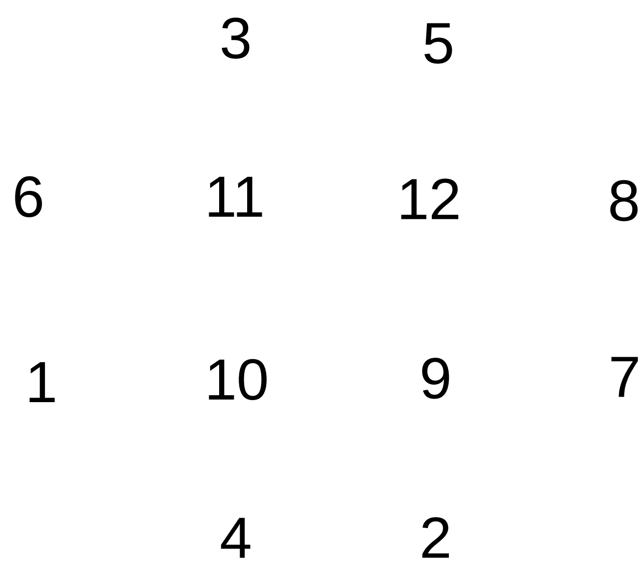
InfiniBand Networks

Nothing can move now - Deadlock

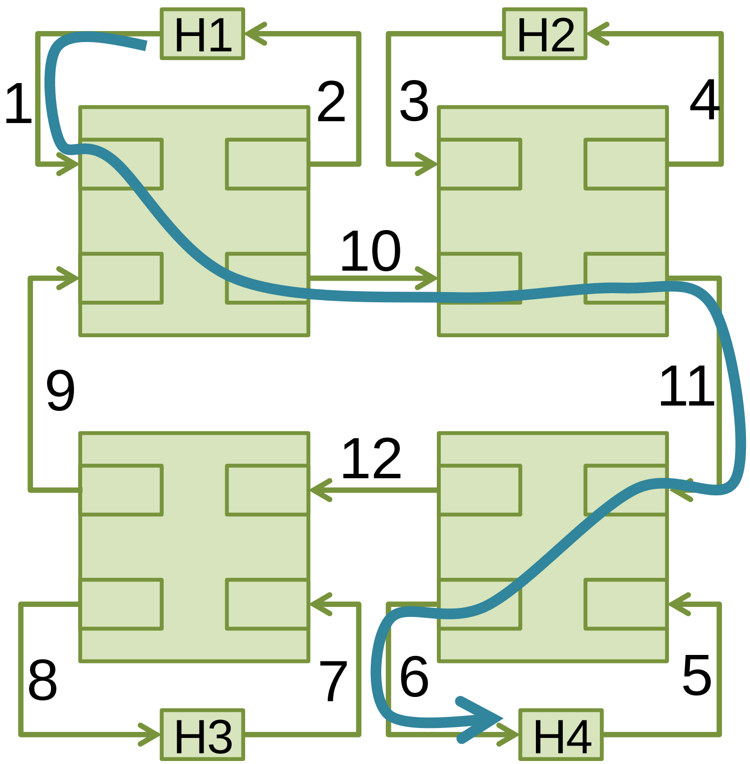


Channel Dependency Graph (CDG)





Channel Dependency Graph (CDG)



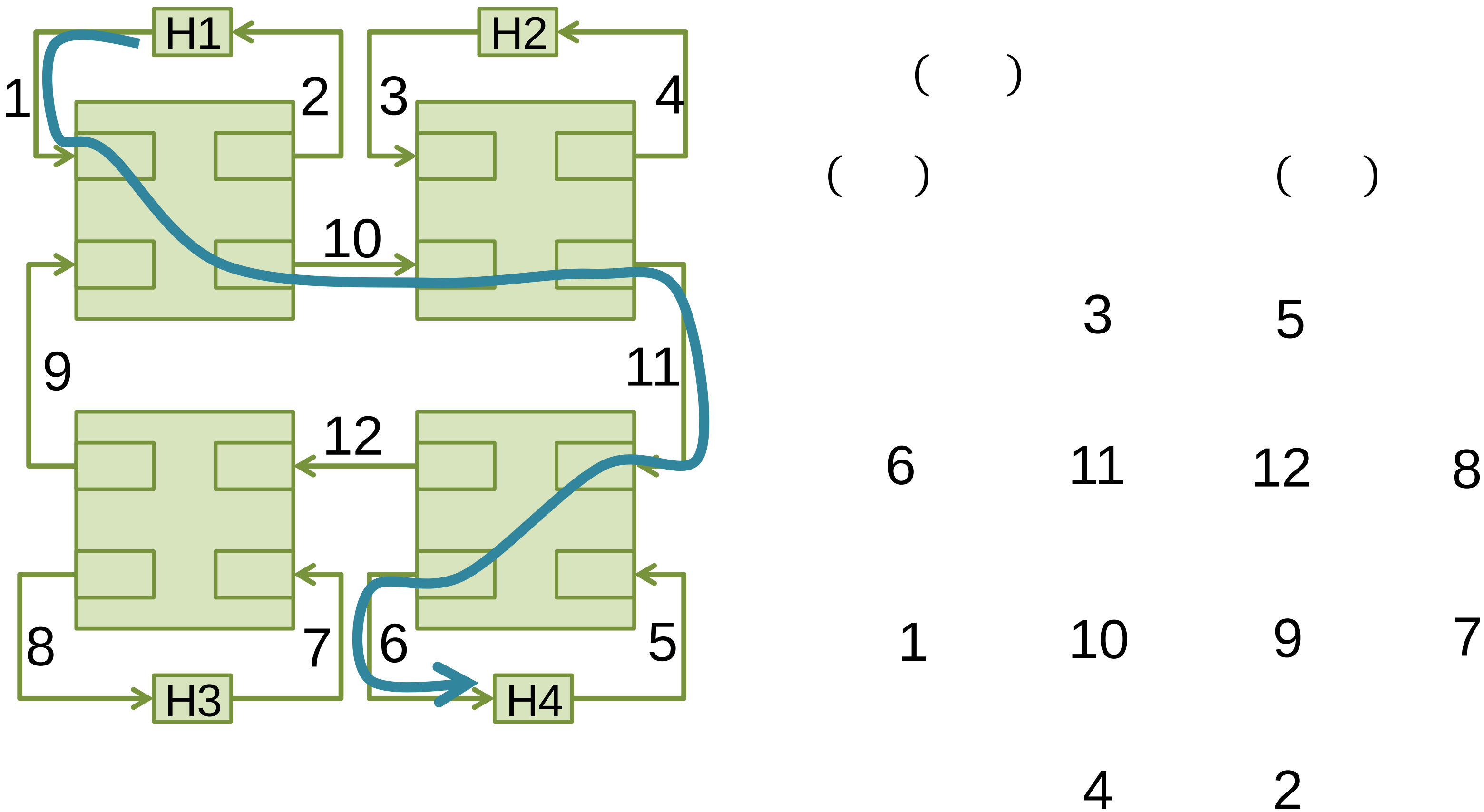
3 5

6 11 12 8

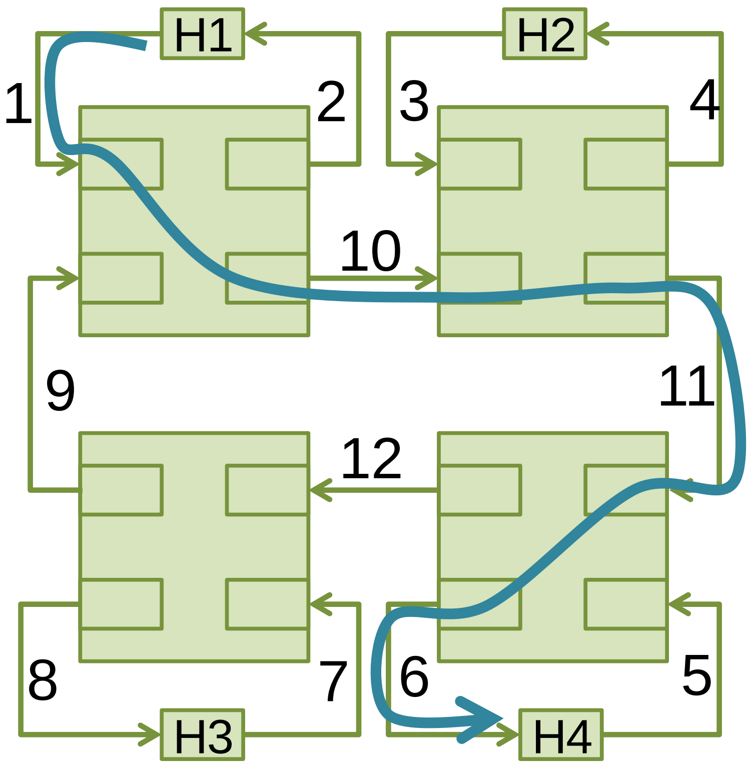
1 10 9 7

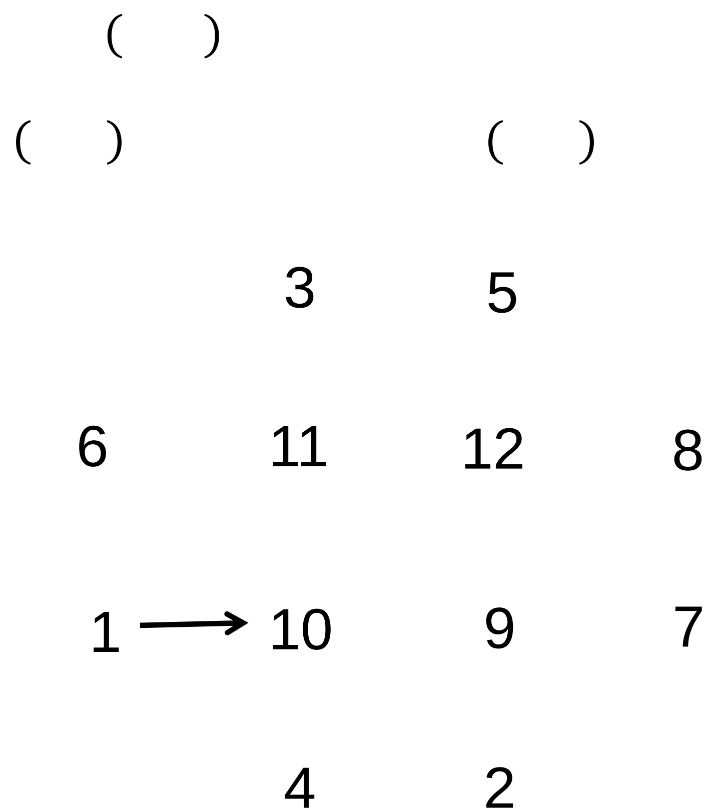
4 2

Channel Dependency Graph (CDG)

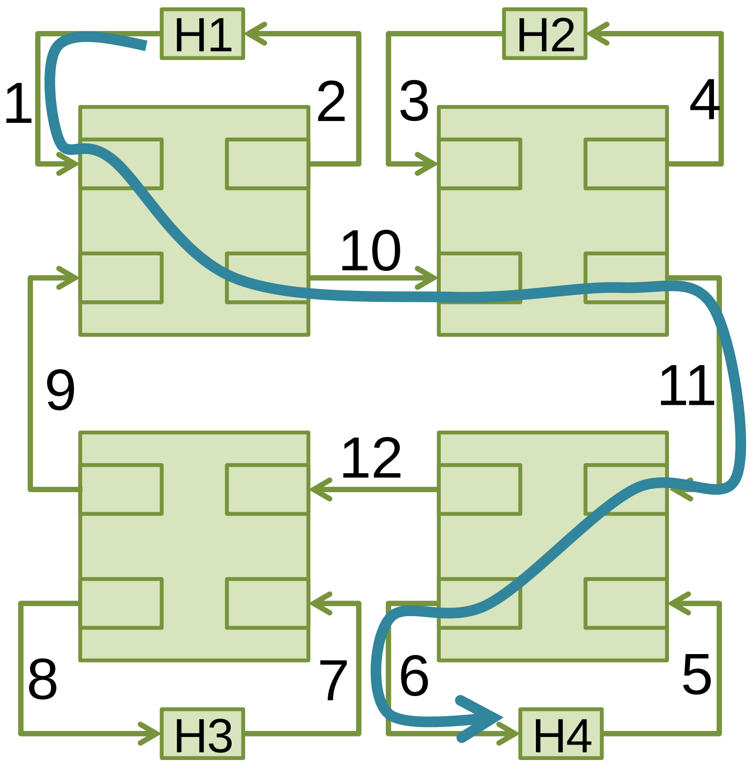


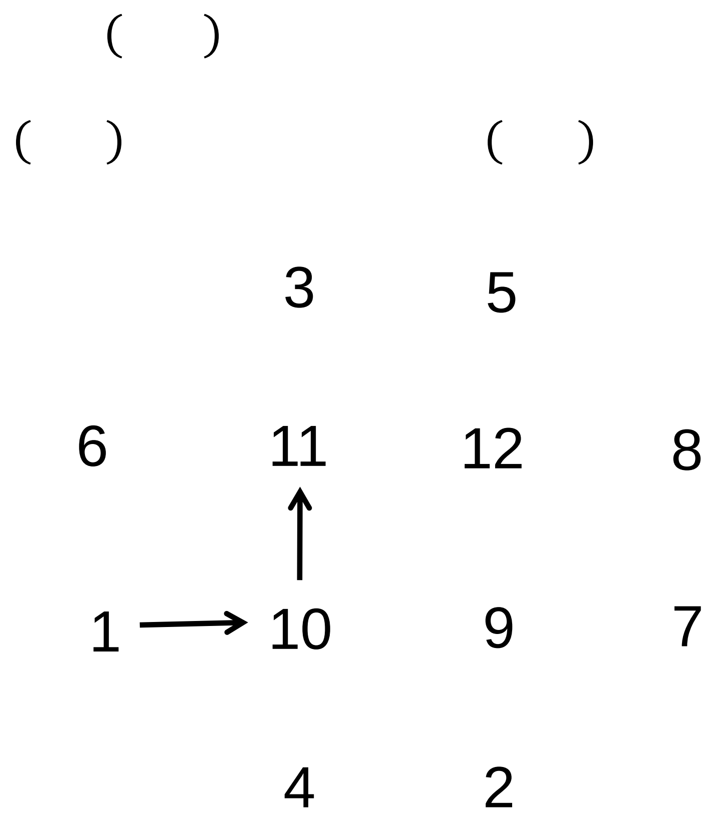
Channel Dependency Graph (CDG)



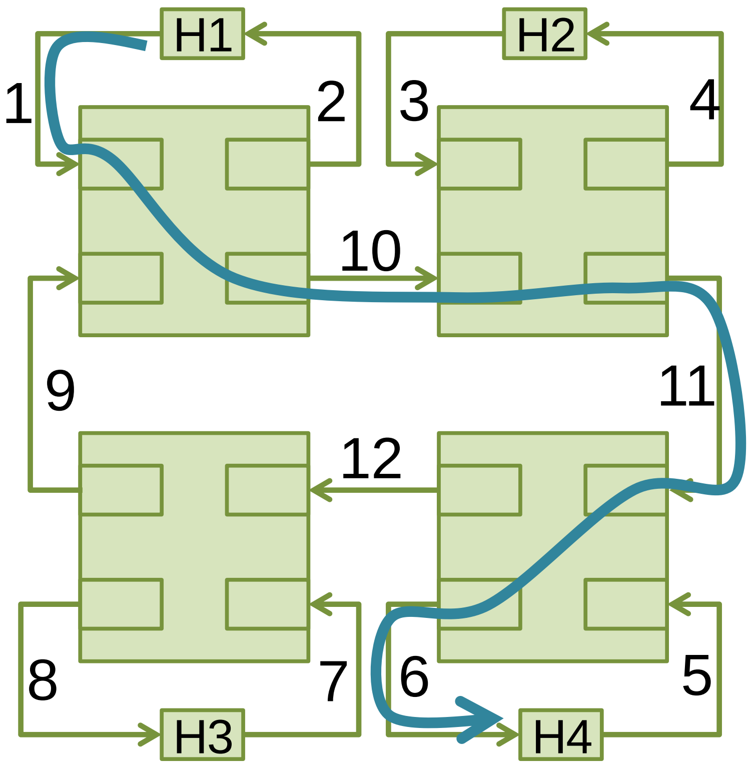


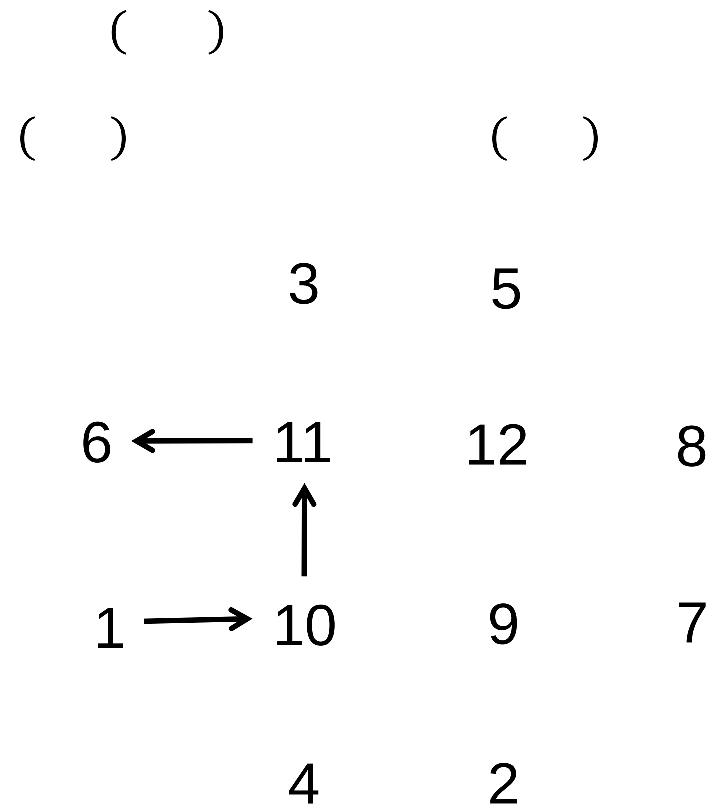
Channel Dependency Graph (CDG)



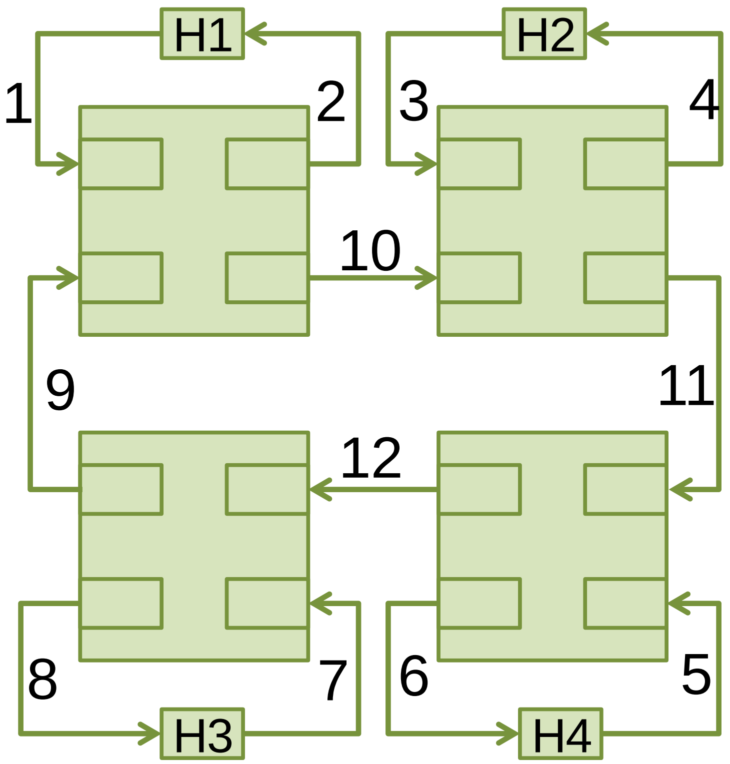


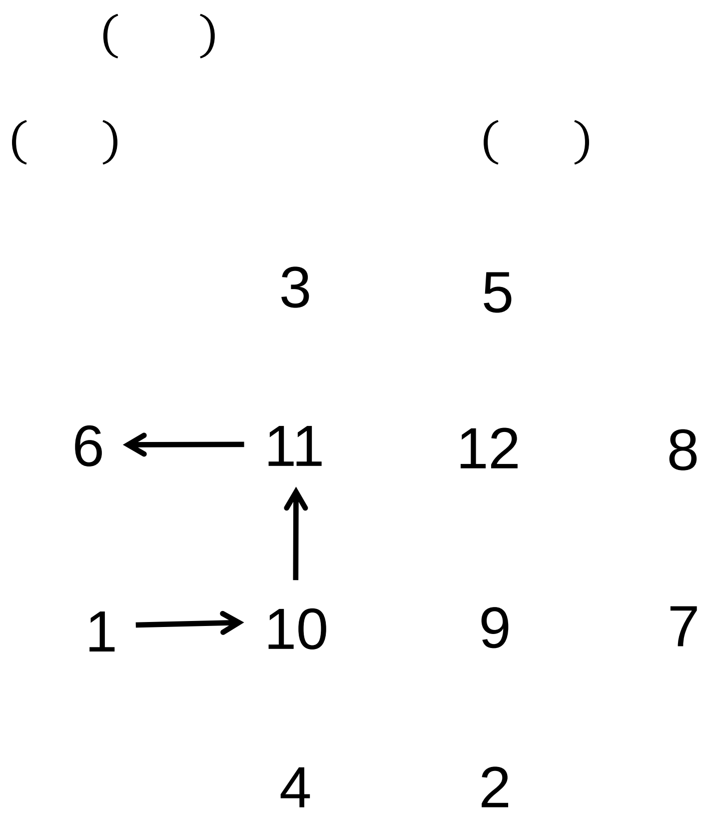
Channel Dependency Graph (CDG)



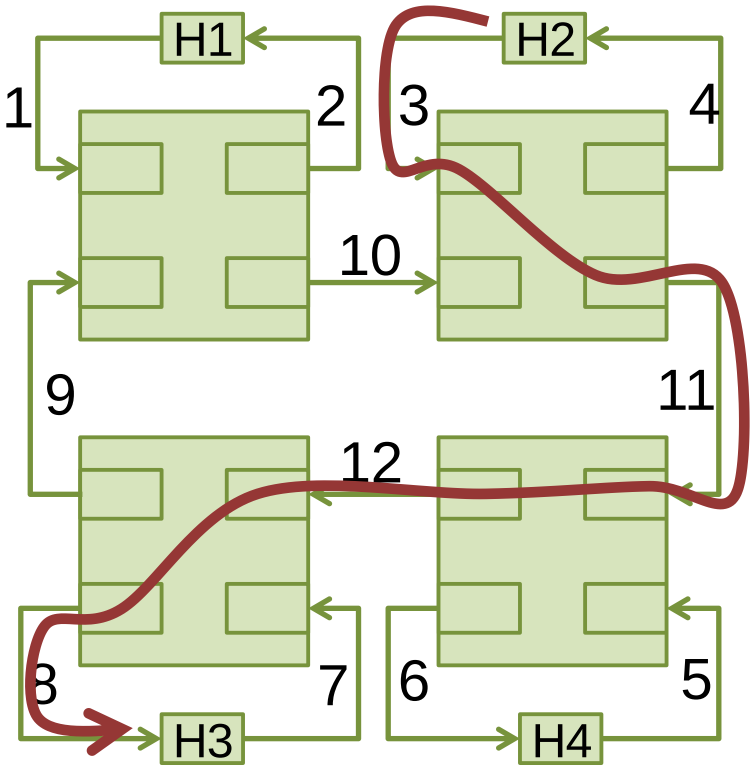


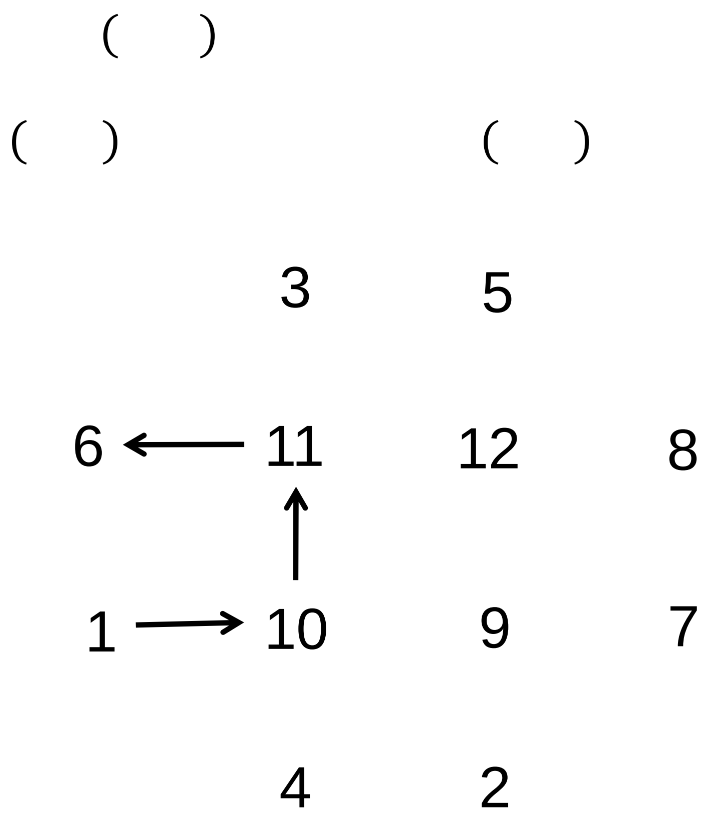
Channel Dependency Graph (CDG)



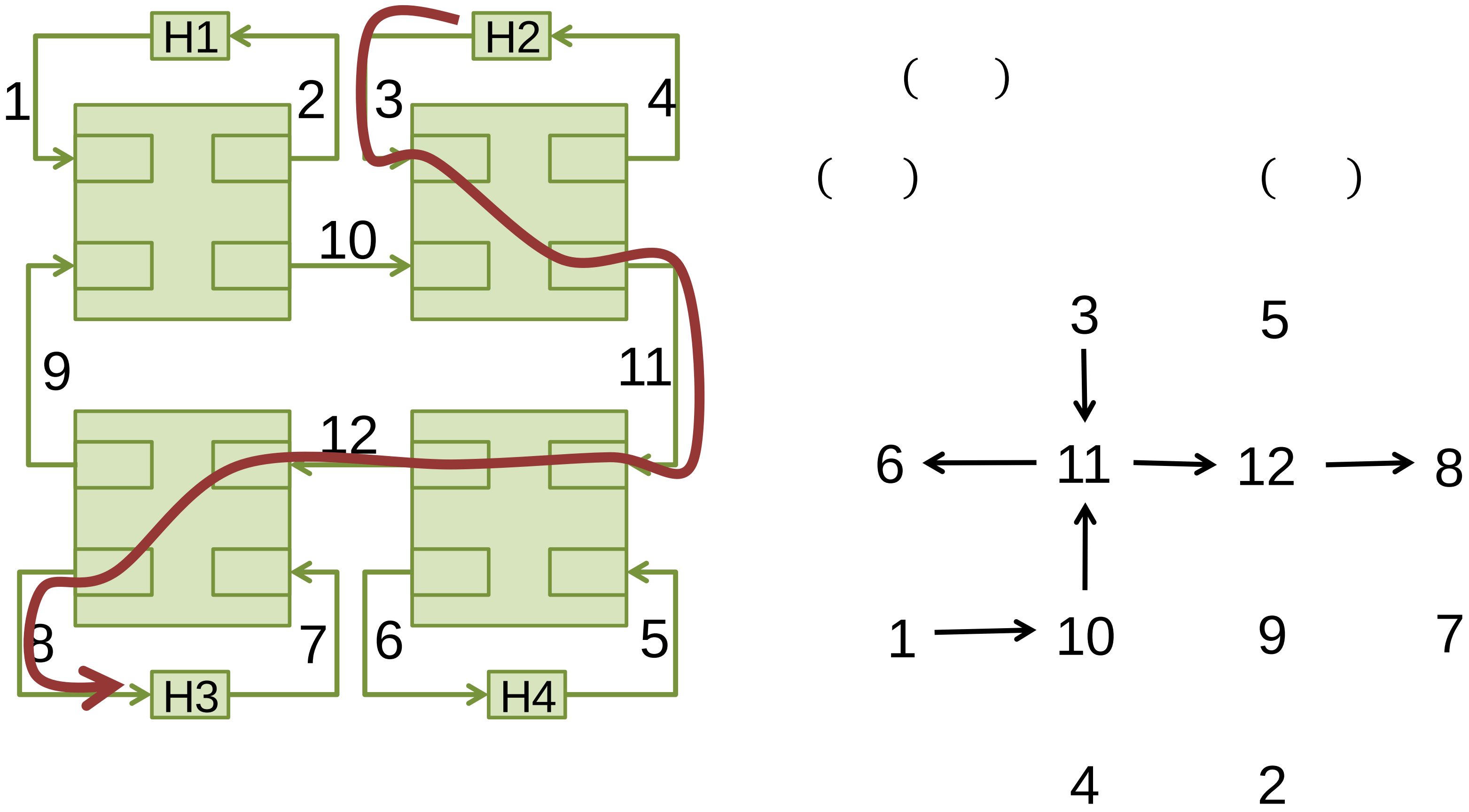


Channel Dependency Graph (CDG)

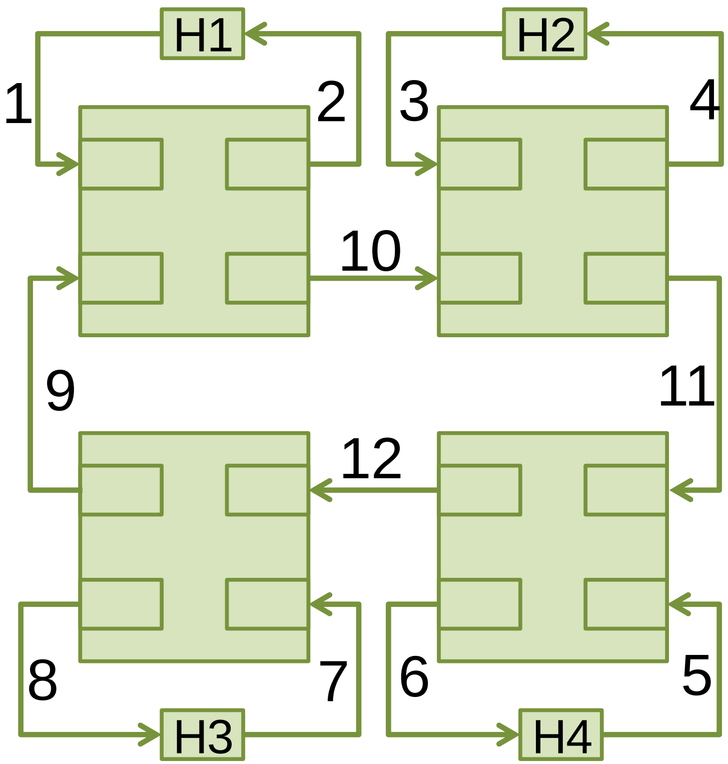


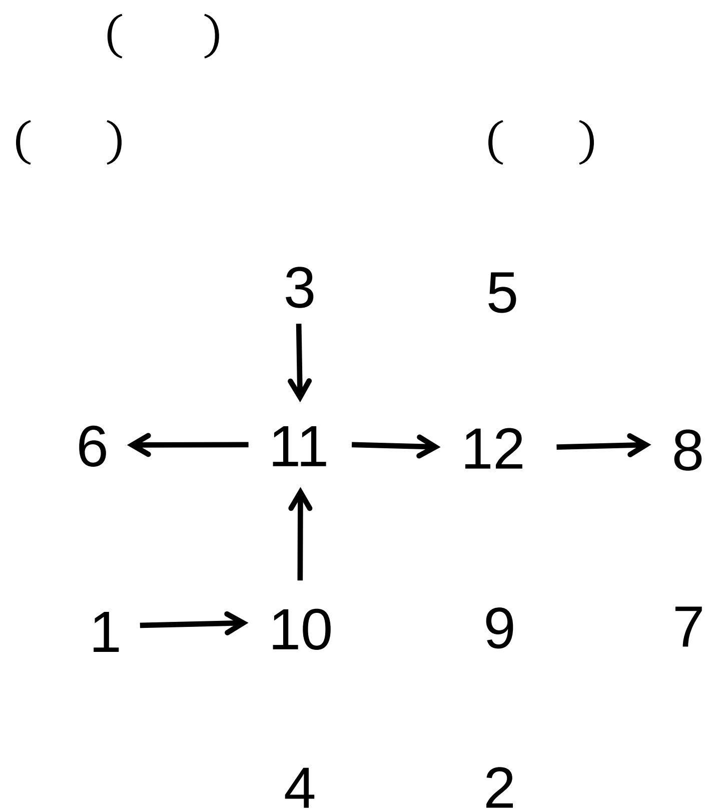


Channel Dependency Graph (CDG)

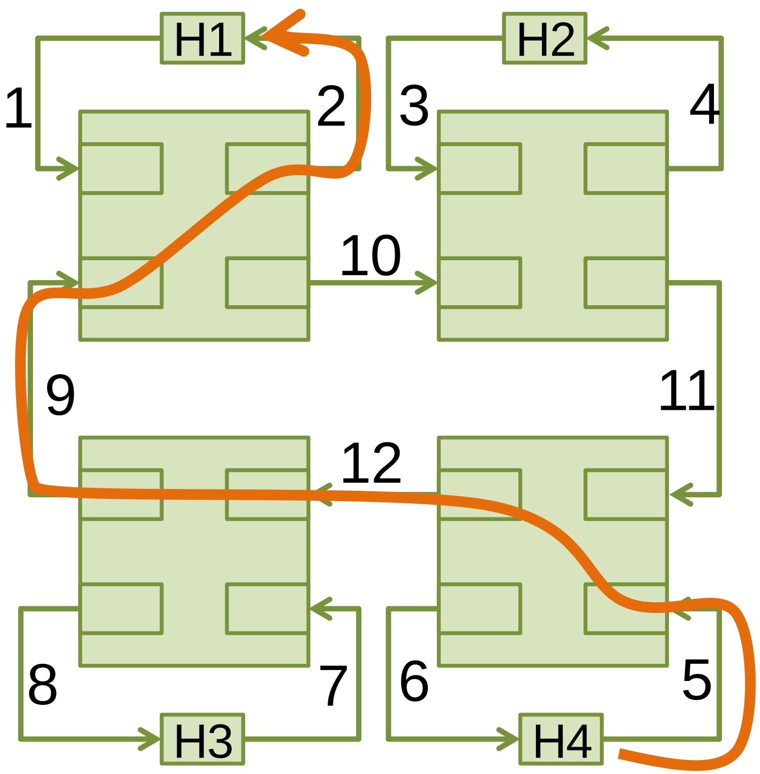


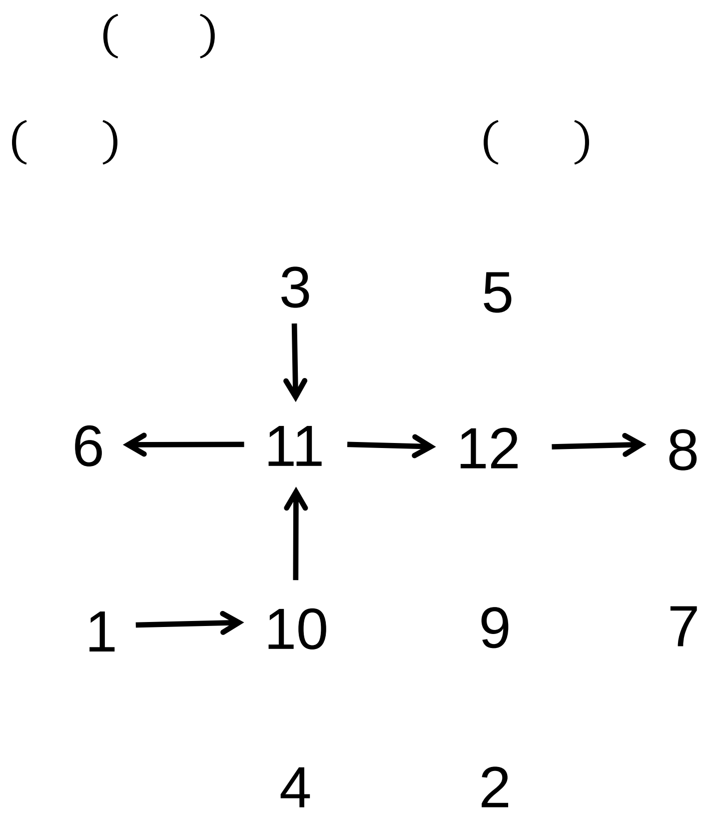
Channel Dependency Graph (CDG)



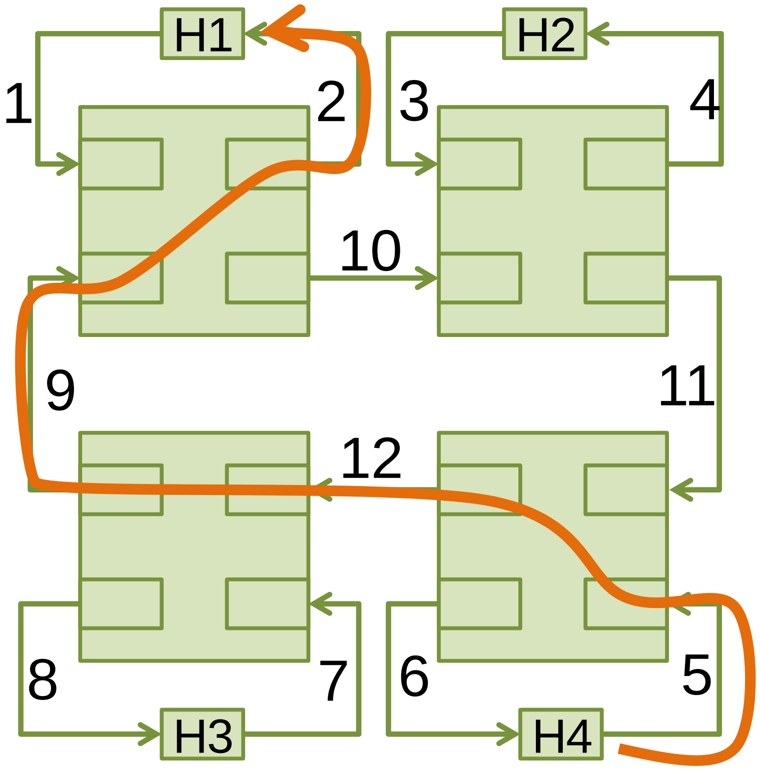


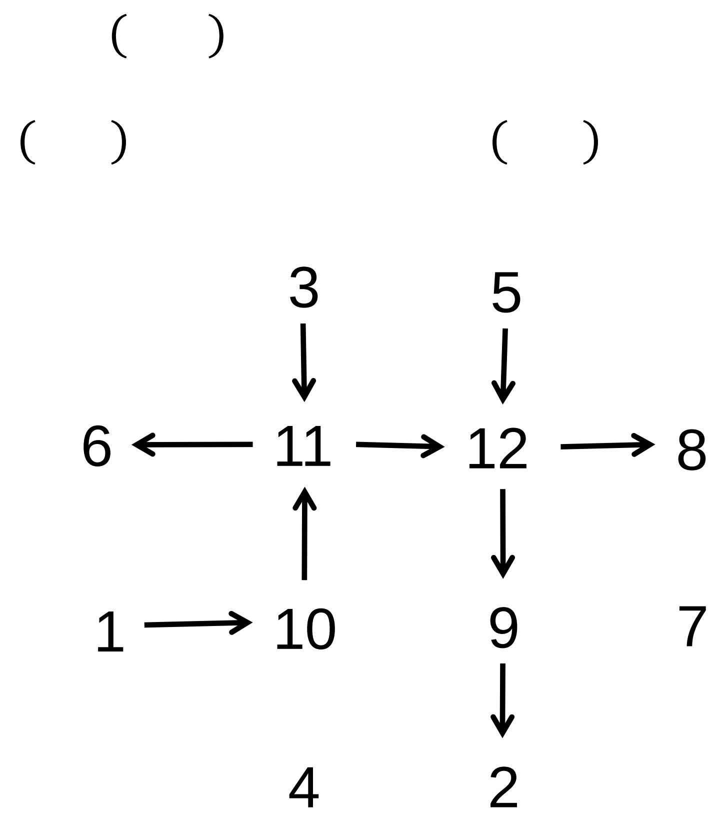
Channel Dependency Graph (CDG)



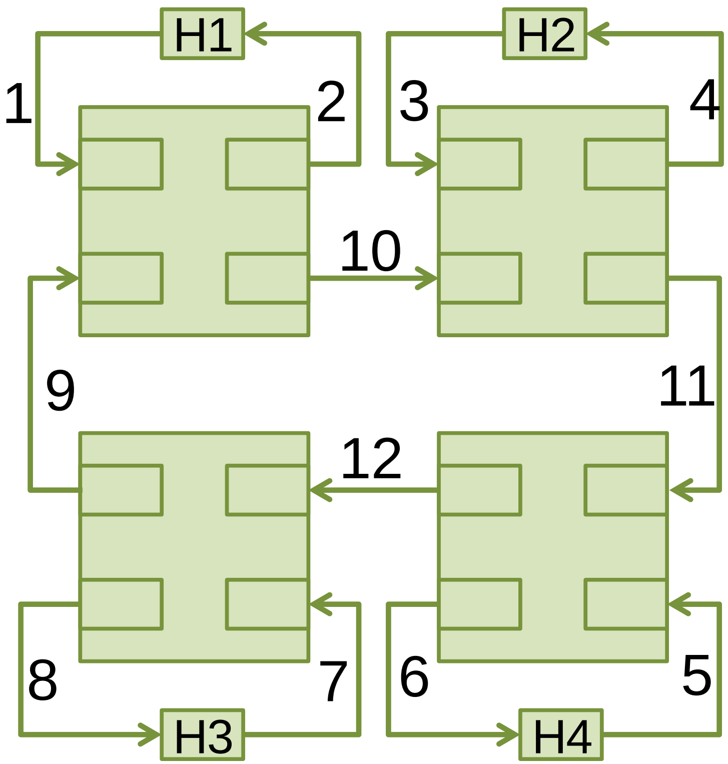


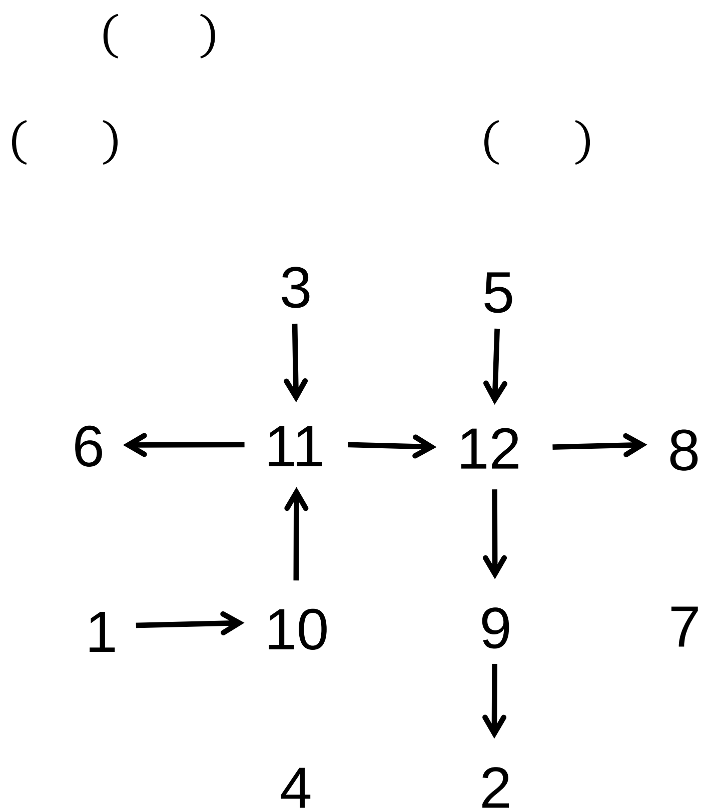
Channel Dependency Graph (CDG)



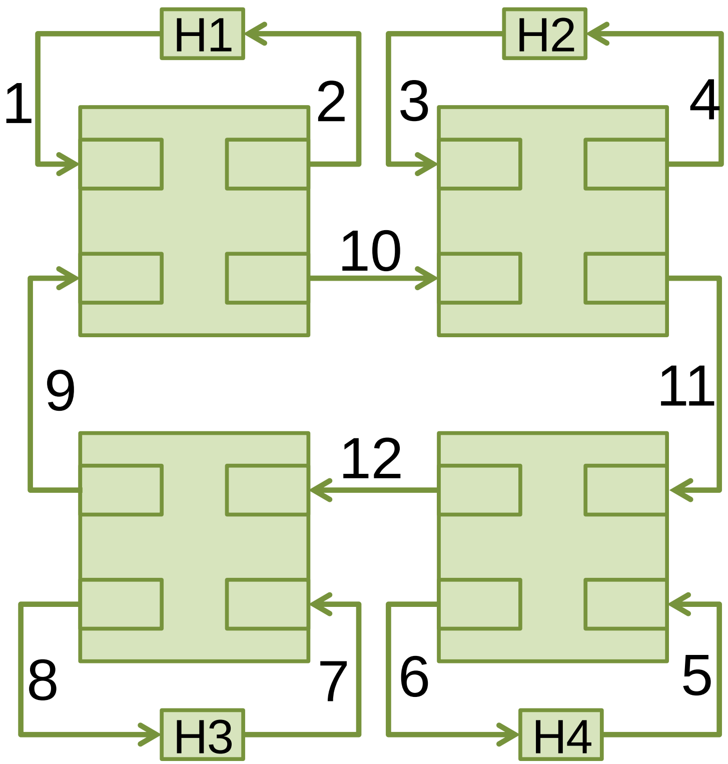


Channel Dependency Graph (CDG)



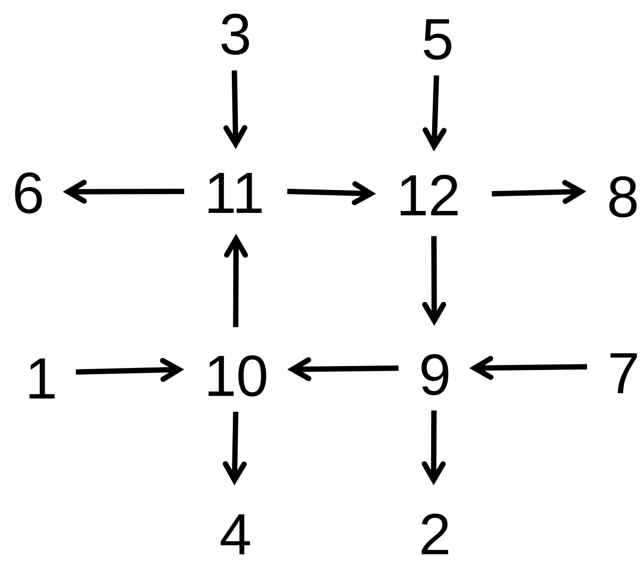


Channel Dependency Graph (CDG)

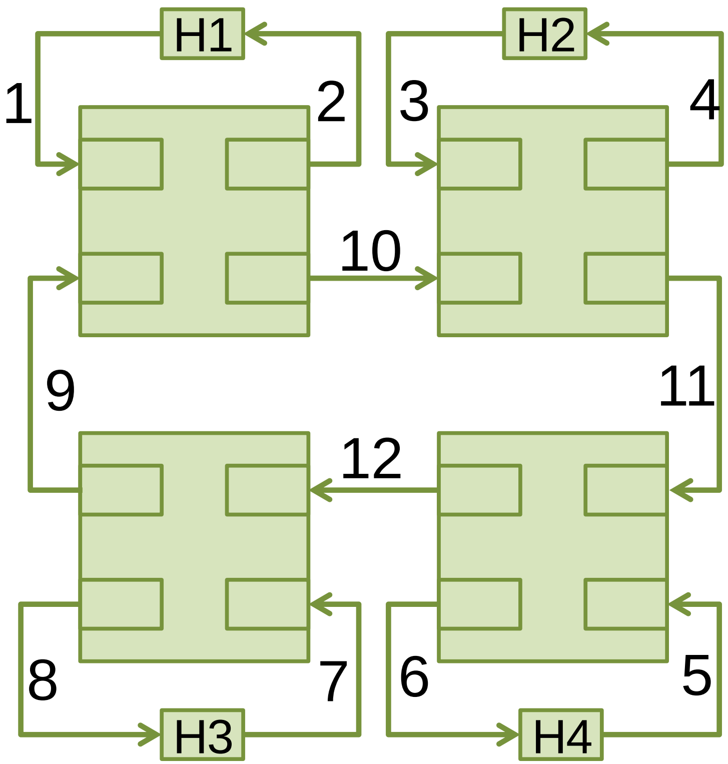


( )

( ) ( )

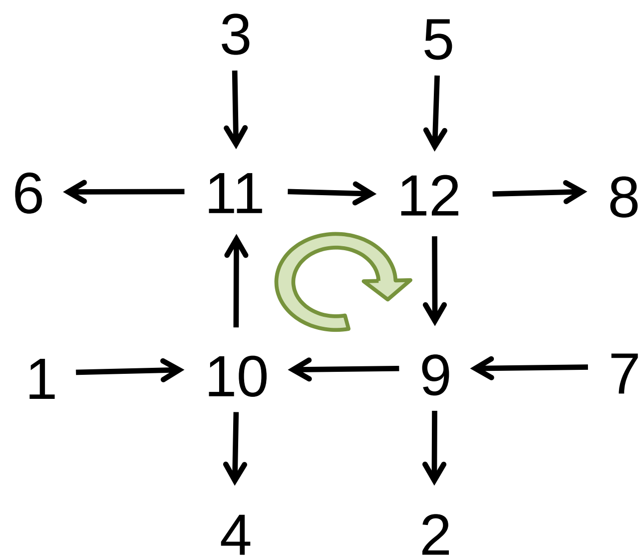


Channel Dependency Graph (CDG)

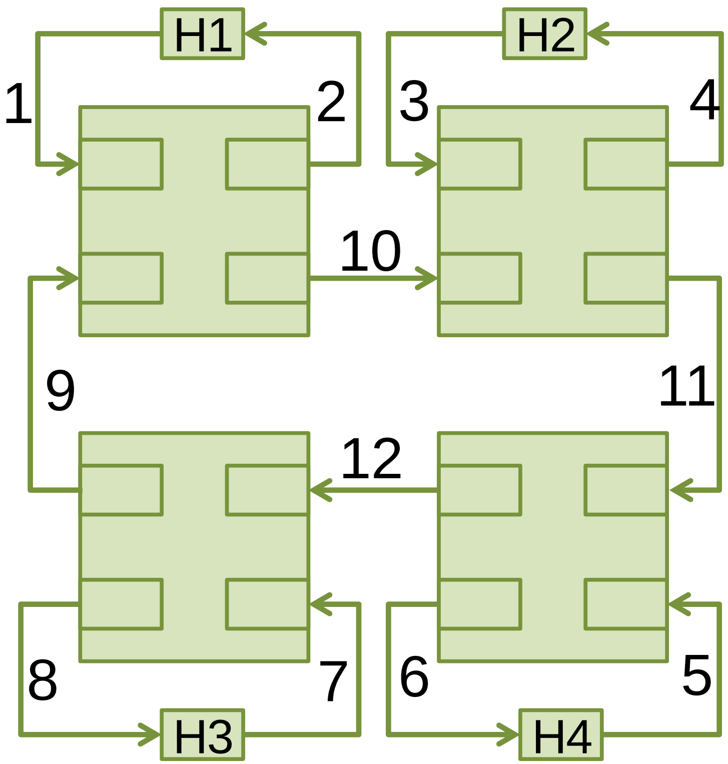


( )

( ) ( )

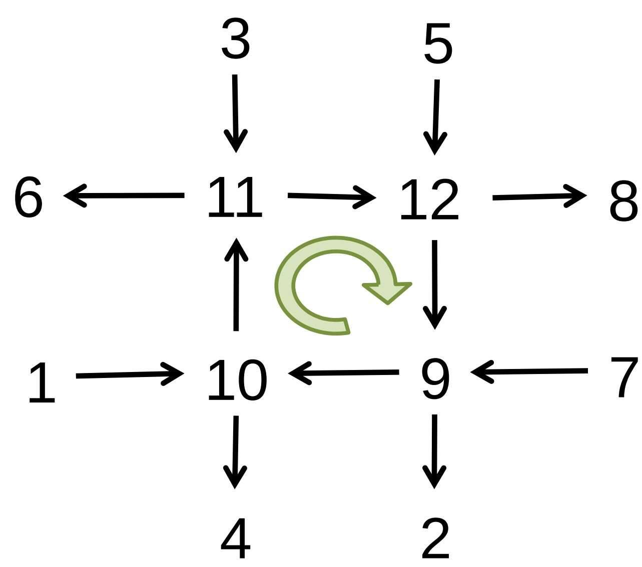


Channel Dependency Graph (CDG)



( )

( ) ( )

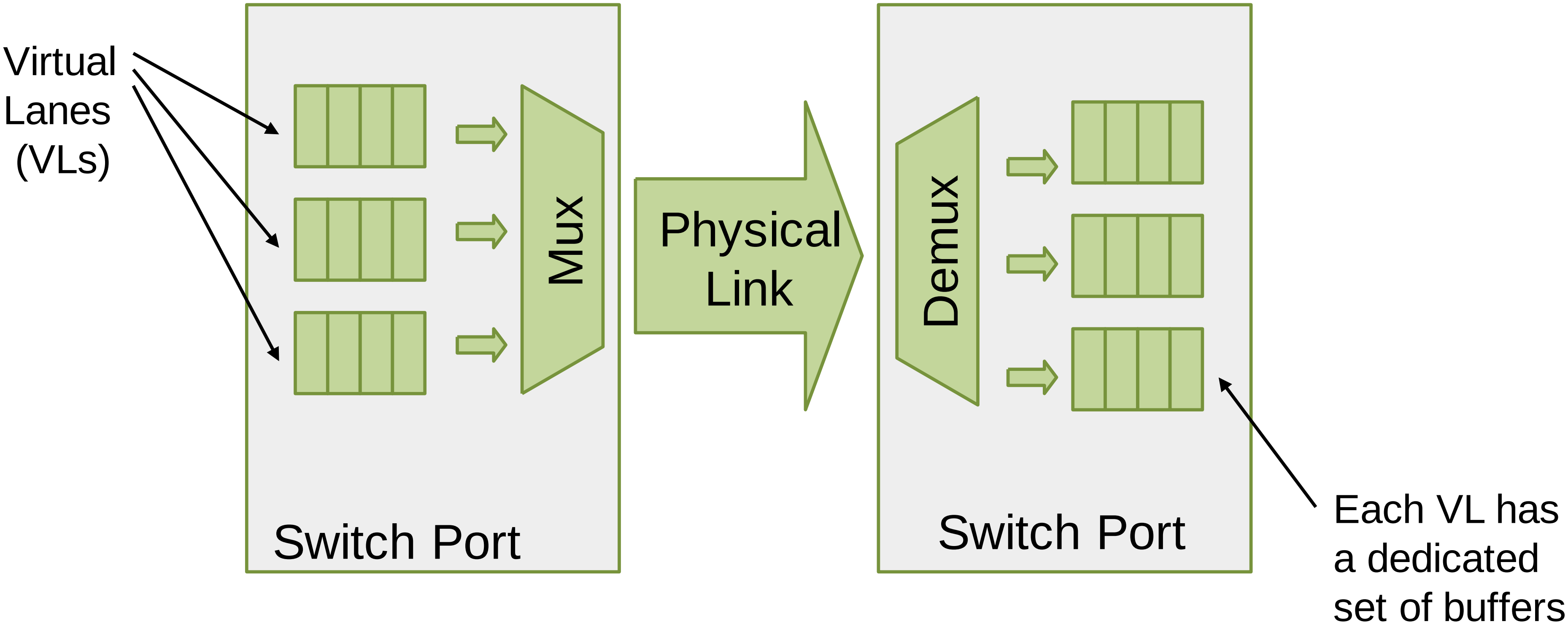


If the CDG is acyclic, no deadlocks can occur.

Dally/Seitz, Deadlock-free message routing in milti-processor interconnection networks, 1987

Dealing with cycles in the CDG

- Ignore the problem, rely on timeouts / retransmissions (MinHop)
- Restrict routing such that no cycles can form, i.e. (Up/Down)
- Use Virtual Lanes (DF-SSSP, LASH)



Virtual Lanes in InfiniBand

How does IB implement VLs:

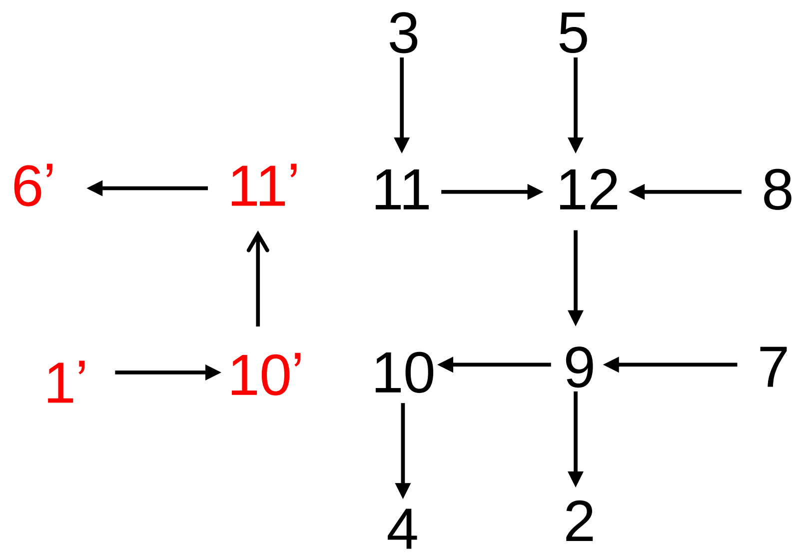
- ***The sender sets a Service Level (SL) in the packet header***
  ( )
- ***Each switch has a SL-to-VL mapping table which maps (input channel, output channel, SL) to the output VL:*** ( )
- ***denotes physical channel a using VL i***

The combination of those relation lets us define a virtual routing function ( )

Switches cannot access input VL, or change the SL!

Utilizing Virtual Lanes - Layering

- **Each path from source to destination uses one VL, each VL forms a layer. If the CDG of each layer is acyclic, there are no cycles.**

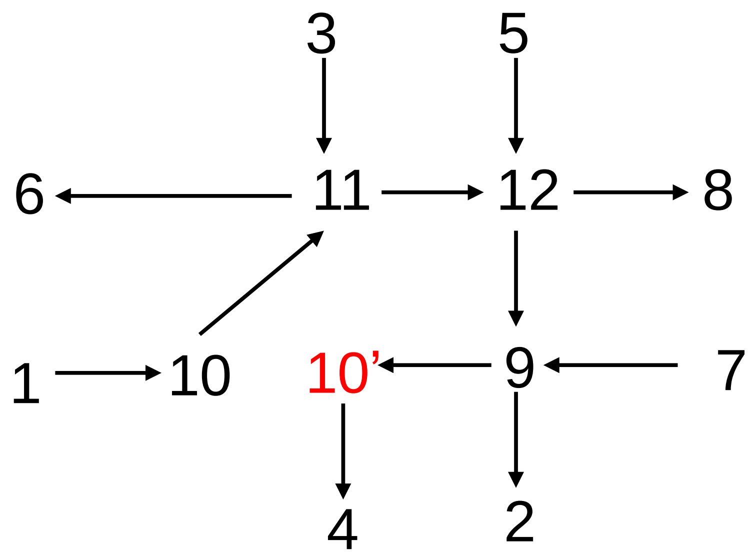


Layering moves three edges! We can do better!

Which paths to move to minimize #VLs? NP-complete!

Utilizing Virtual Lanes – VL Hopping

- **IB allows changing the VL within switches → needs less resources to break cycles**

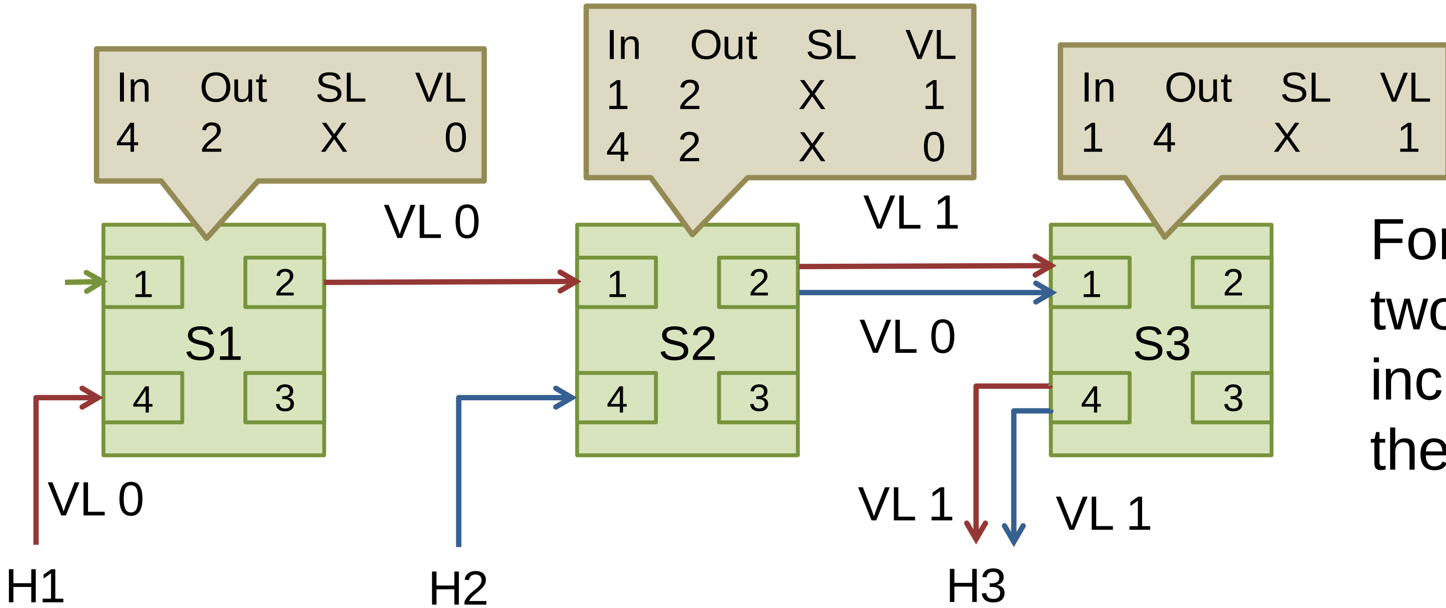


Which paths to move to minimize #VLs? NP-complete?

Incrementing the VL

- **If we increment the VL at every hop, the CDG is acyclic:**
  ( ) ***thus***
  ( ) ***thus we can sort D topologically, since < is a total order, therefore D is acyclic.***

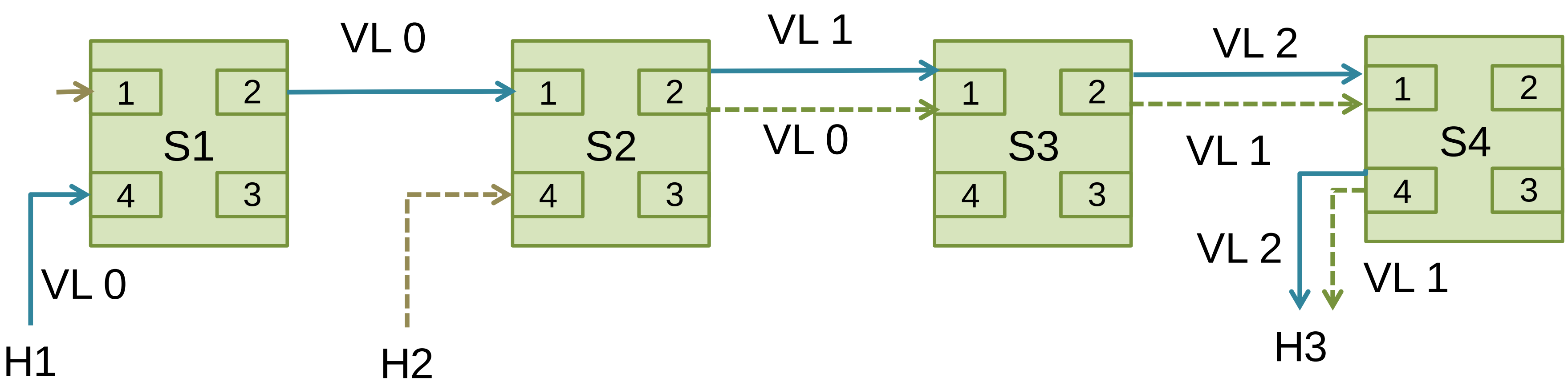
- **The number of VLs used = number of hops, good for low-diameter topologies!**



For diameter-two networks, incrementing the VL is trivial.

Incrementing the VL (diameter > 2)

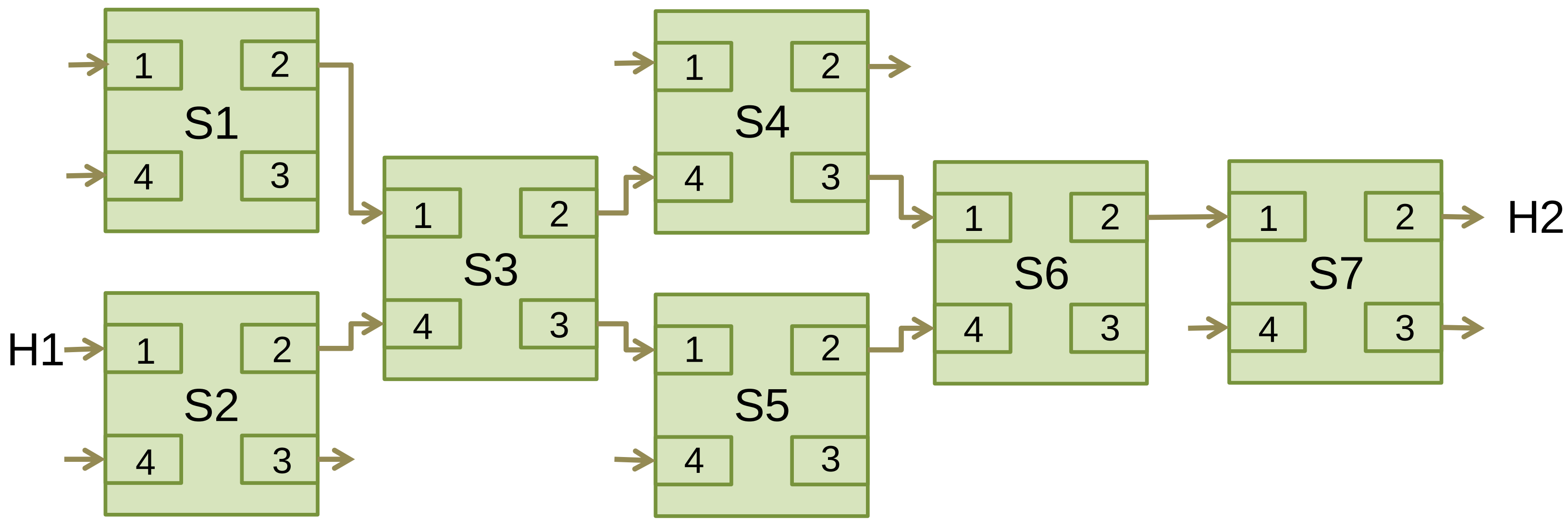
Same input/output
port, but different VL!



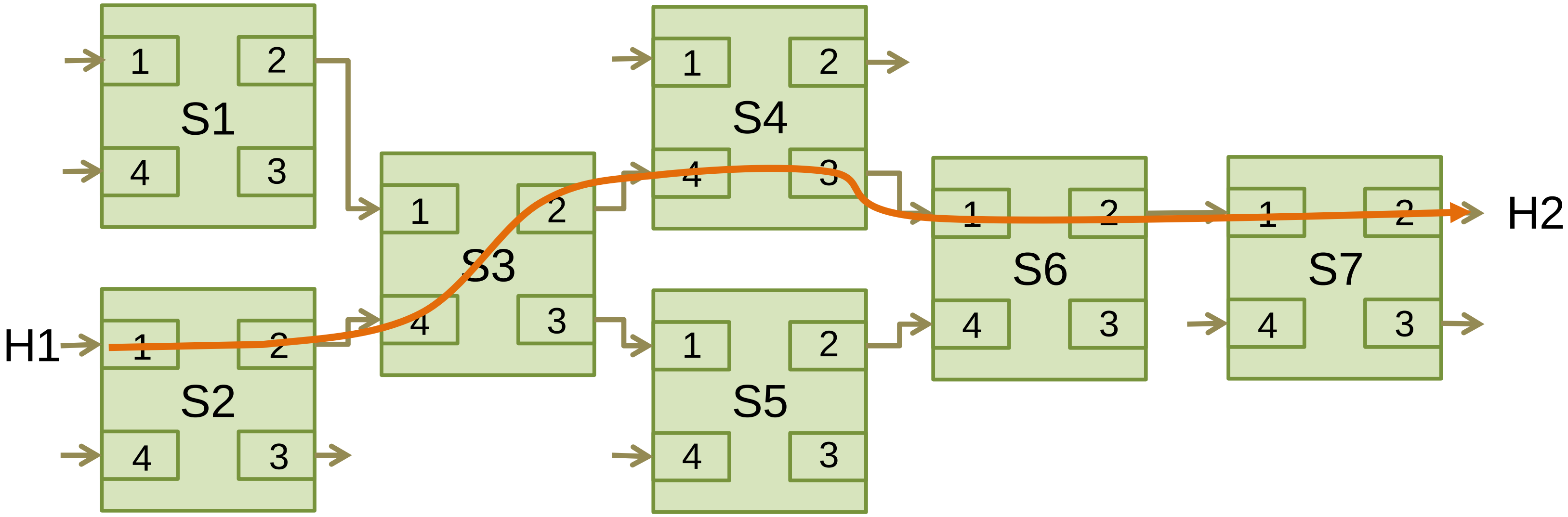
Leverage the full power of SL-to-VL mapping and use different SLs for different source/destination pairs!

- **Different SL for each pair? →16 SLs in IB, only four leaf switches!**

Incrementing the VL (diameter > 2)

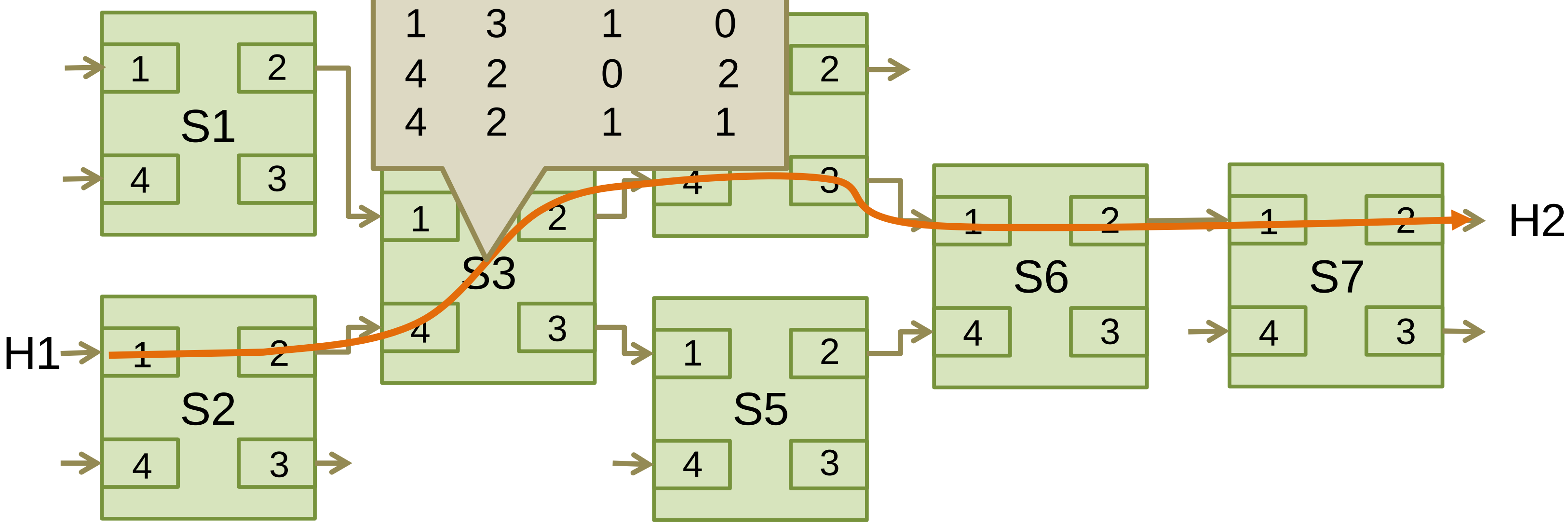


Incrementing the VL (diameter > 2)



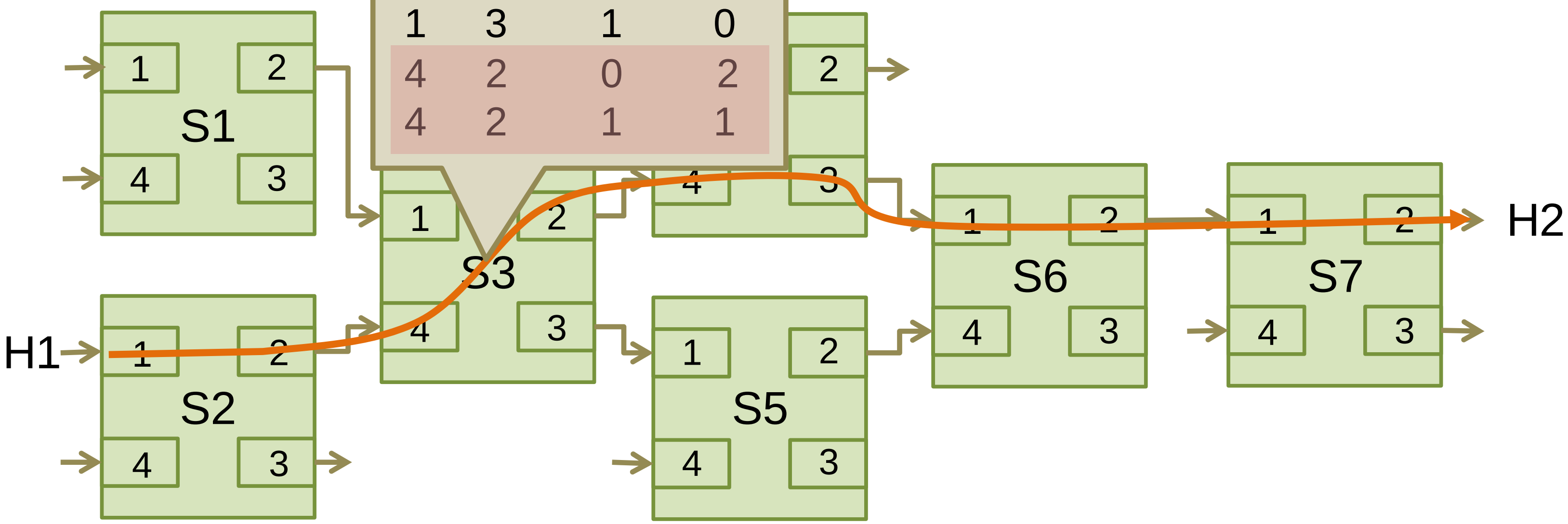
Incrementing the VL (diameter > 2)

In	Out	SL	VL
1	2	0	0
1	3	1	0
4	2	0	2
4	2	1	1



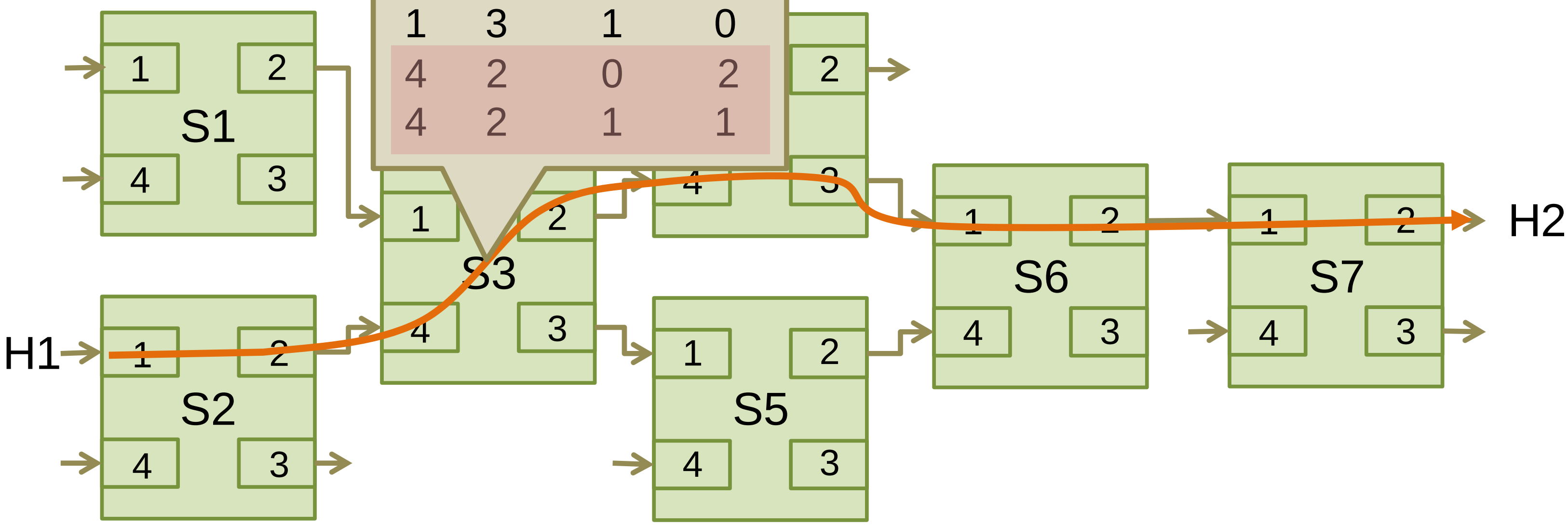
Incrementing the VL (diameter > 2)

In	Out	SL	VL
1	2	0	0
1	3	1	0
4	2	0	2
4	2	1	1



Incrementing the VL (diameter > 2)

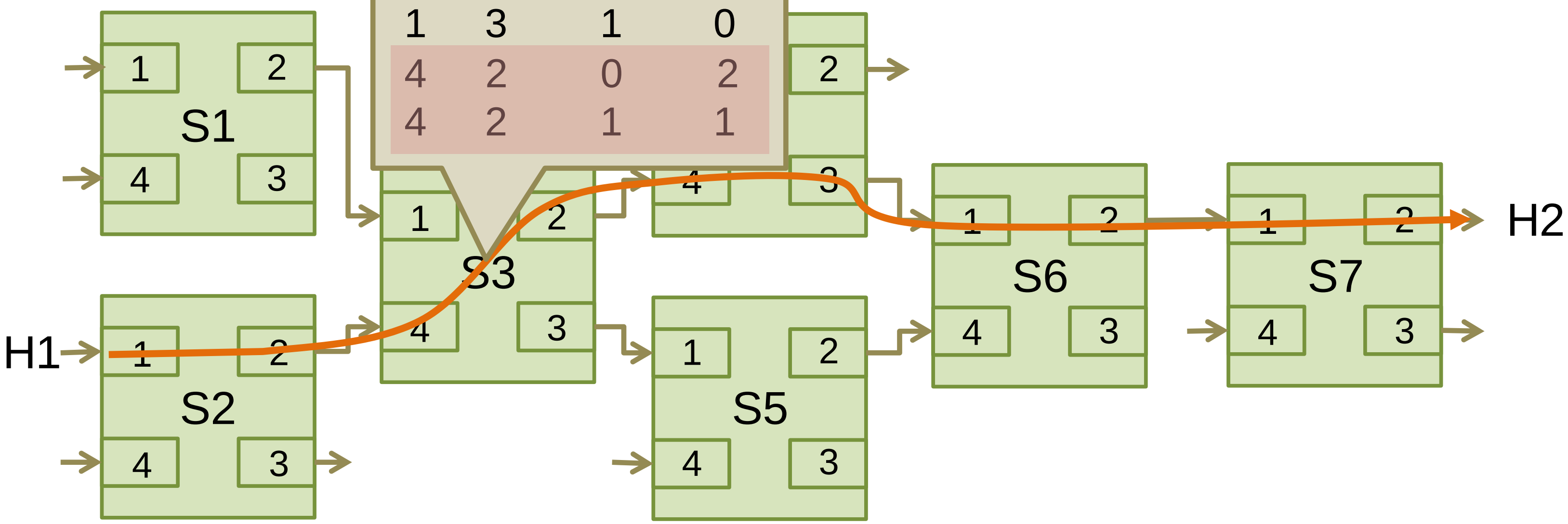
In	Out	SL	VL
1	2	0	0
1	3	1	0
4	2	0	2
4	2	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1			

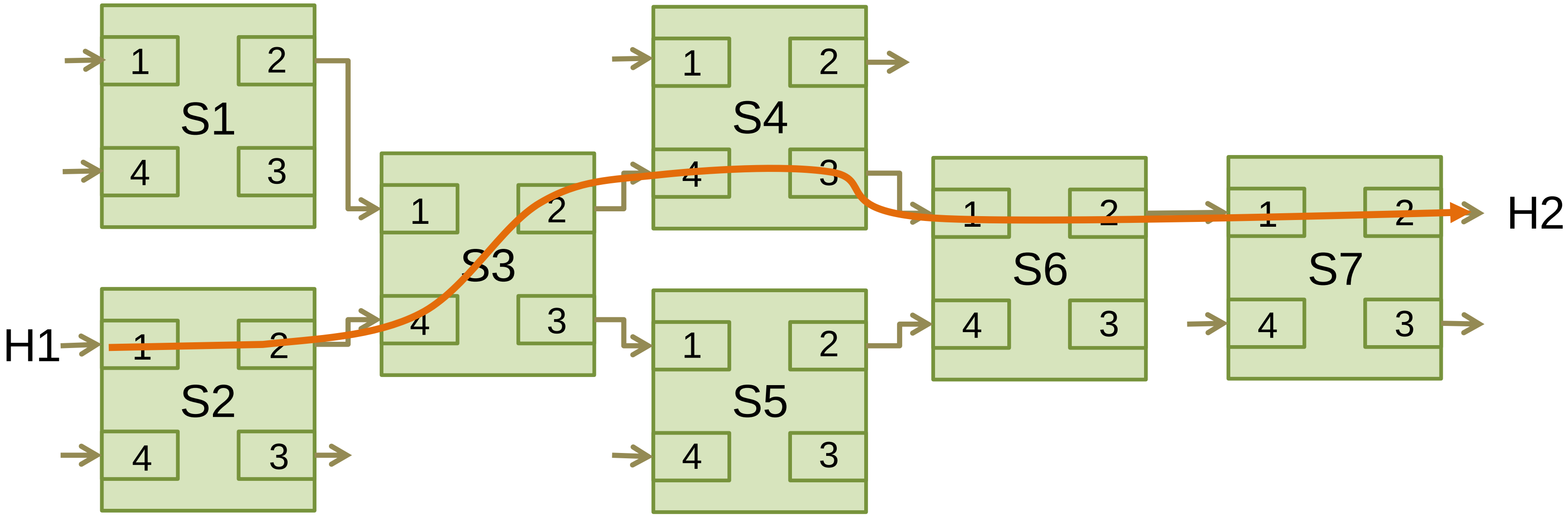
Incrementing the VL (diameter > 2)

In	Out	SL	VL
1	2	0	0
1	3	1	0
4	2	0	2
4	2	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	

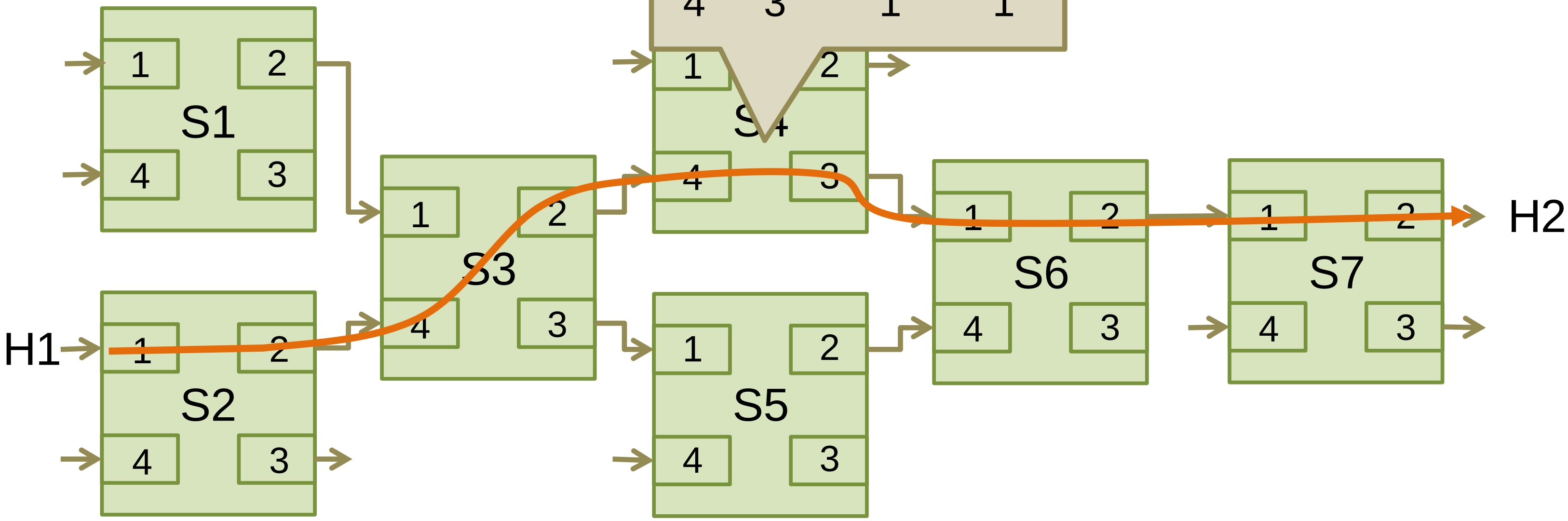
Incrementing the VL (diameter > 2)



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	

Incrementing the … r > 2)

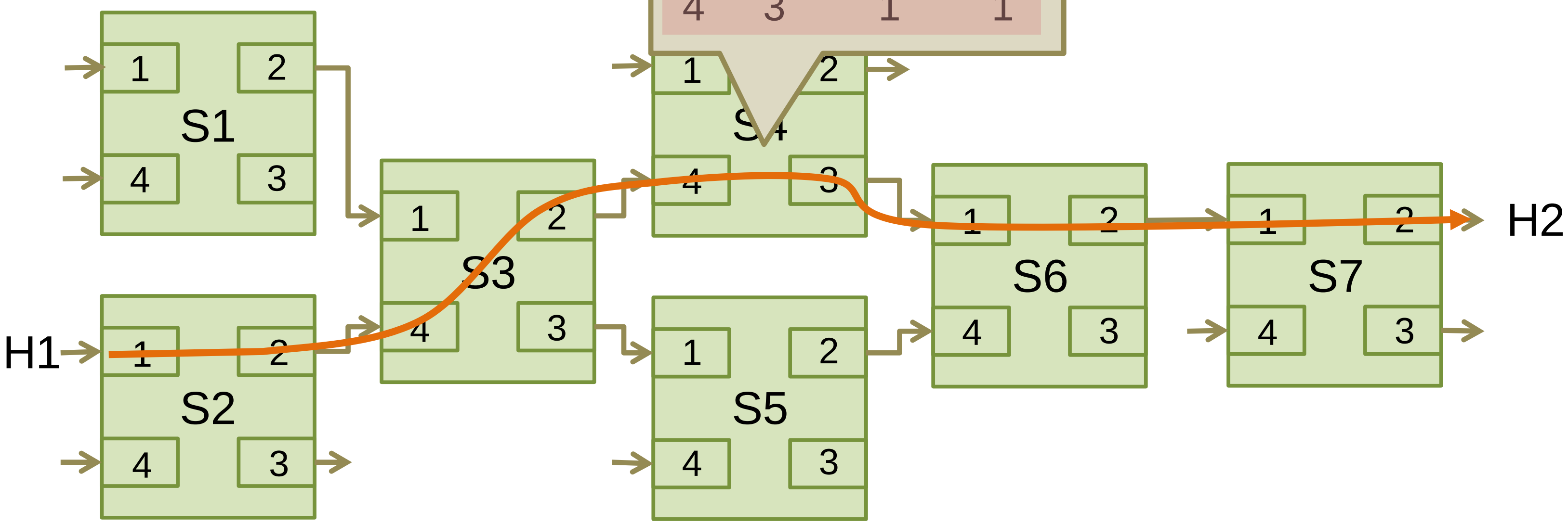
In	Out	SL	VL
1	2	0	0
4	2	1	3
4	3	0	1
4	3	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	

Incrementing the … r > 2)

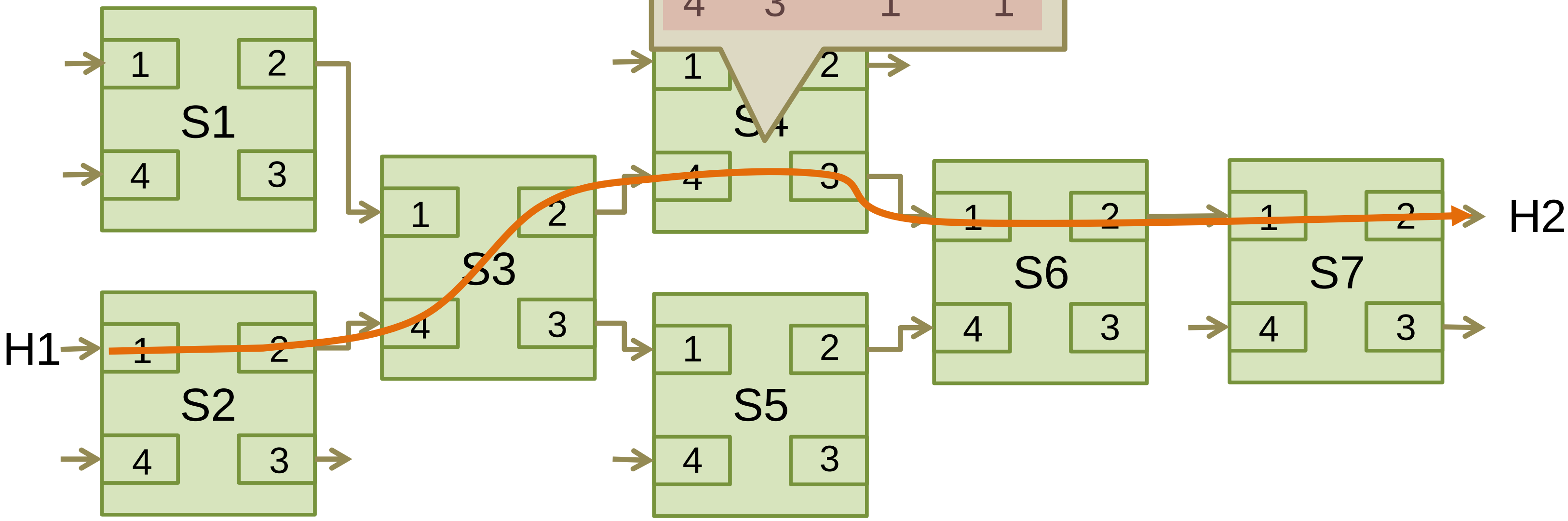
In	Out	SL	VL
1	2	0	0
4	2	1	3
4	3	0	1
4	3	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	

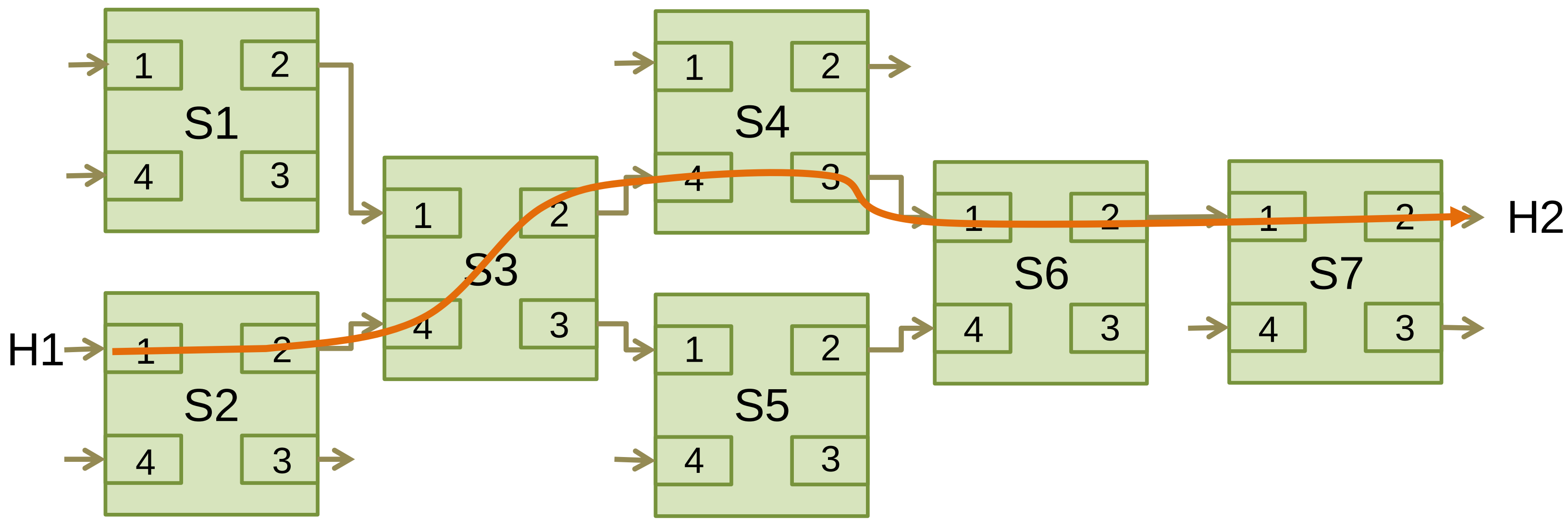
Incrementing the ... r > 2)

In	Out	SL	VL
1	2	0	0
4	2	1	3
4	3	0	1
4	3	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	

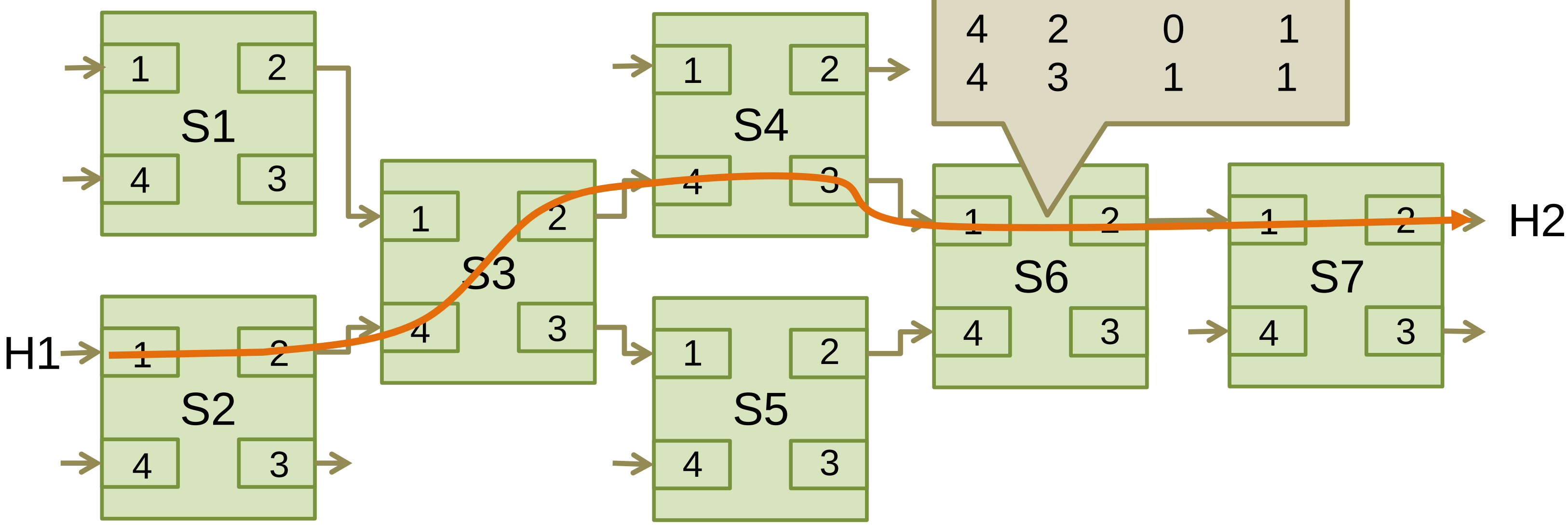
Incrementing the VL (diameter > 2)



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	

Incrementing the VL (dian

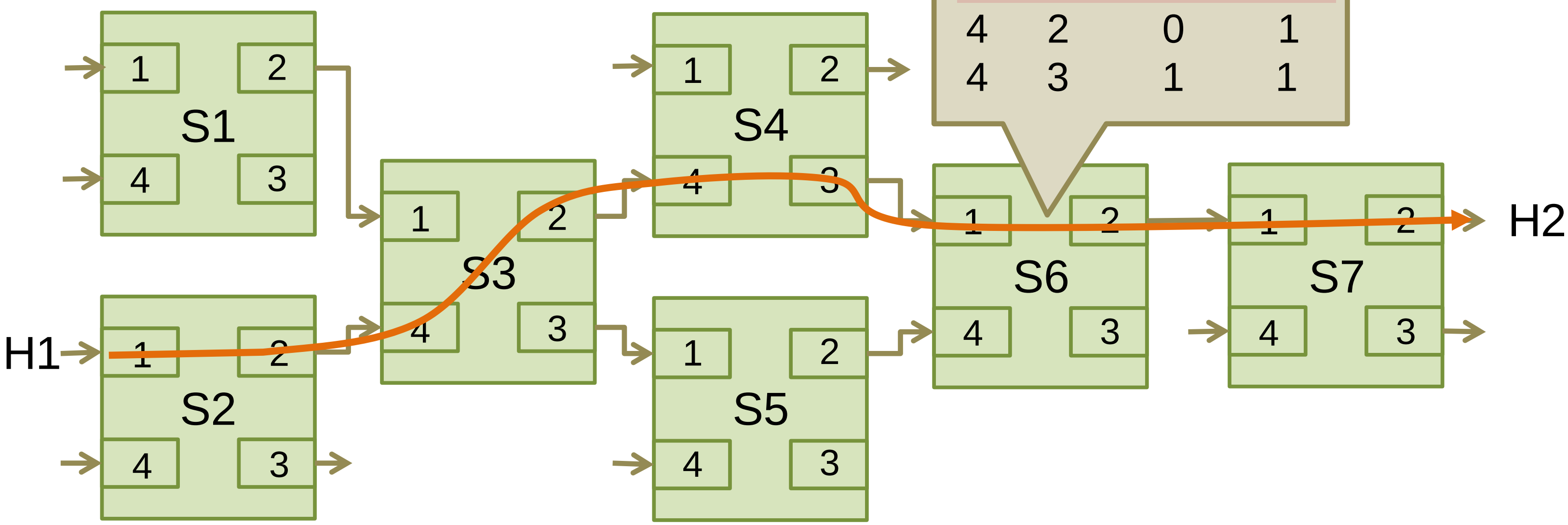
In	Out	SL	VL
1	2	0	0
1	2	2	3
4	2	0	1
4	3	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	

Incrementing the VL (dian

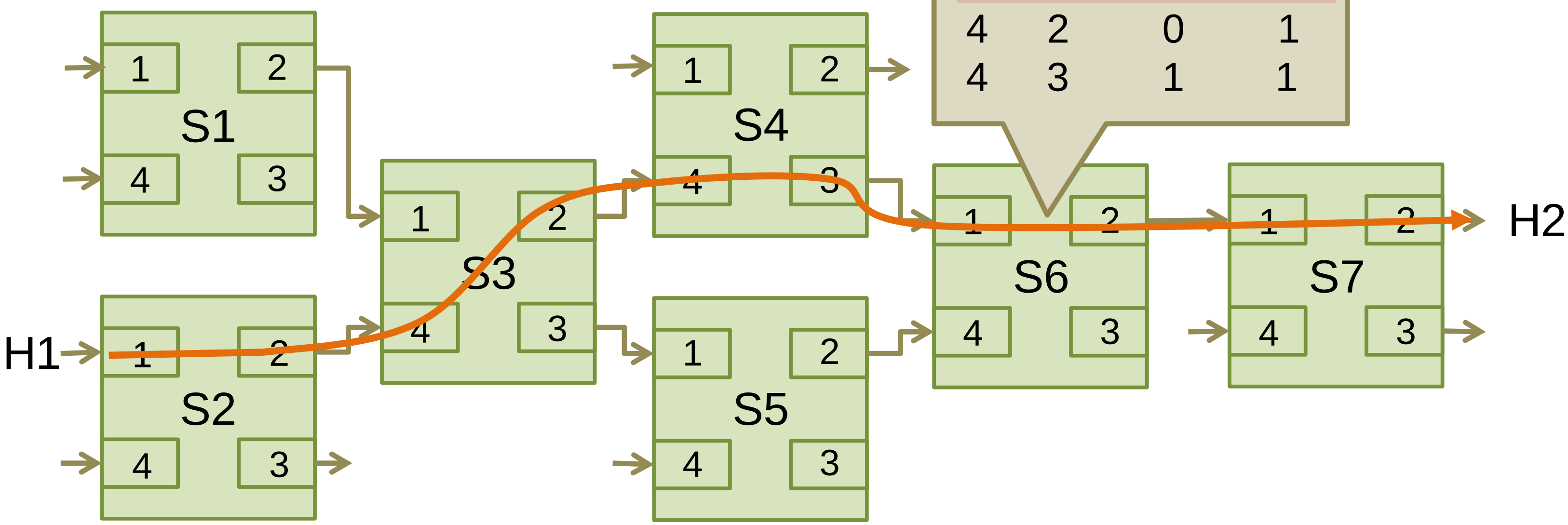
In	Out	SL	VL
1	2	0	0
1	2	2	3
4	2	0	1
4	3	1	1



SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	

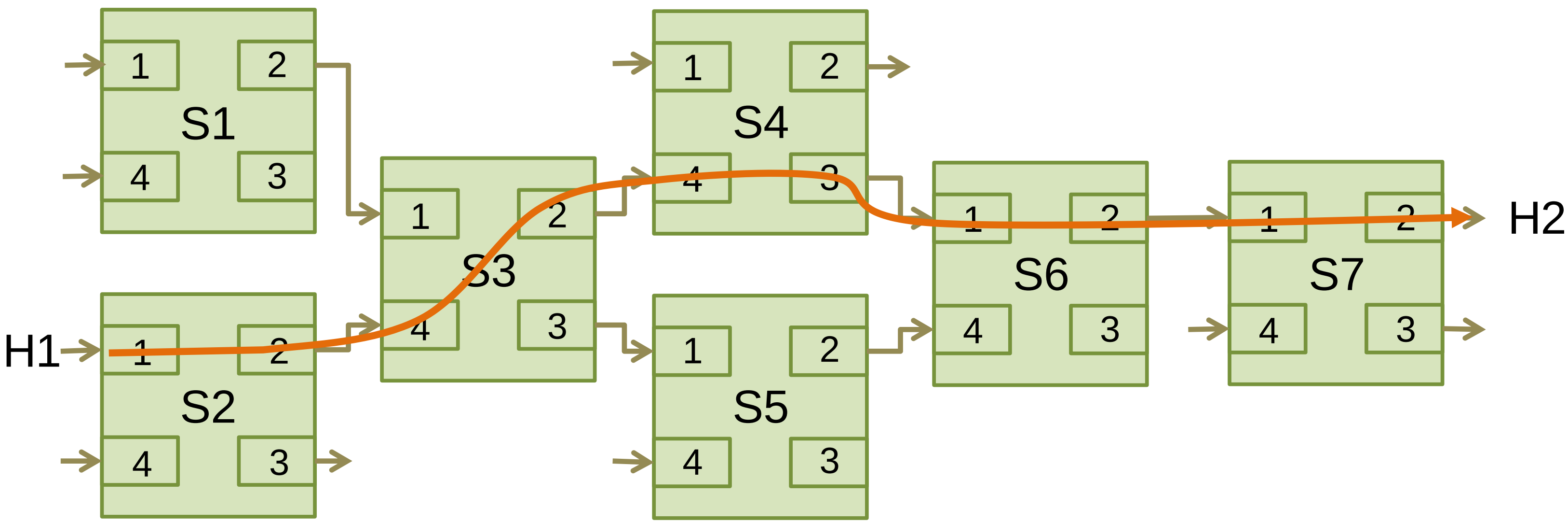
Incrementing the VL (dian

In	Out	SL	VL
1	2	0	0
1	2	2	3
4	2	0	1
4	3	1	1



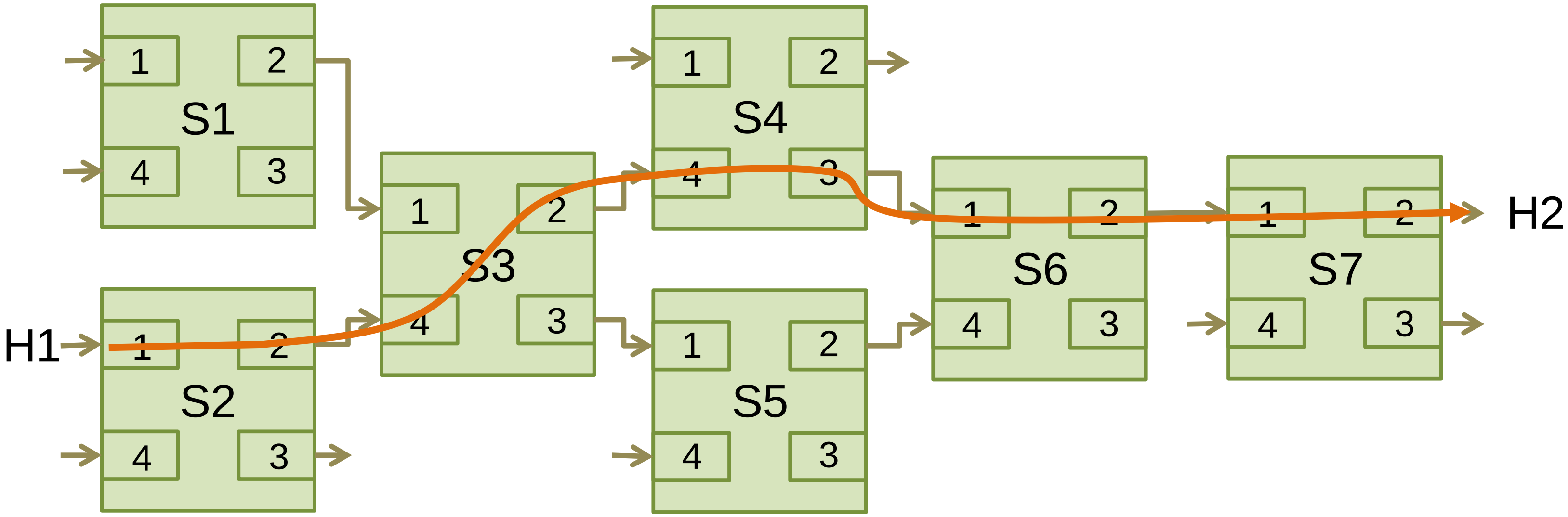
SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



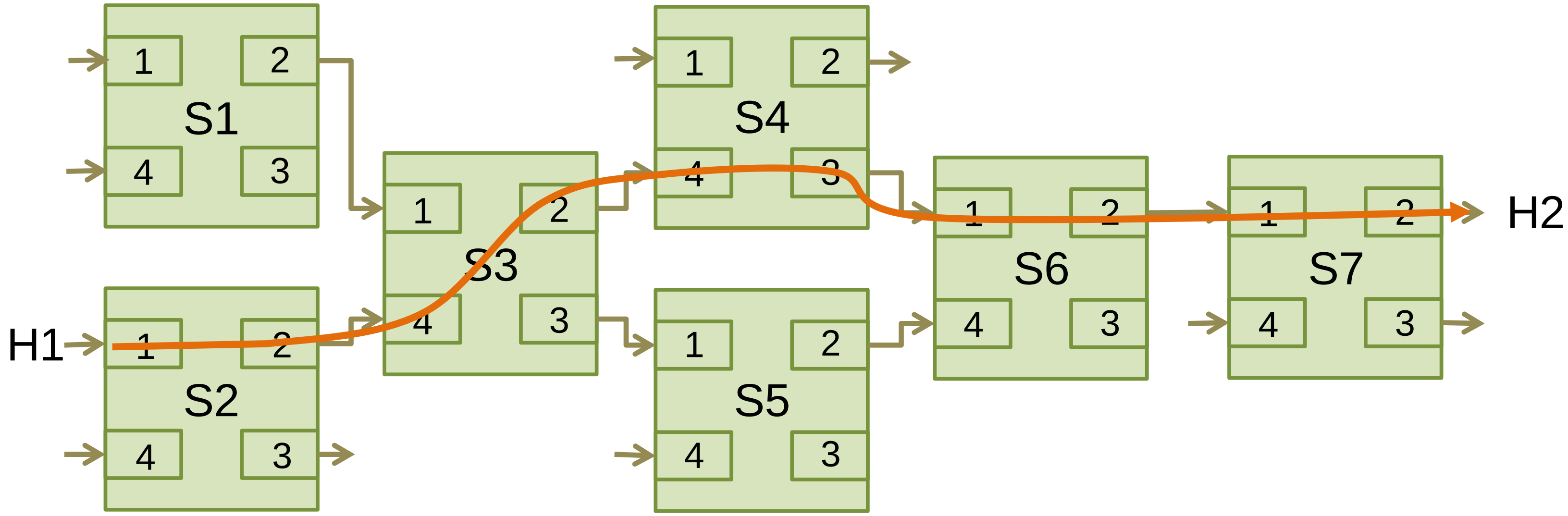
SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



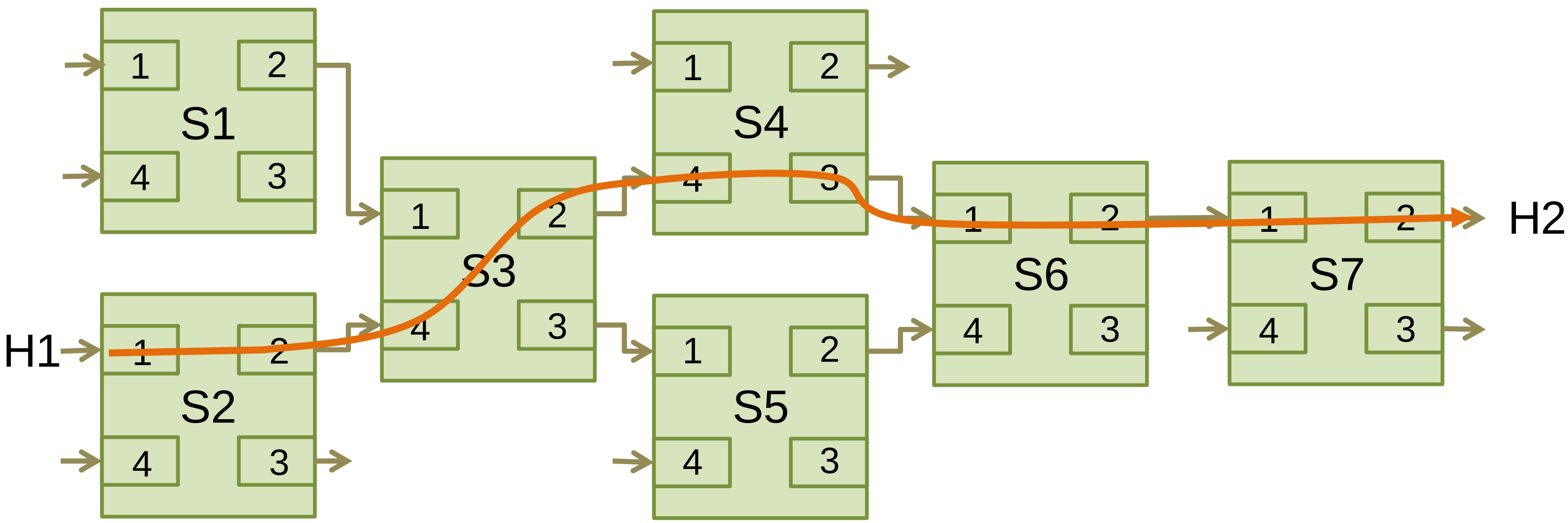
SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



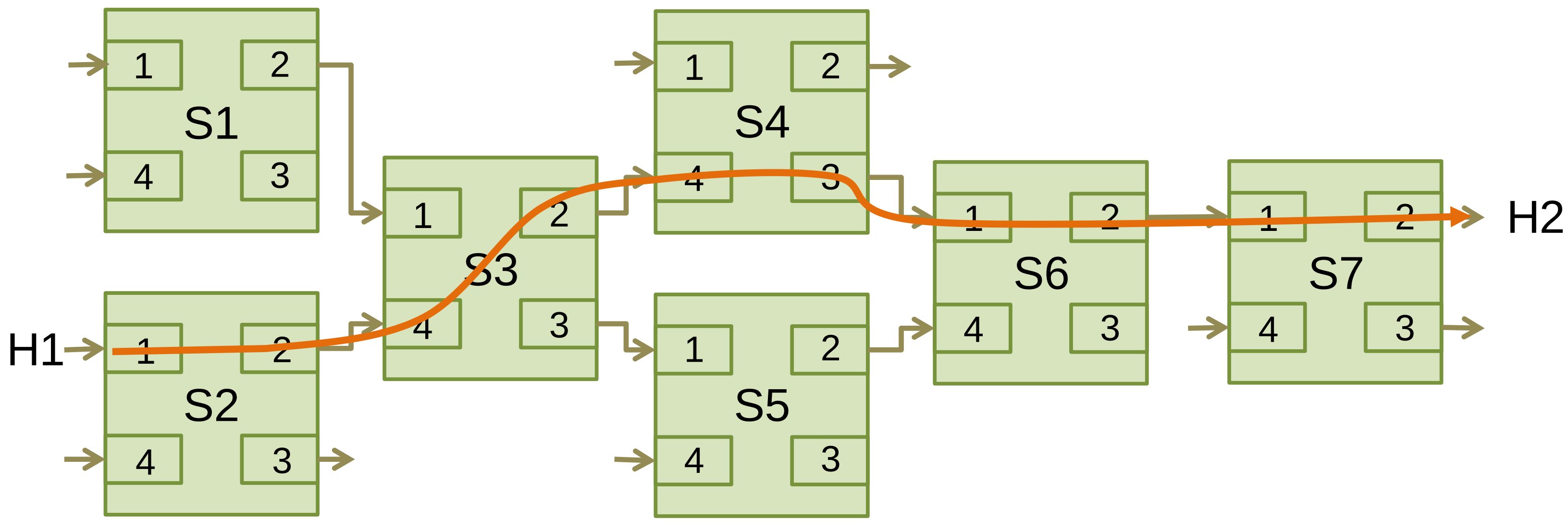
SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



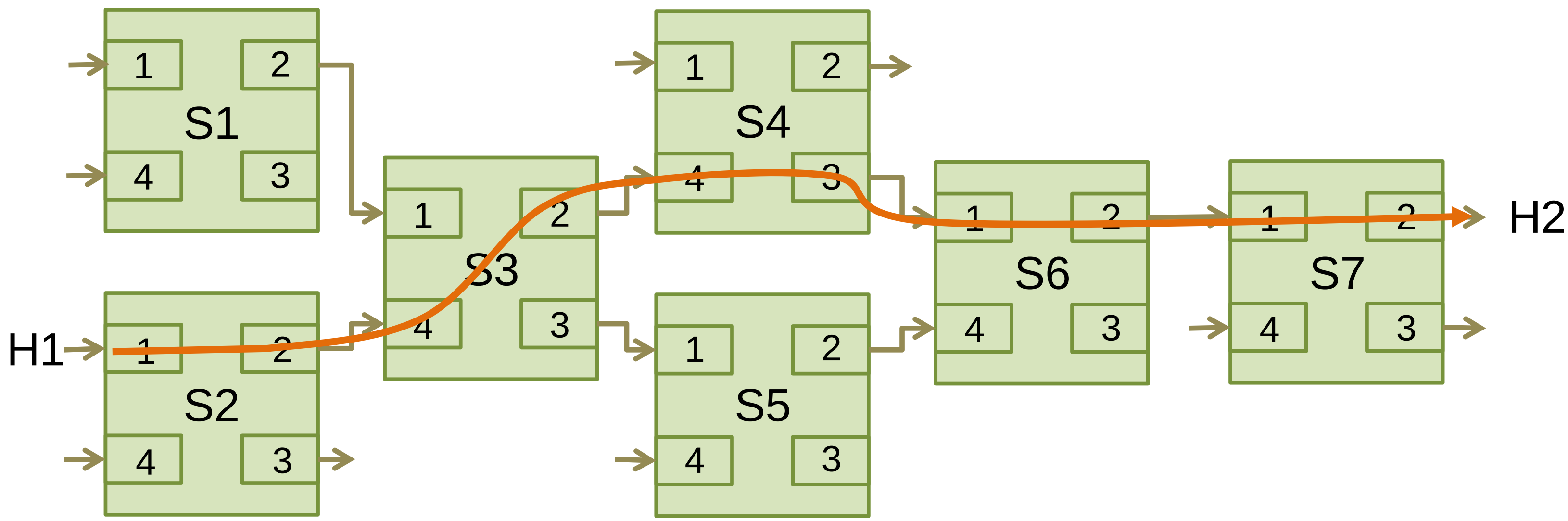
SL 0	SL 1	SL 2	SL 3	…
2	1	0	0	
1	1	0	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



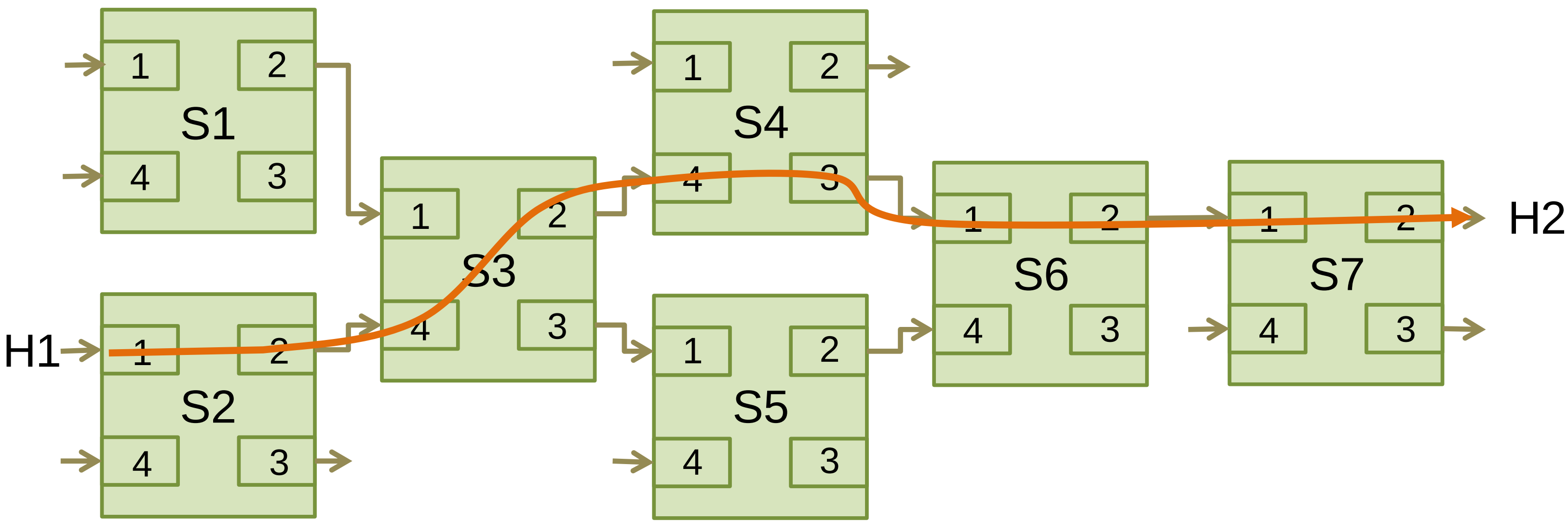
SL 0	SL 1	SL 2	SL 3	…
2	1	1	0	
1	1	2	0	
0	0	3	0	

Incrementing the VL (diameter > 2)



SL 0	SL 1	SL 2	SL 3	…
2	1	1	0	
1	1	2	0	
0	0	3	0	

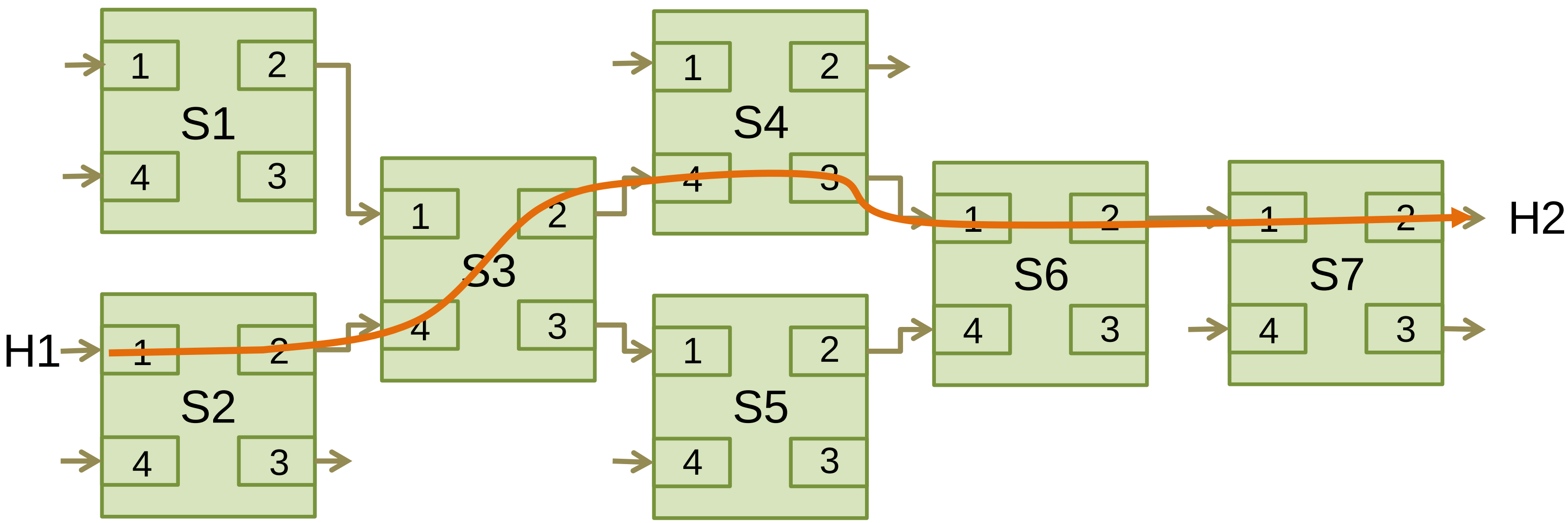
Incrementing the VL (diameter > 2)



SL 0	SL 1	SL 2	SL 3	…
2	1	1	0	
1	1	2	0	
0	0	3	0	

( )
( )
( )

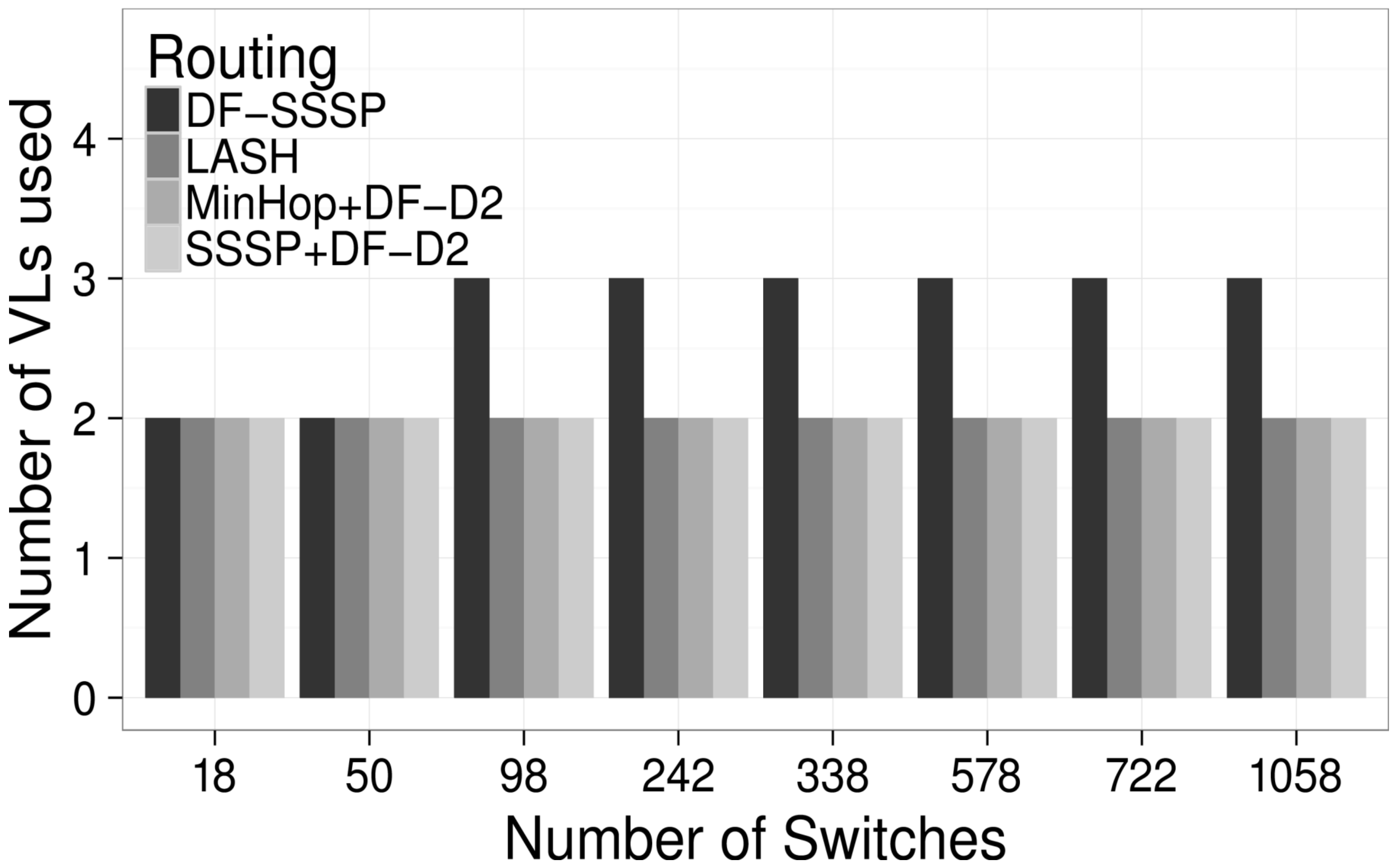
Incrementing the VL (diameter > 2)



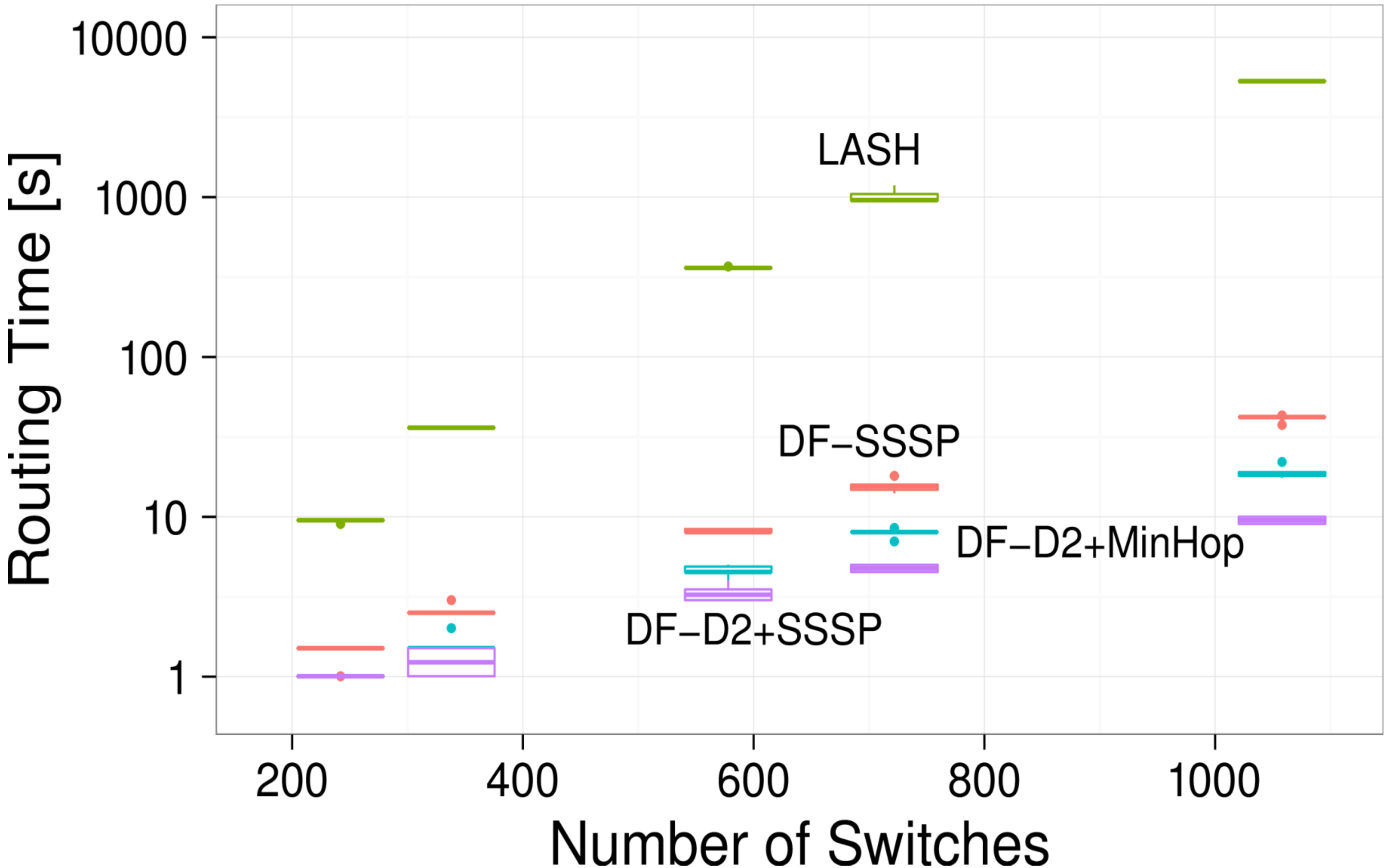
SL 0	SL 1	SL 2	SL 3	…
2	1	1	0	
1	1	2	0	
0	0	3	0	

( )
( )
( )

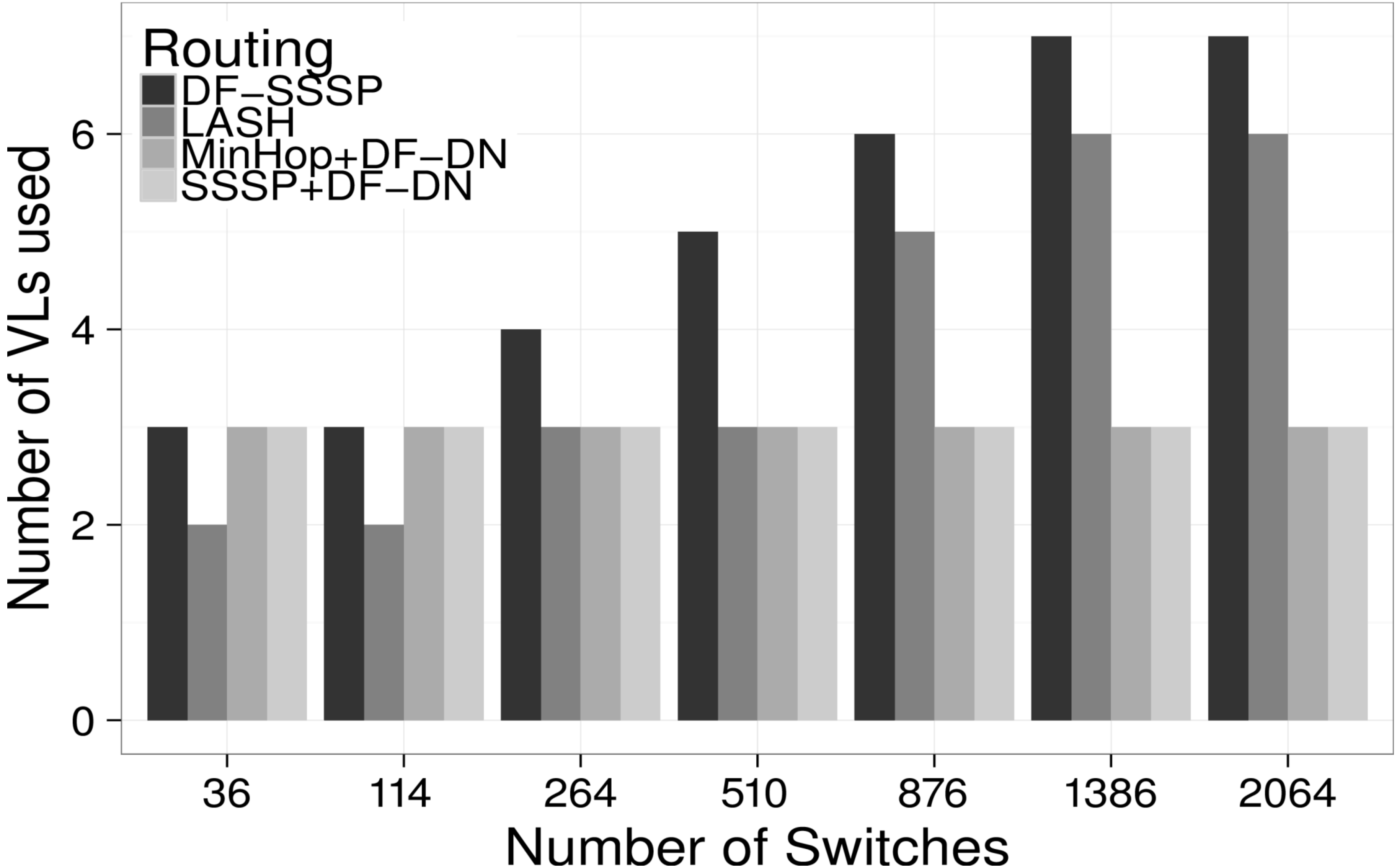
Results: Slim Fly Topologies (Diameter Two)



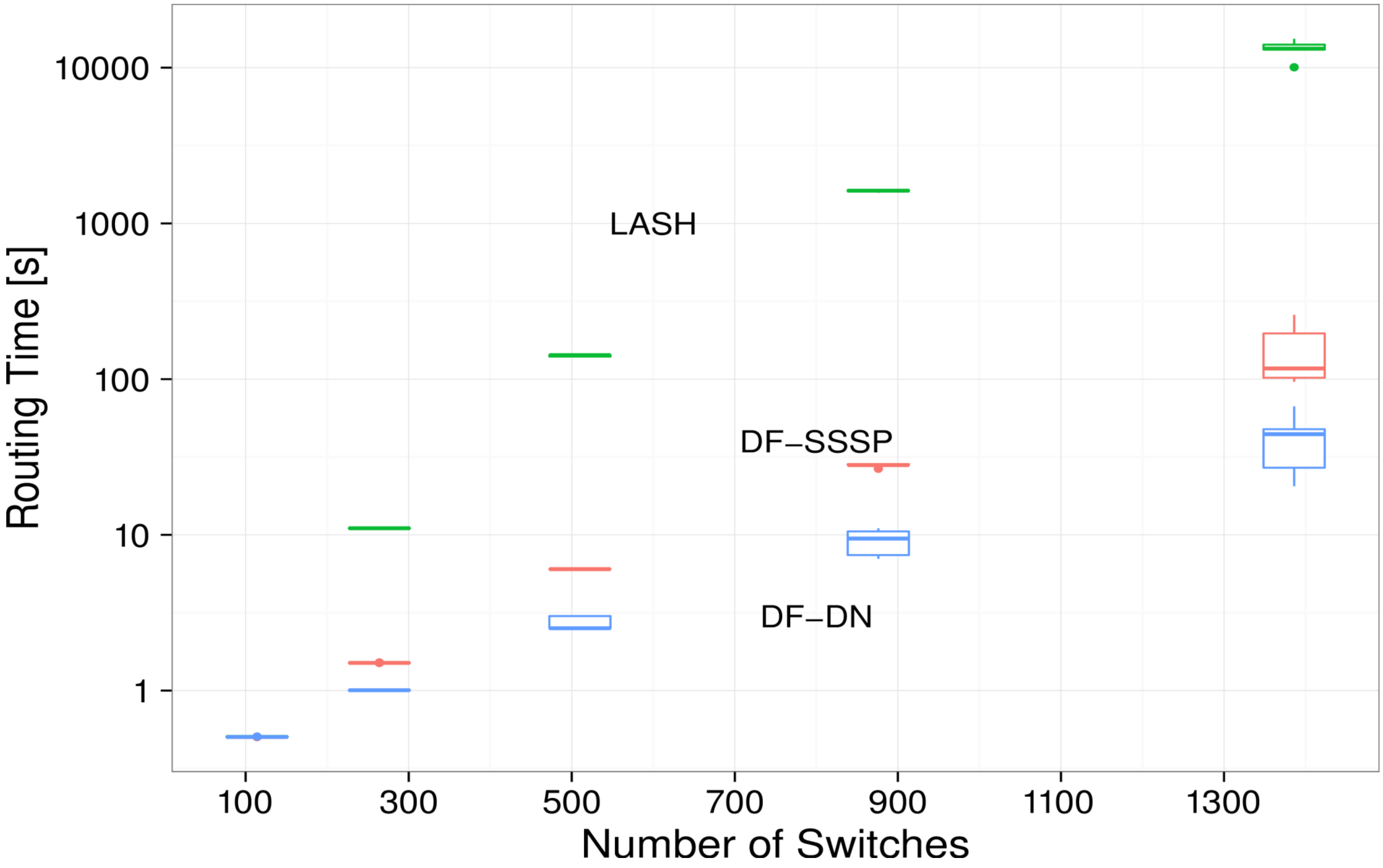
Results: Slim Fly Topologies (Diameter Two)



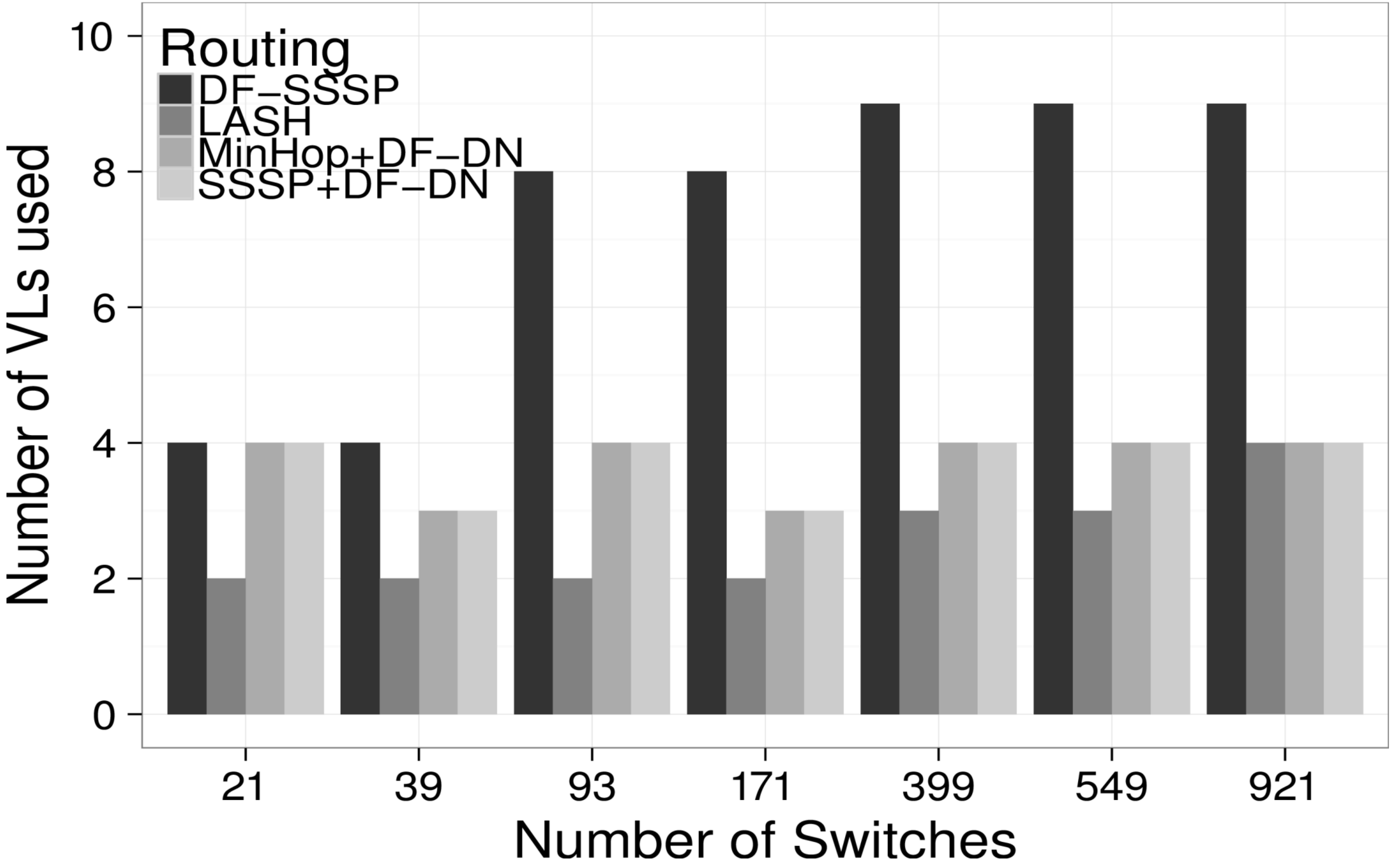
Results: Dragonfly Topologies (Diameter 3)



Results: Dragonfly Topologies



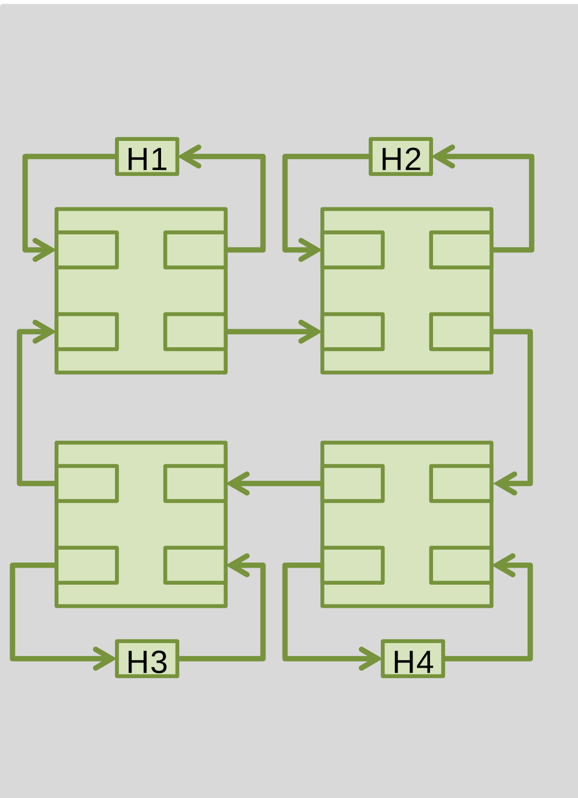
Results: Orthogonal Fat Tree Topologies

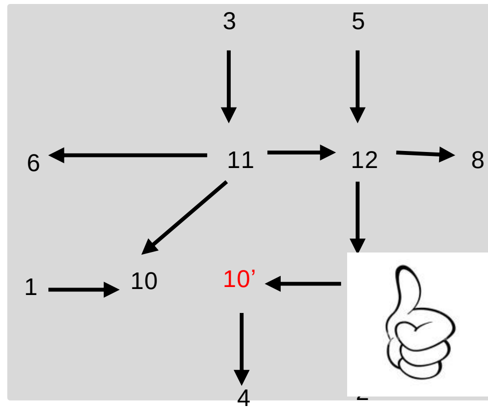


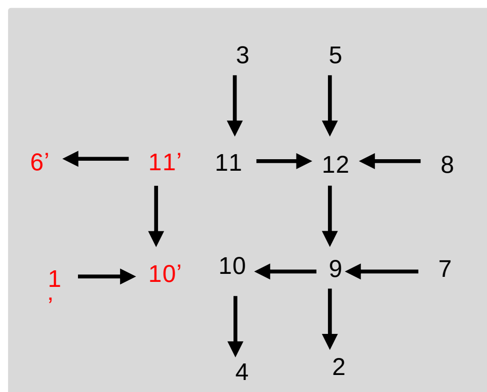
Age-based arbitration

- VLs are supposed to be used for QoS and deadlock-avoidance
- Arbiters can prioritize packets based on VLs → not explored in this work, but shown to improve performance by 30% on other networks
- In our heuristic the VL value corresponds to the age of packets
- Saving VLs in deadlock-avoidance allows to use them for separation of traffic classes → Relevant for datacenters!

Conclusions







( )