Routing, Deadlocks and all that

OPENFABRICS ALLIANCE

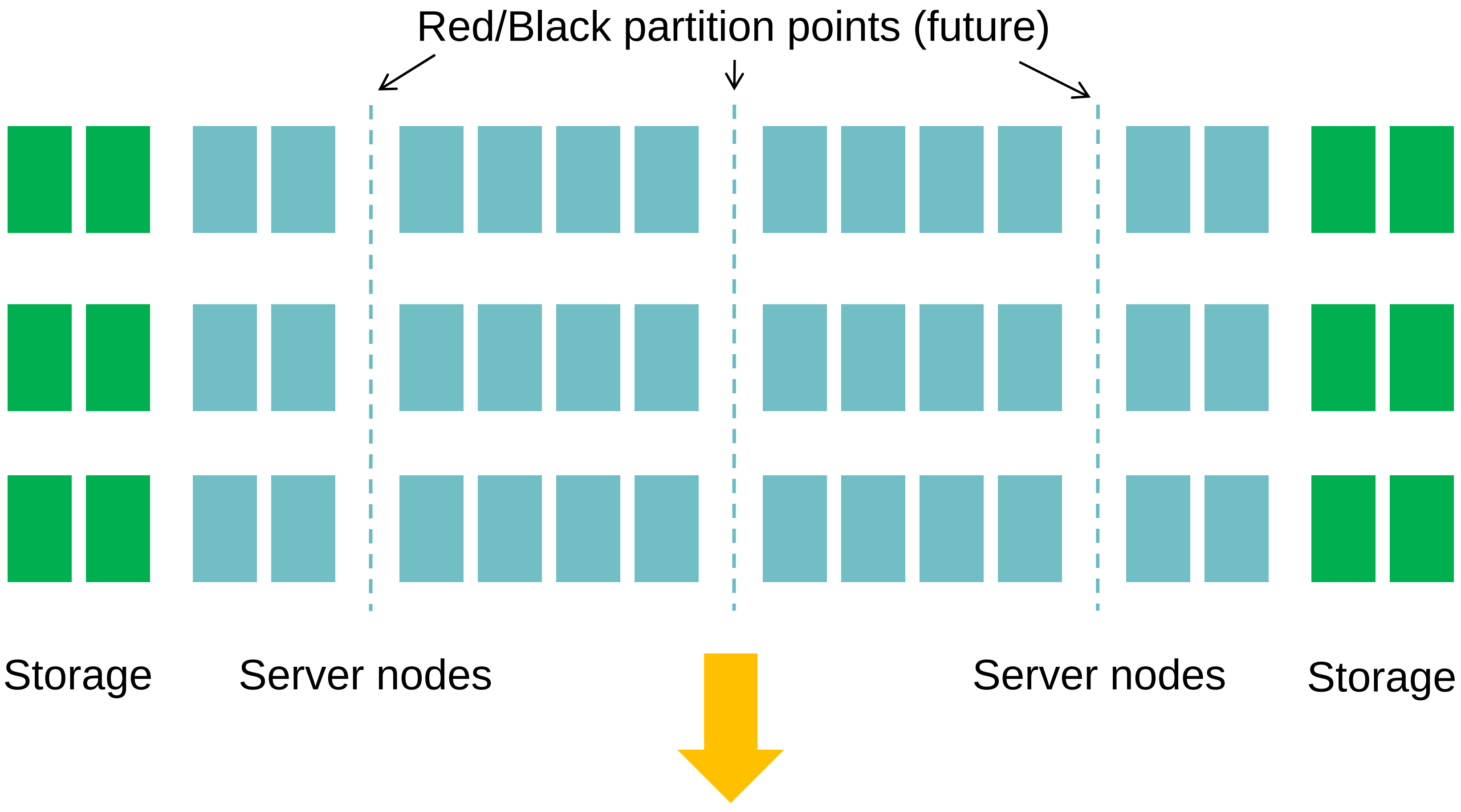
Issues from the deployment of the Redsky machine
Bob Pearson, Dave McMillen System Fabric Works

Credits

- Matt Bohnsack – Sandia
- Doug Doerfler – Sandia
- Line Holen – Sun
- Lars Paul Huse - Sun
- Bjorn-Dag Johnson – Sun
- Dave McMillen – SFW
- John Naegle – Sandia
- Rob Netzer – SFW
- Bob Pearson (me) – SFW
- Sven-Arne Reinemo – Simula
- Hal Rosenstock
- Jim Schutt – Sandia
- Eitan Zahavi – Mellanox

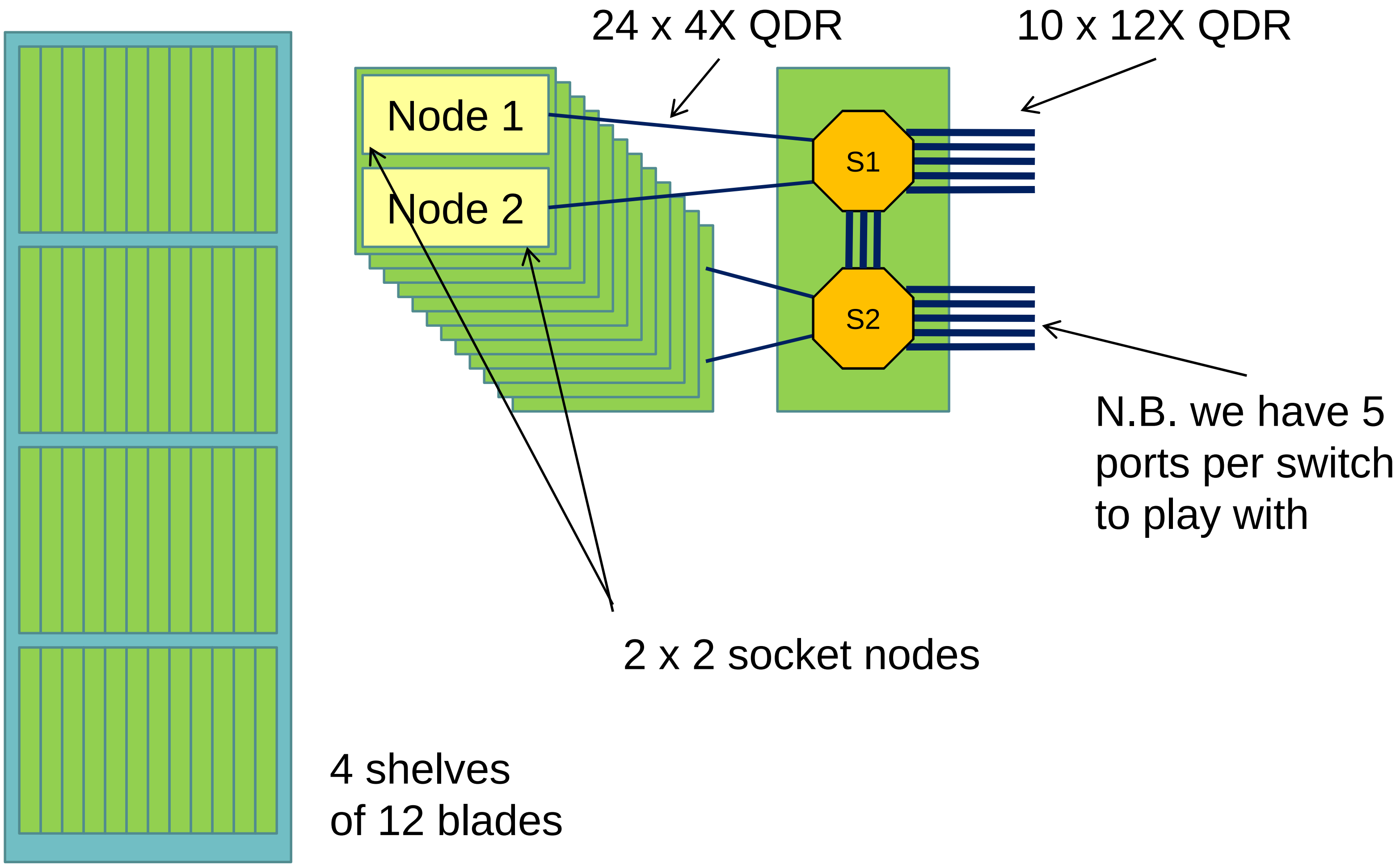
REDSKY

Redsky



Direction of future growth

C48 Rack

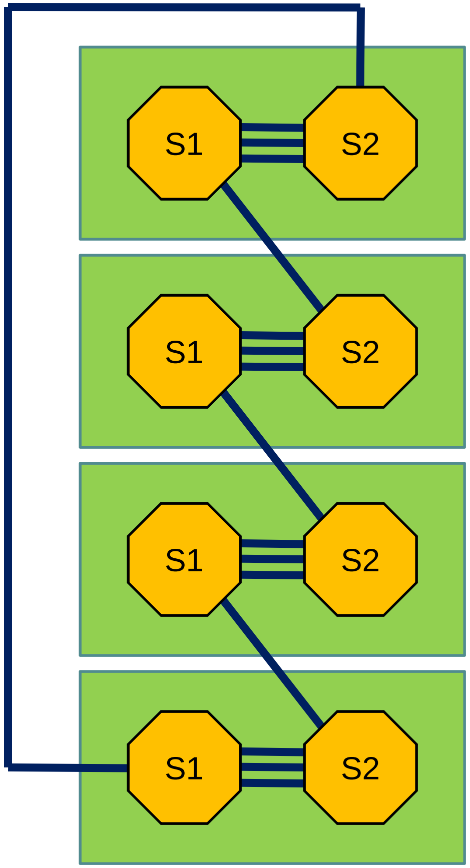


N.B. we have 5 ports per switch to play with

2 x 2 socket nodes

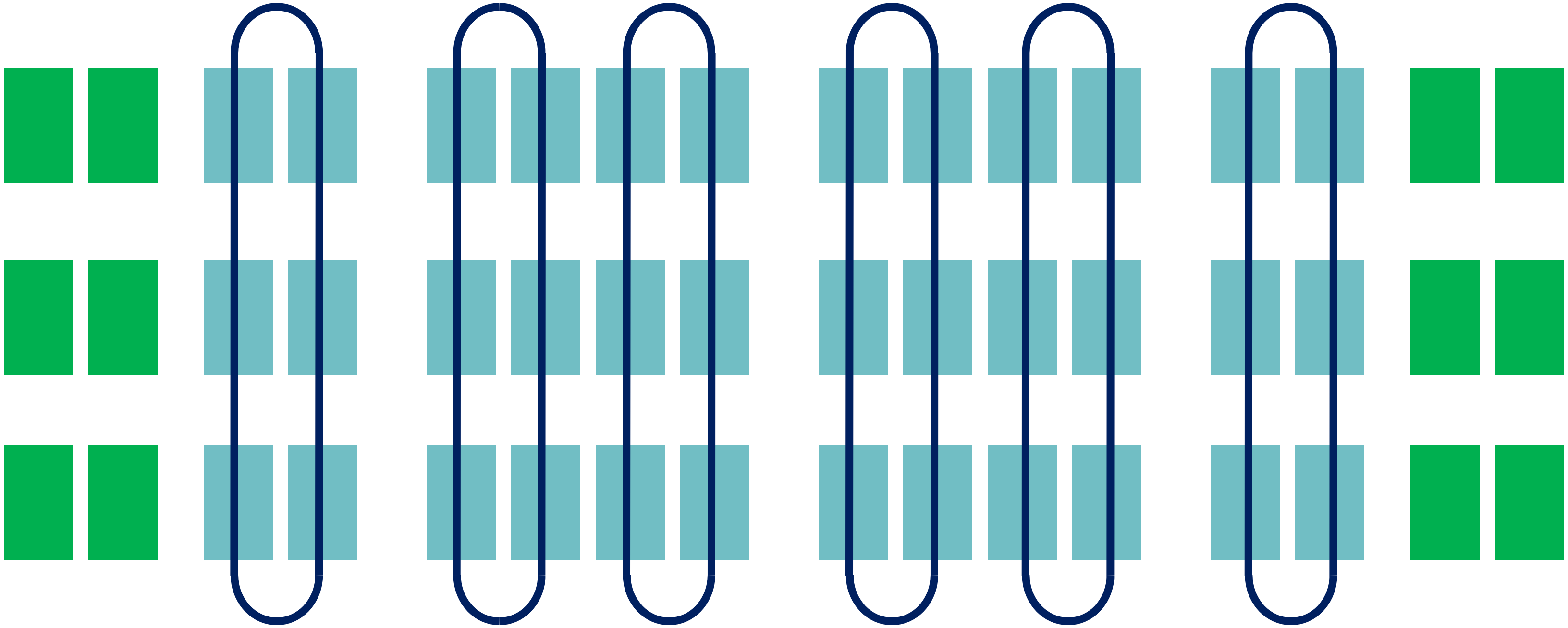
4 shelves
of 12 blades

Z Axis Wiring



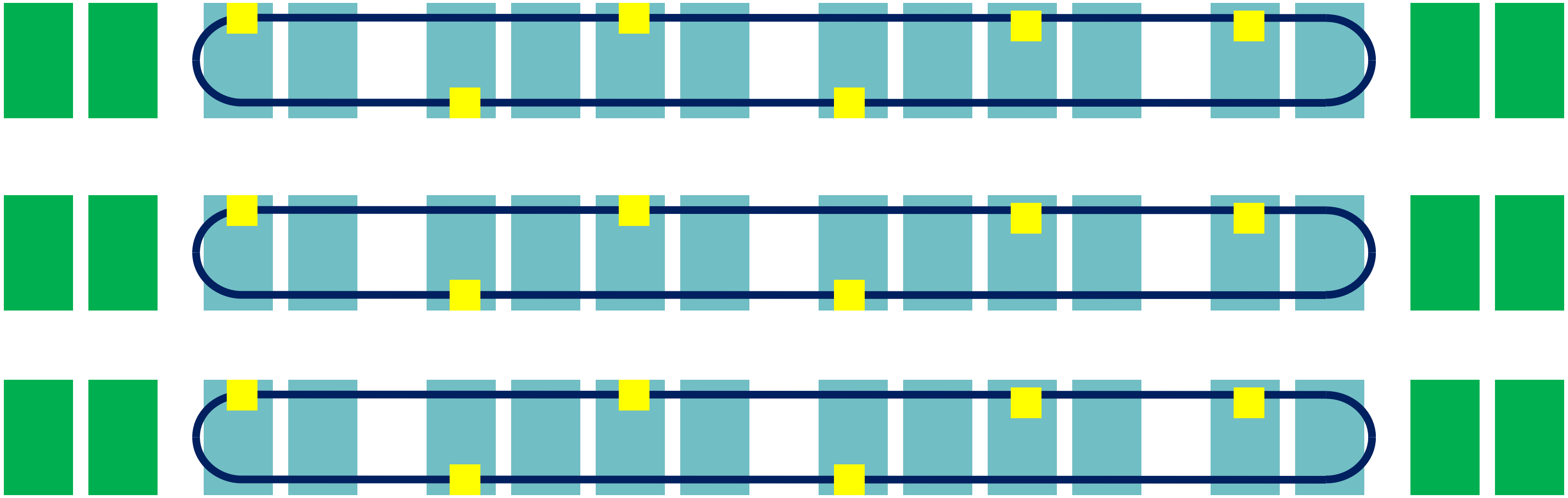
Uses 1 of the 5
12X links out of each switch

Y Axis Wiring



Uses 2 more of the 5 links

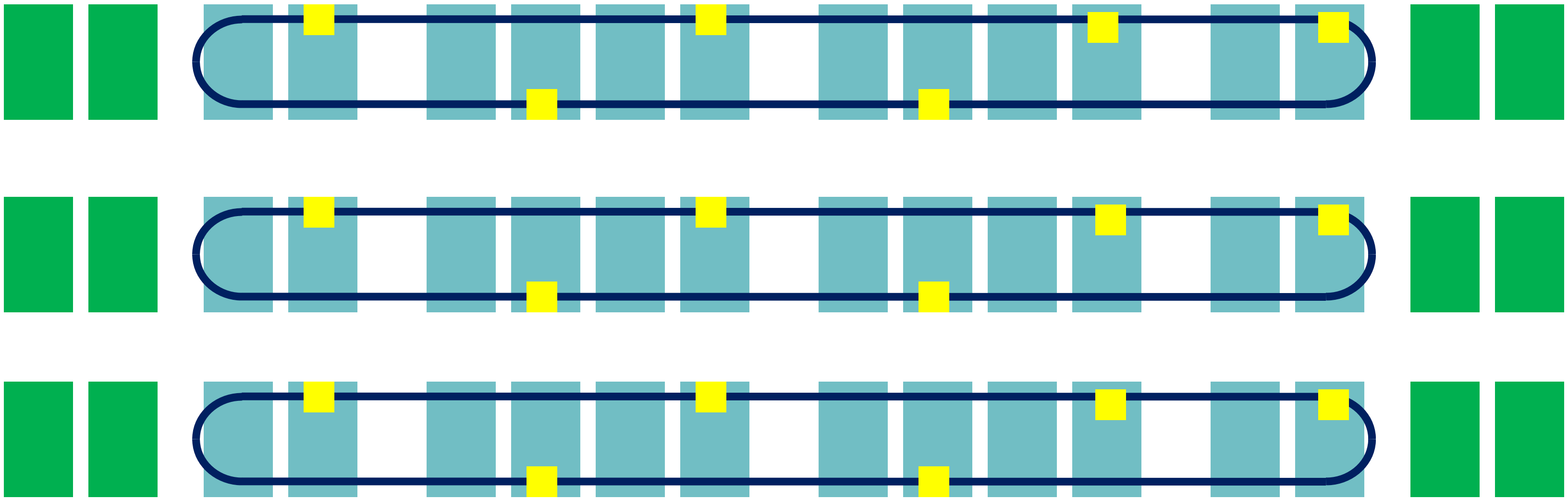
X Axis Wiring(1)



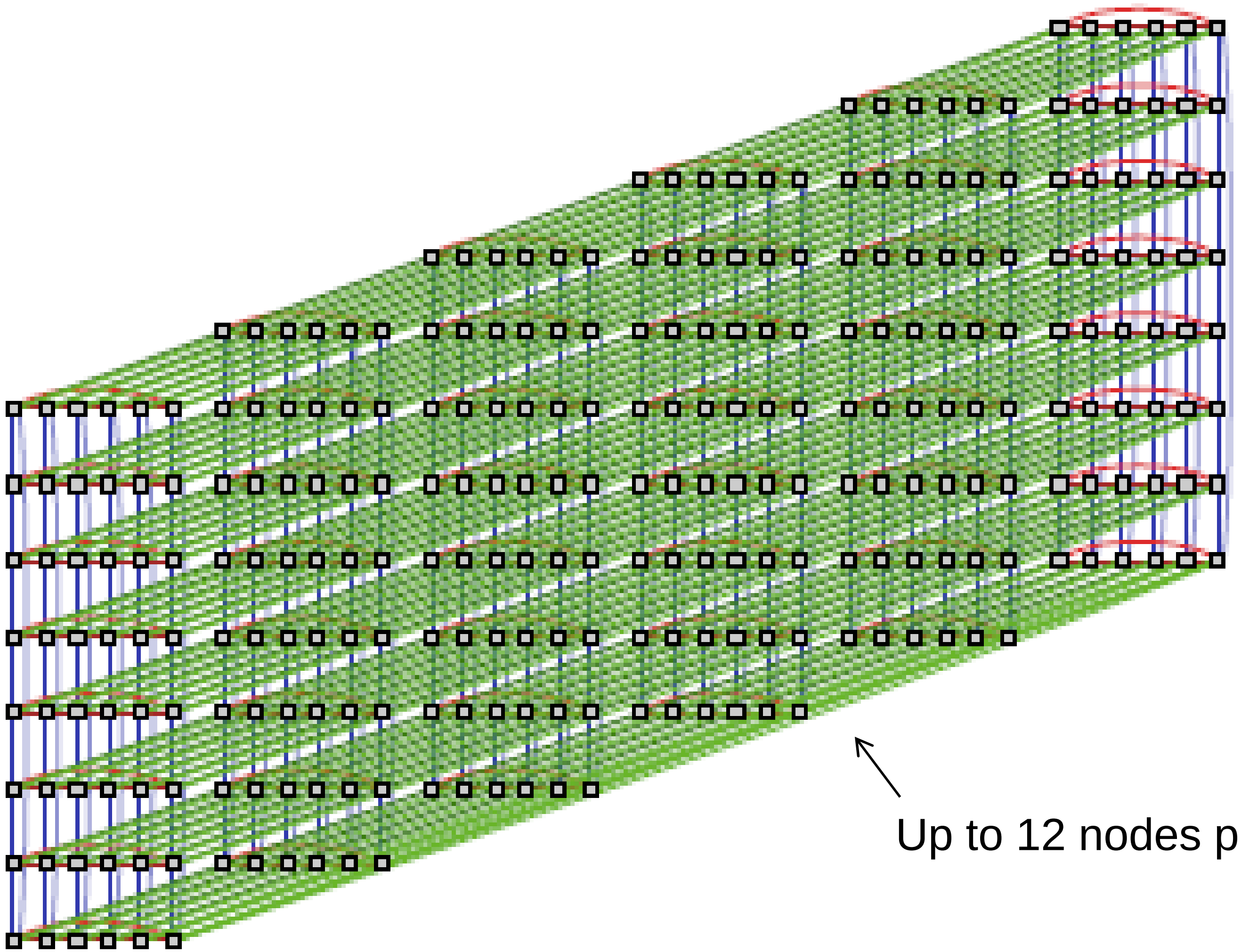
(recall the Y axis uses racks in pairs)

Uses the final 2 links

X Axis Wiring(2)



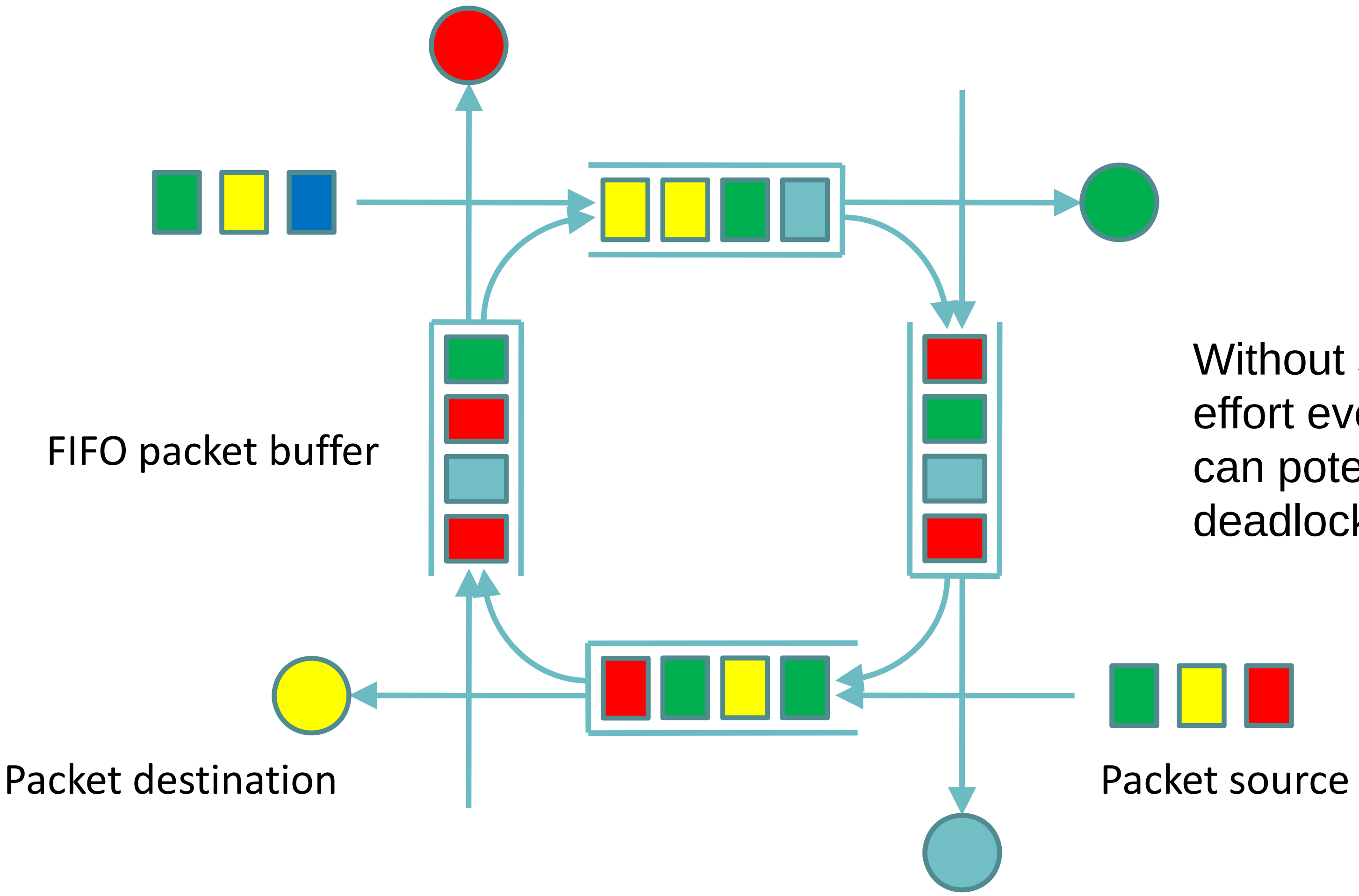
6x6x8 Torus



Up to 12 nodes per switch

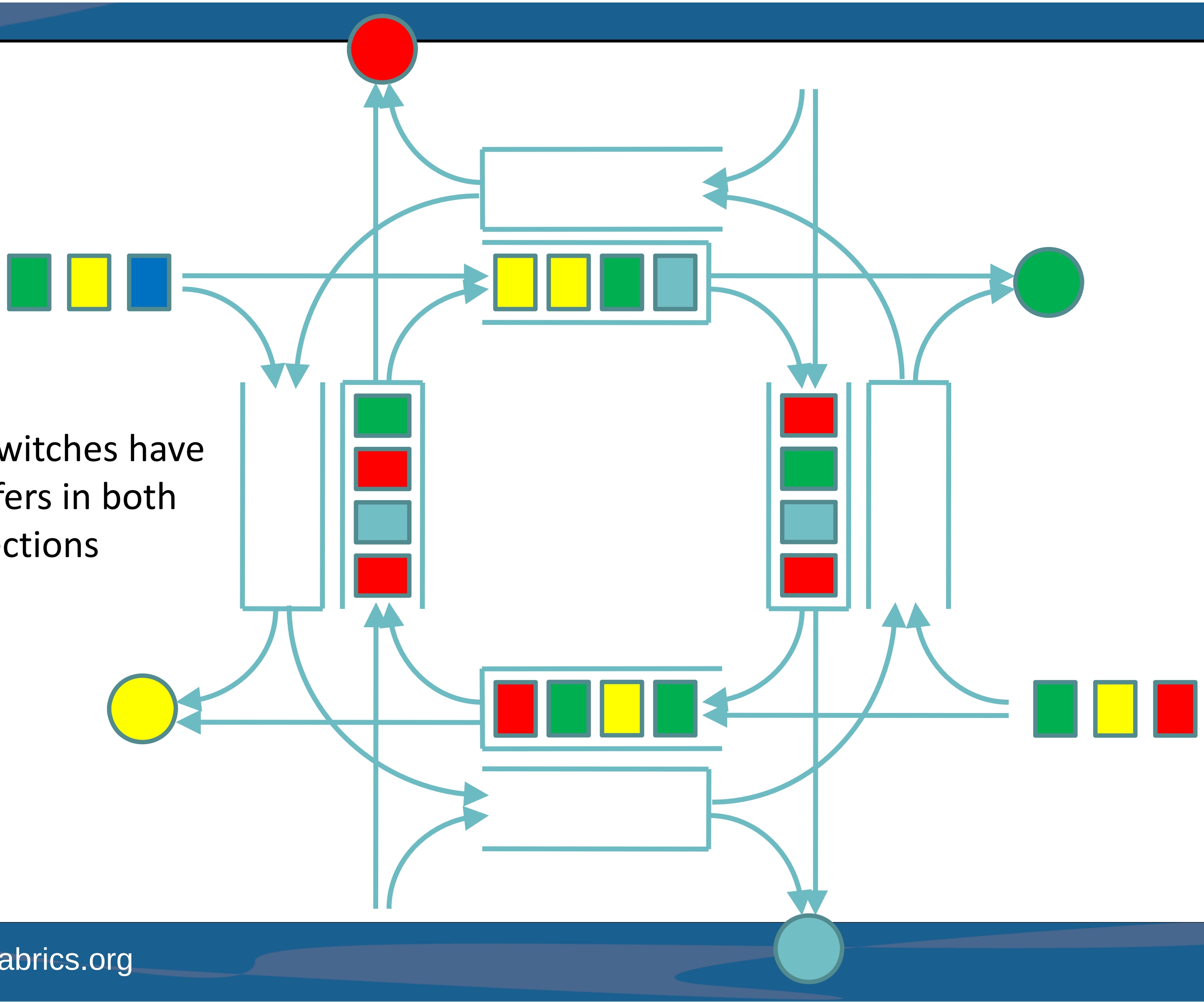
IB FABRIC DEADLOCKS

Example Deadlocked Loop



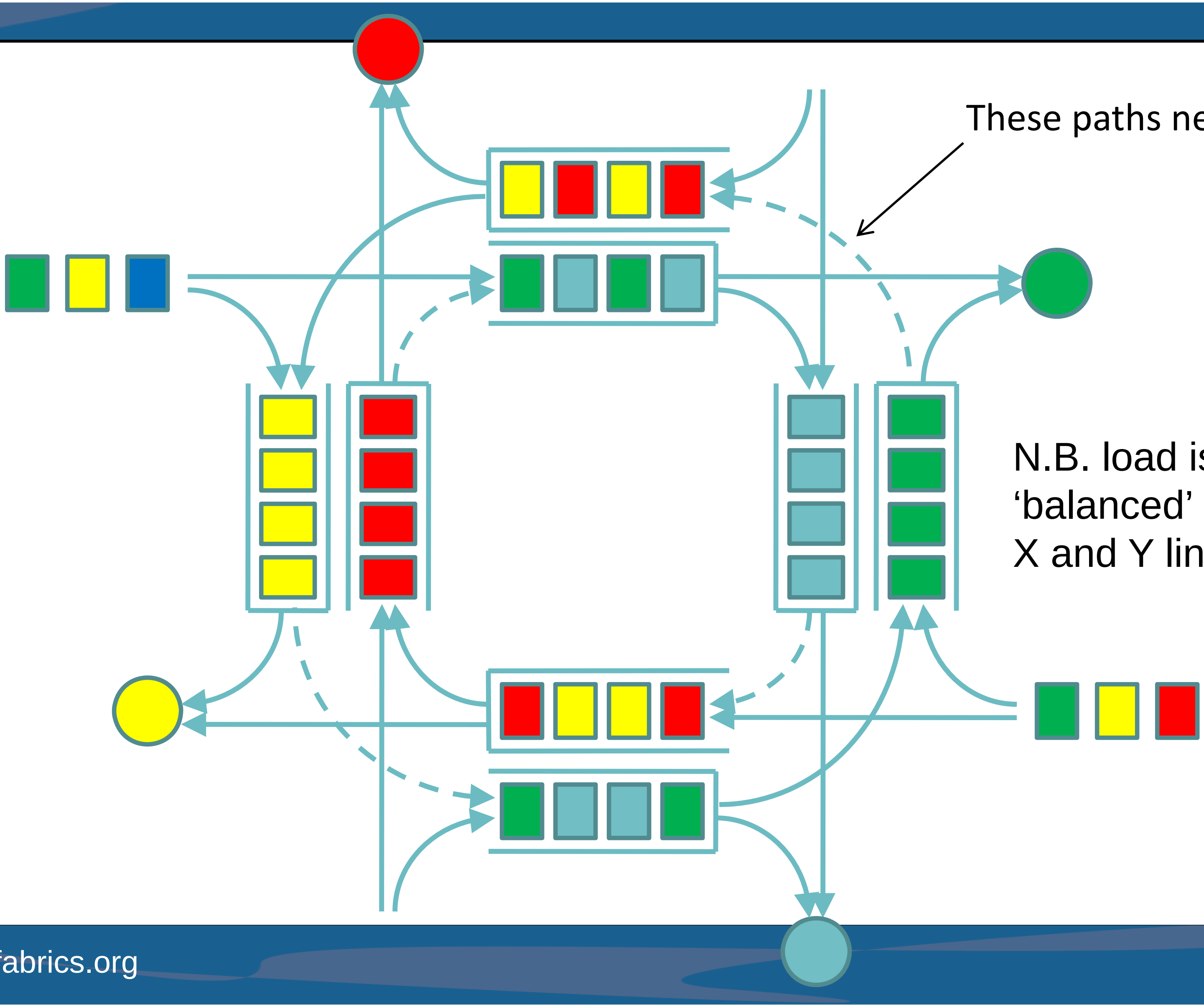
Without some effort every loop can potentially deadlock

Square IB Network



IB switches have buffers in both directions

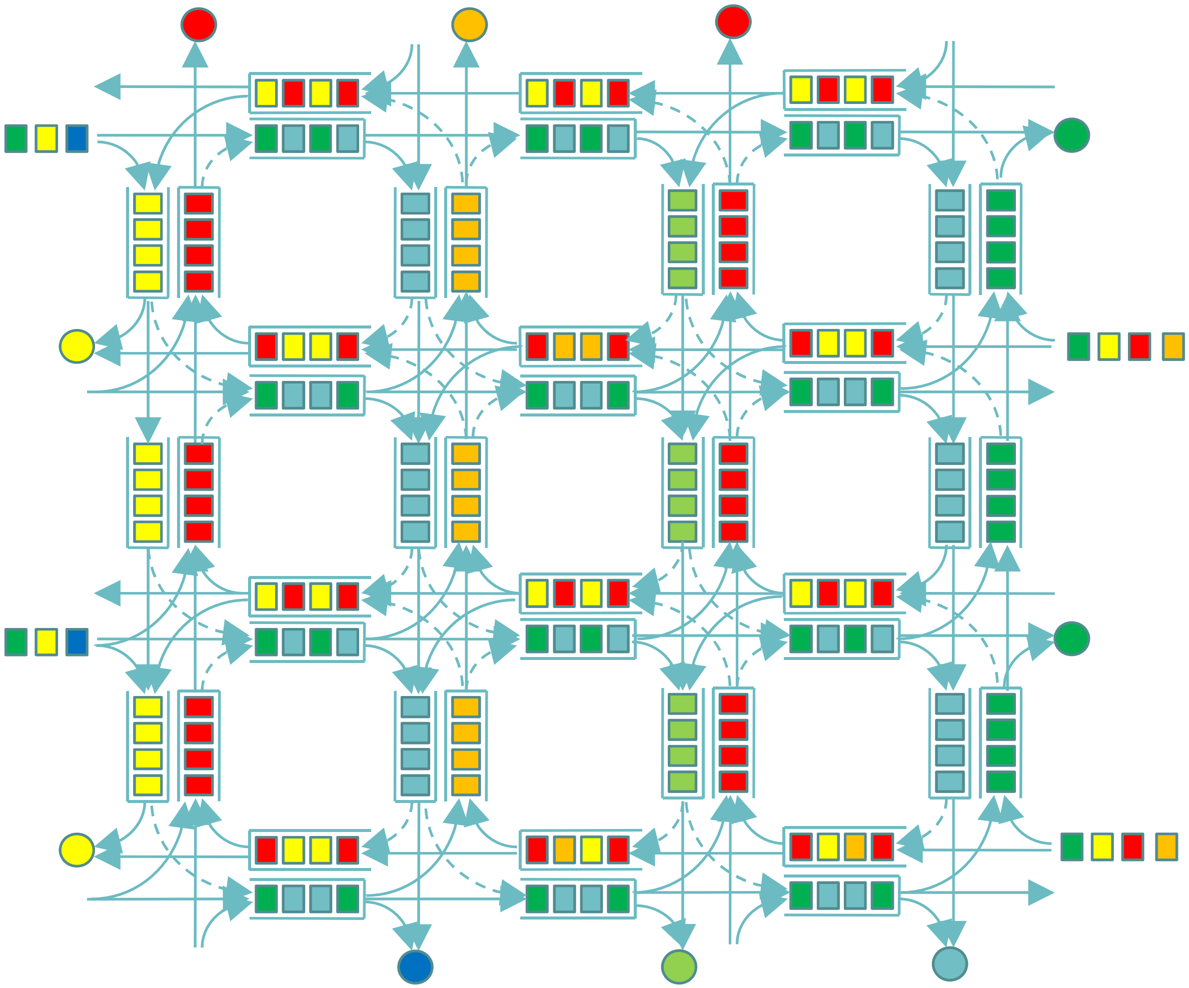
DOR Routing



These paths never used

N.B. load is ‘balanced’ between X and Y links

2D Open Mesh Example



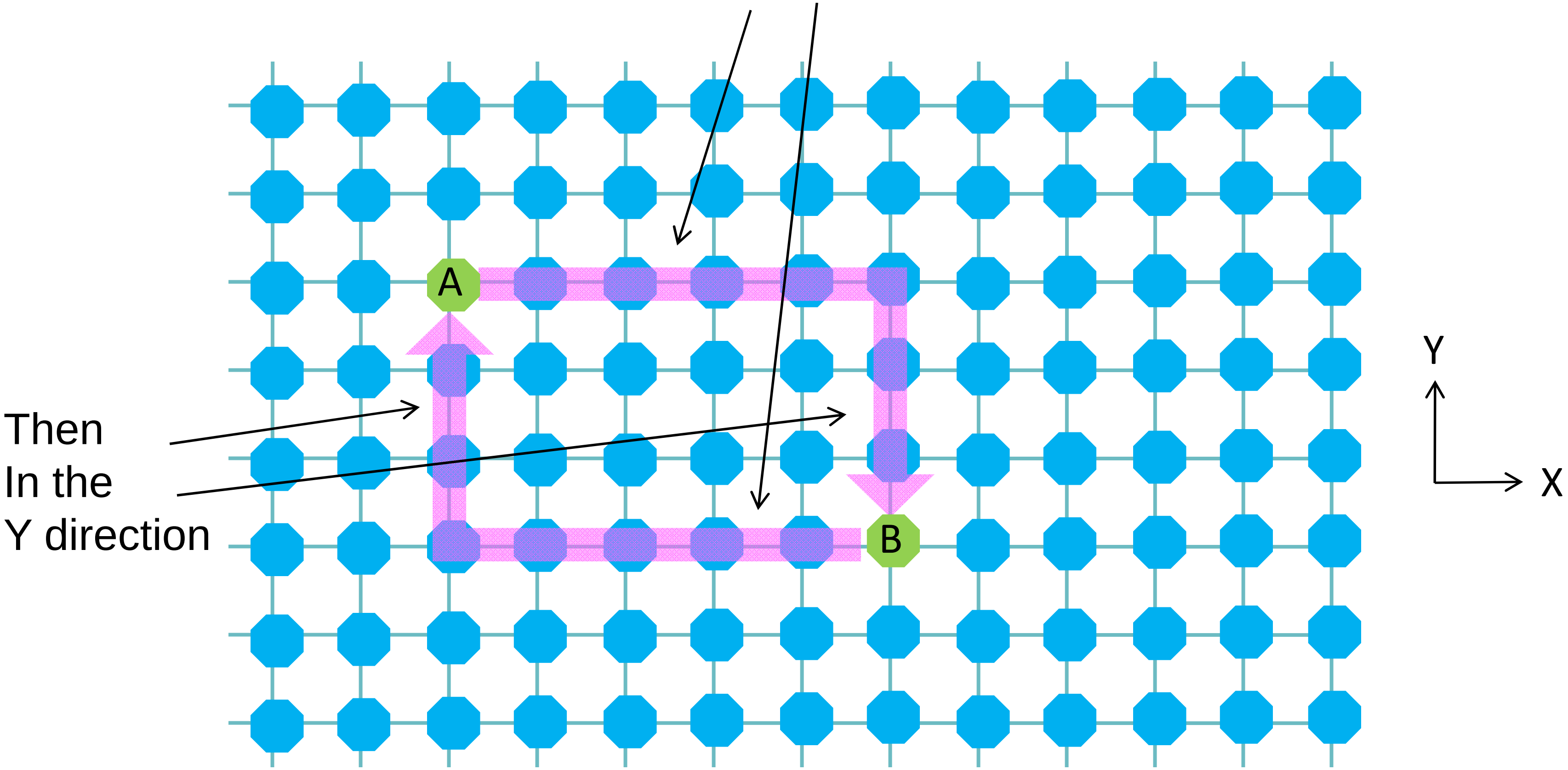
Shortest Distance DOR

- Order the dimensions, e.g. X < Y < Z
- Find all shortest paths from A to B
- Select path that moves in the lowest (selected) dimension first and then the next and so on

DOR Routing Example

First move in the X direction

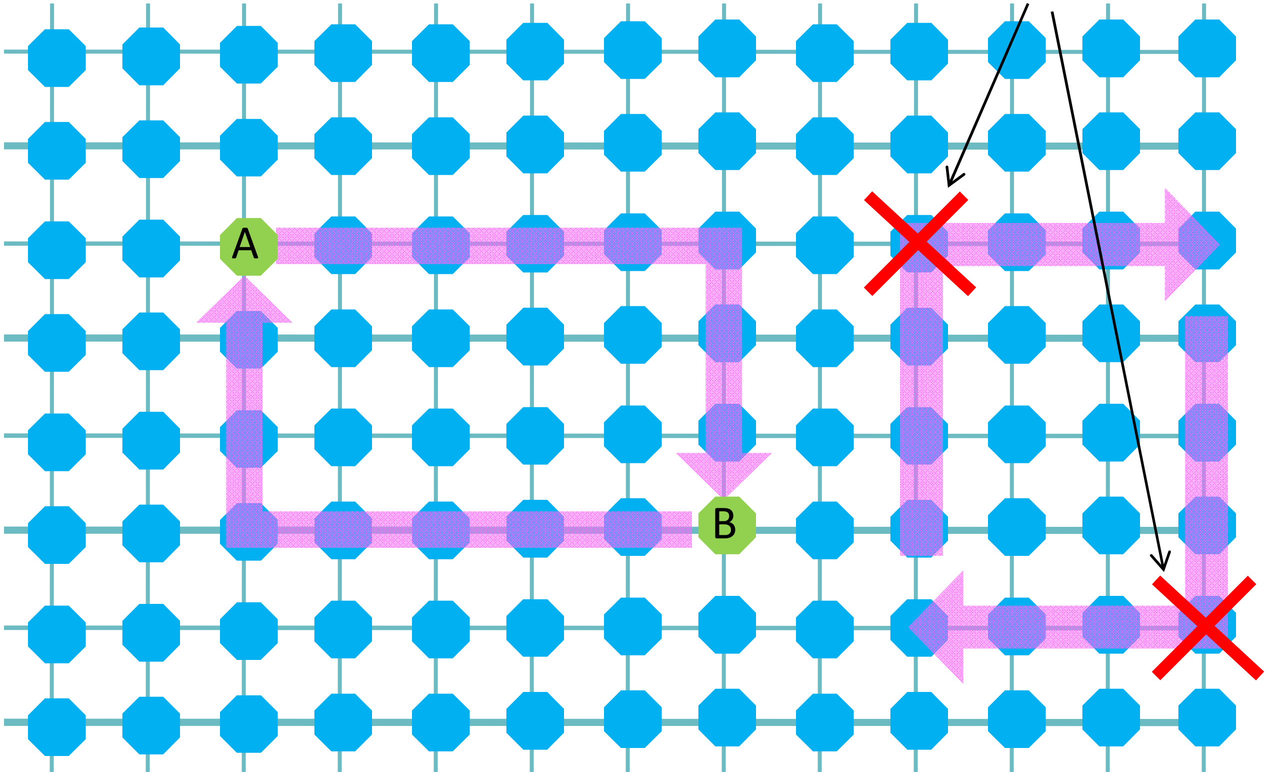
Then
In the
Y direction



Note forward and return paths are different in general

DOR Routing Example

Upper left and lower right corners do not occur



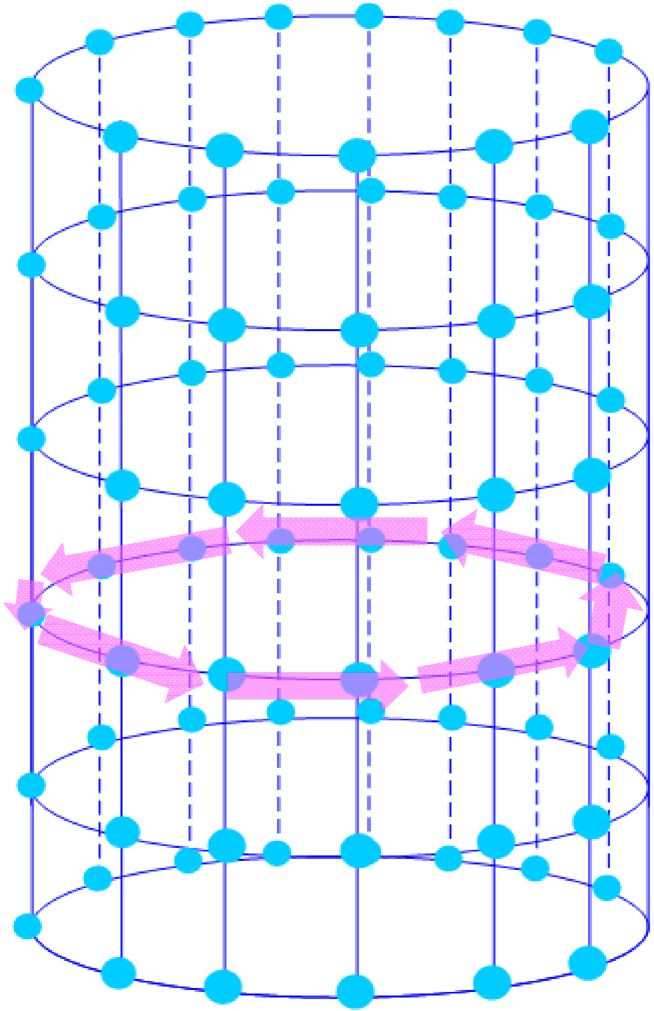
Without all 4 corners can not create closed credit loops!!
Identical return paths would lead to deadlocks

Summary

- Shortest distance DOR routing solves ‘local’ deadlock problem
- No credit loops in Cartesian meshes, of any dimension
- But! Can have “topological” credit loops
  - Overlapping locally ‘legal’ path segments can combine to create credit loop

Deadlock - Example 1

Closed or periodic boundary conditions

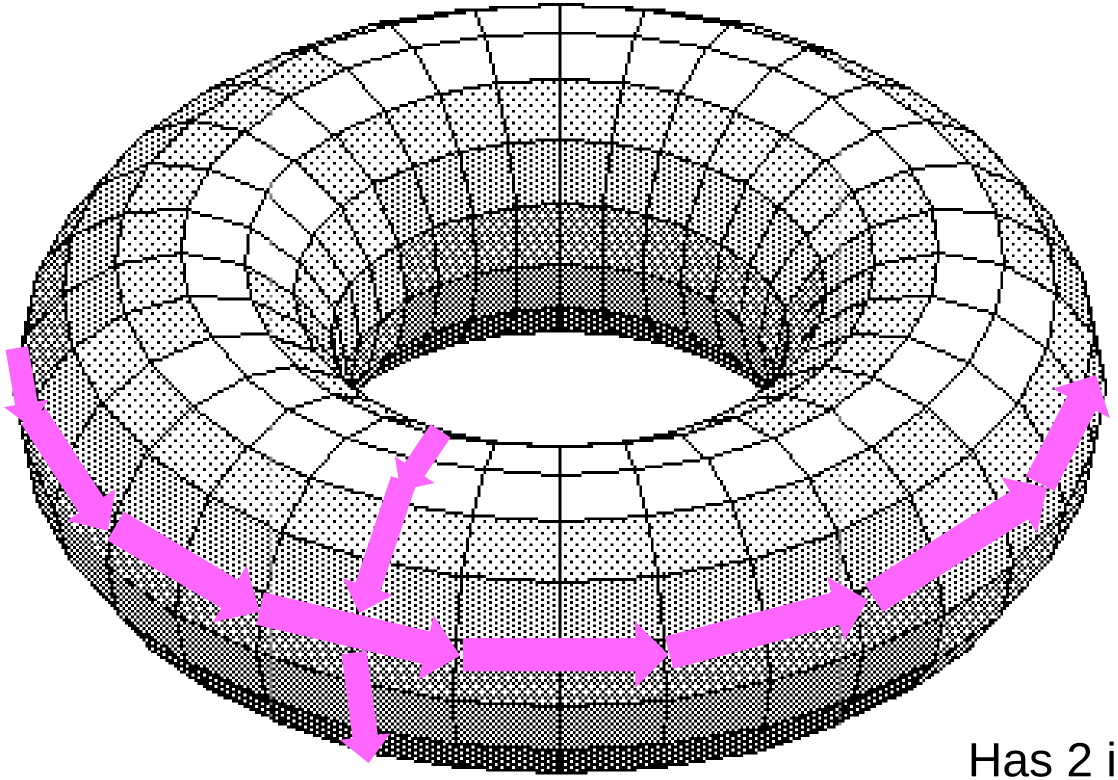


Mesh is locally Cartesian. No local credit loops

Paths around circumference overlaps ‘legal’ DOR path segments

Deadlock - Example 2

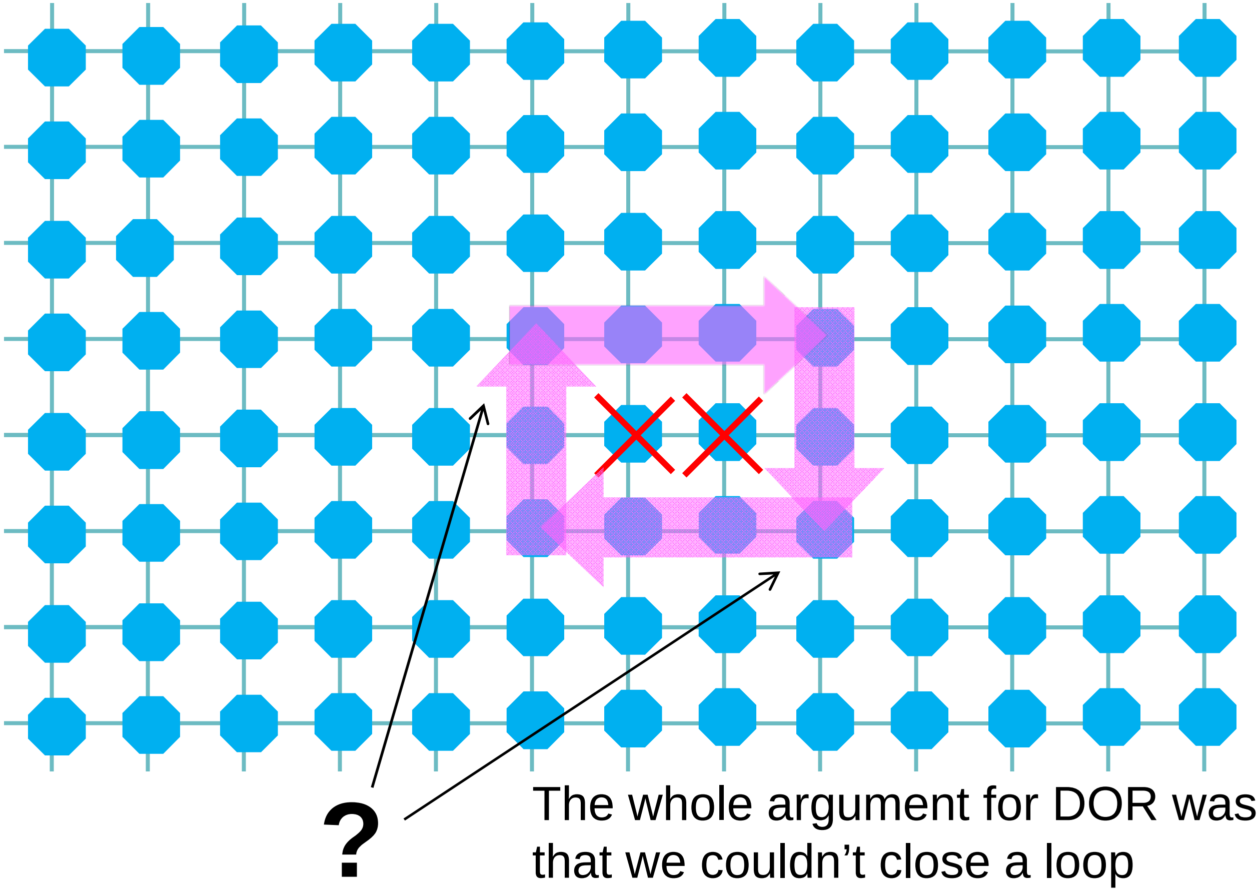
2-torus



Has 2 independent types of credit loops along X and Y dimensions

Deadlock - Example 3

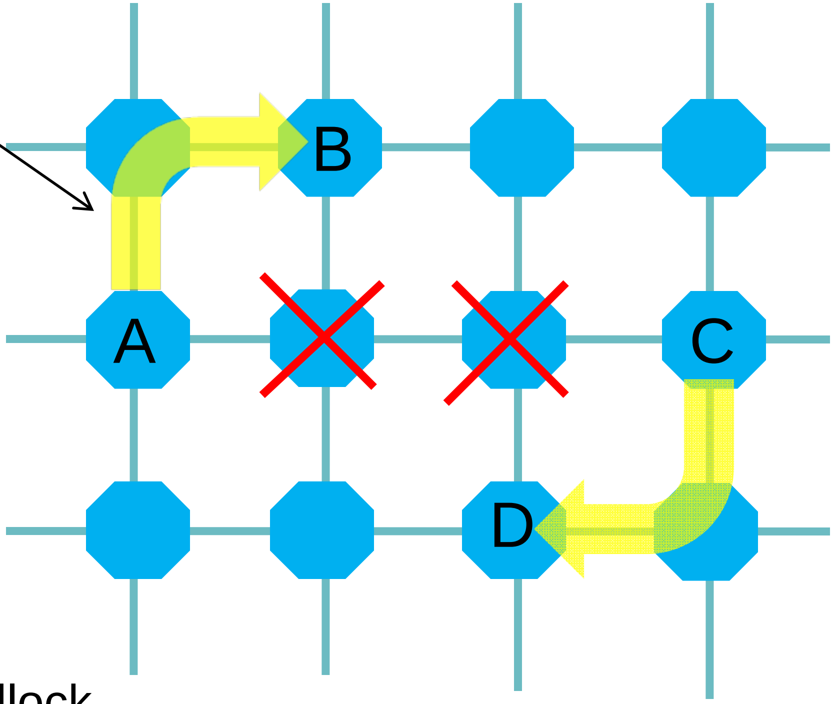
Mesh defect
broken links
or switches



The whole argument for DOR was
that we couldn't close a loop

Deadlock - Example 3

Unique shortest path



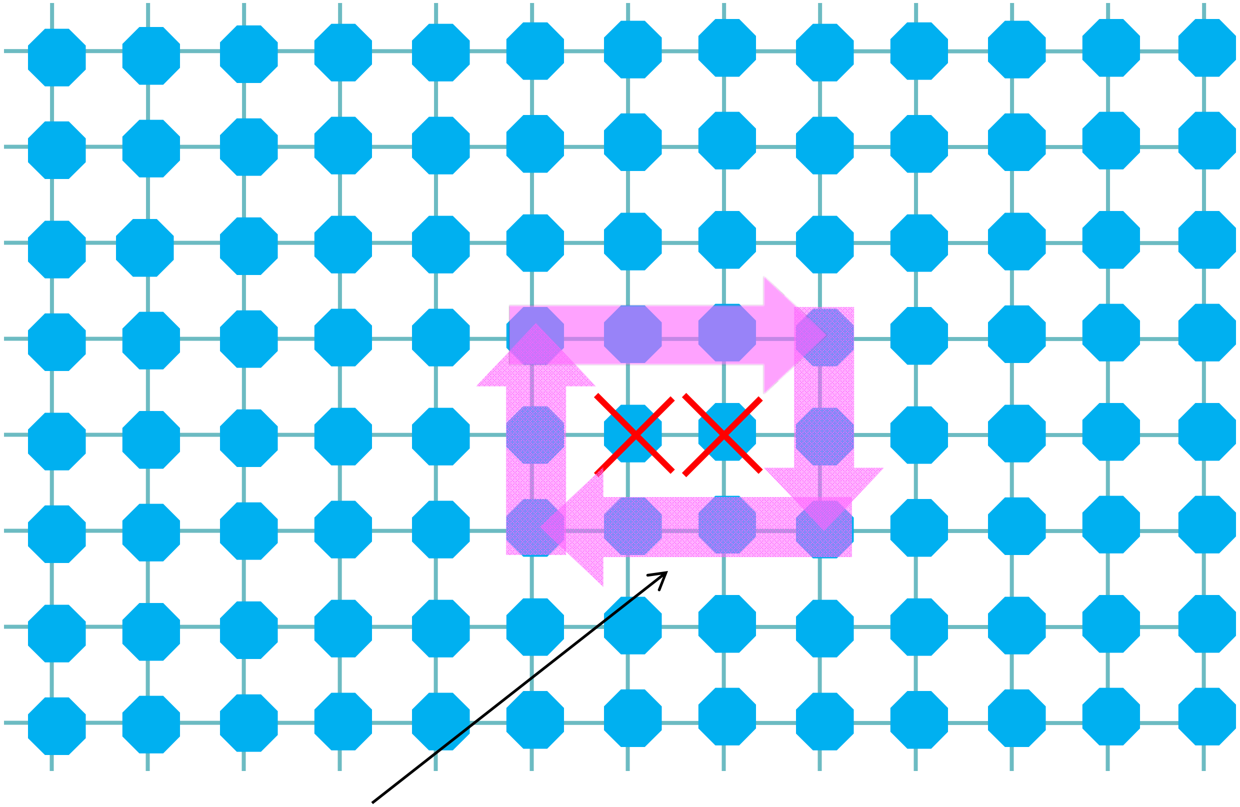
The DOR path no longer exists and any remaining path from A to B can contribute to a deadlock

Have choice of:

- NO path from A to B
  (OK in dual rail fabric)
- path that can cause a deadlock

Deadlock - Example 3

Mesh defect
broken links
or switches

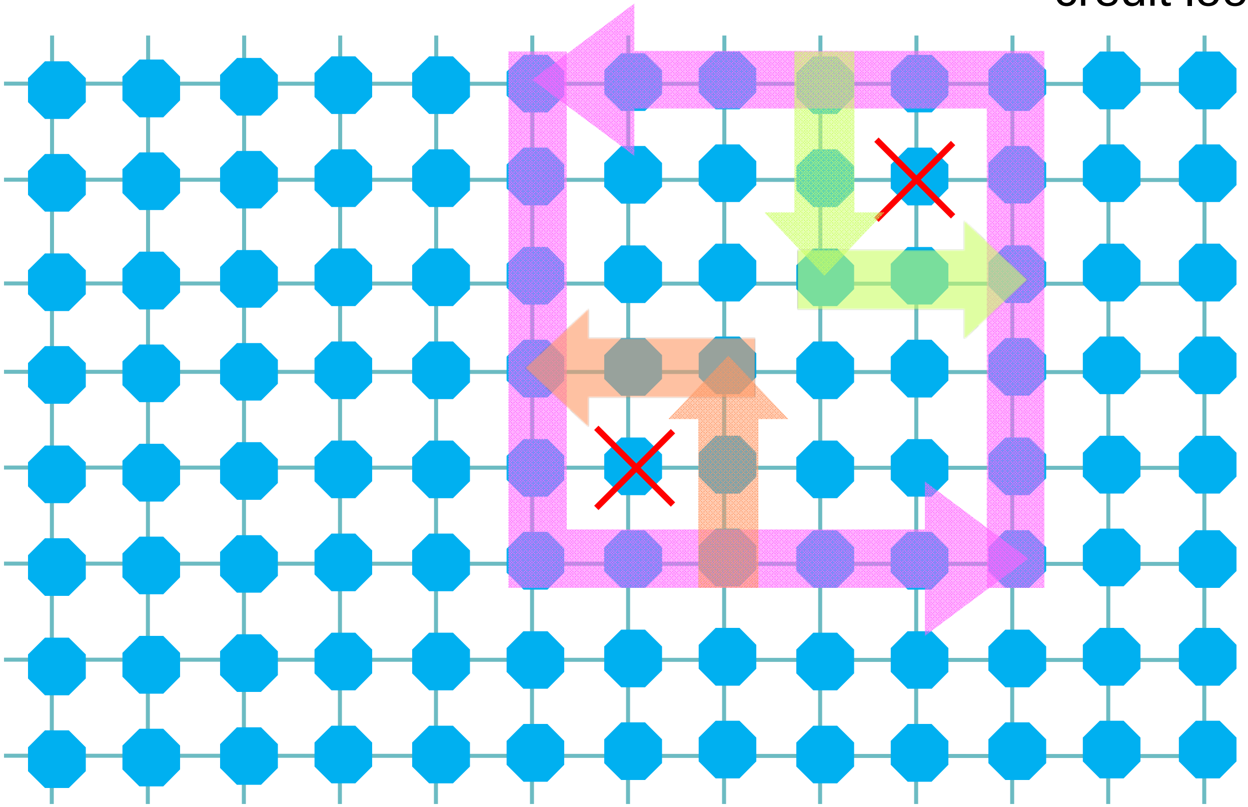


Deadlock possible!

Deadlock - Example 4

Mesh with multiple defects
broken links
or switches

Can have many
credit loops



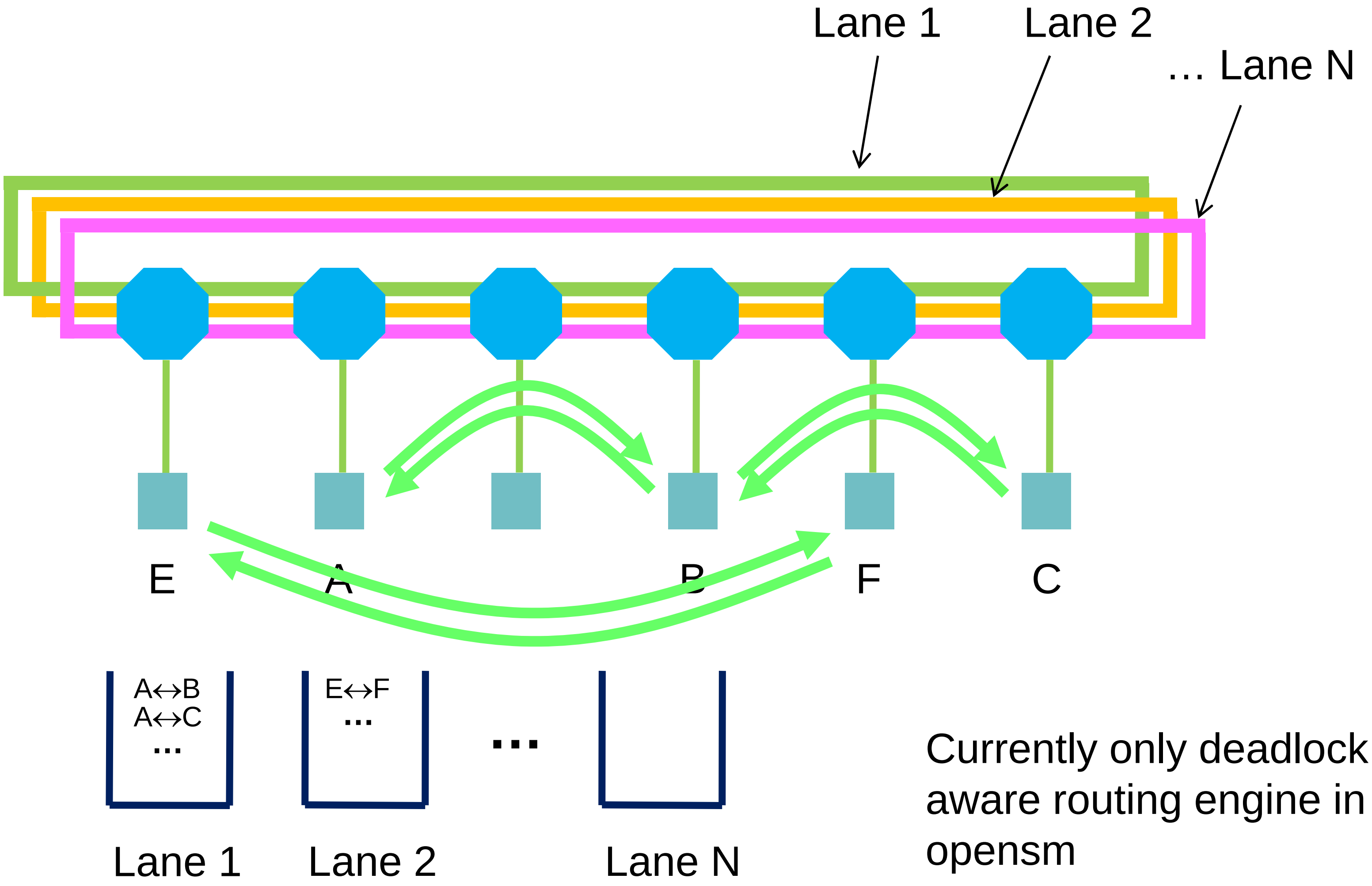
Summary

- Large scale structures and local defects can lead to credit loops even if DOR based routing is used
- Defects force DOR to be violated or paths to be missing.

- However, we can eliminate credit loops by using VLs
  - VLs add additional buffer queue resources

DEADLOCK AVOIDANCE WITH LASH

Lash

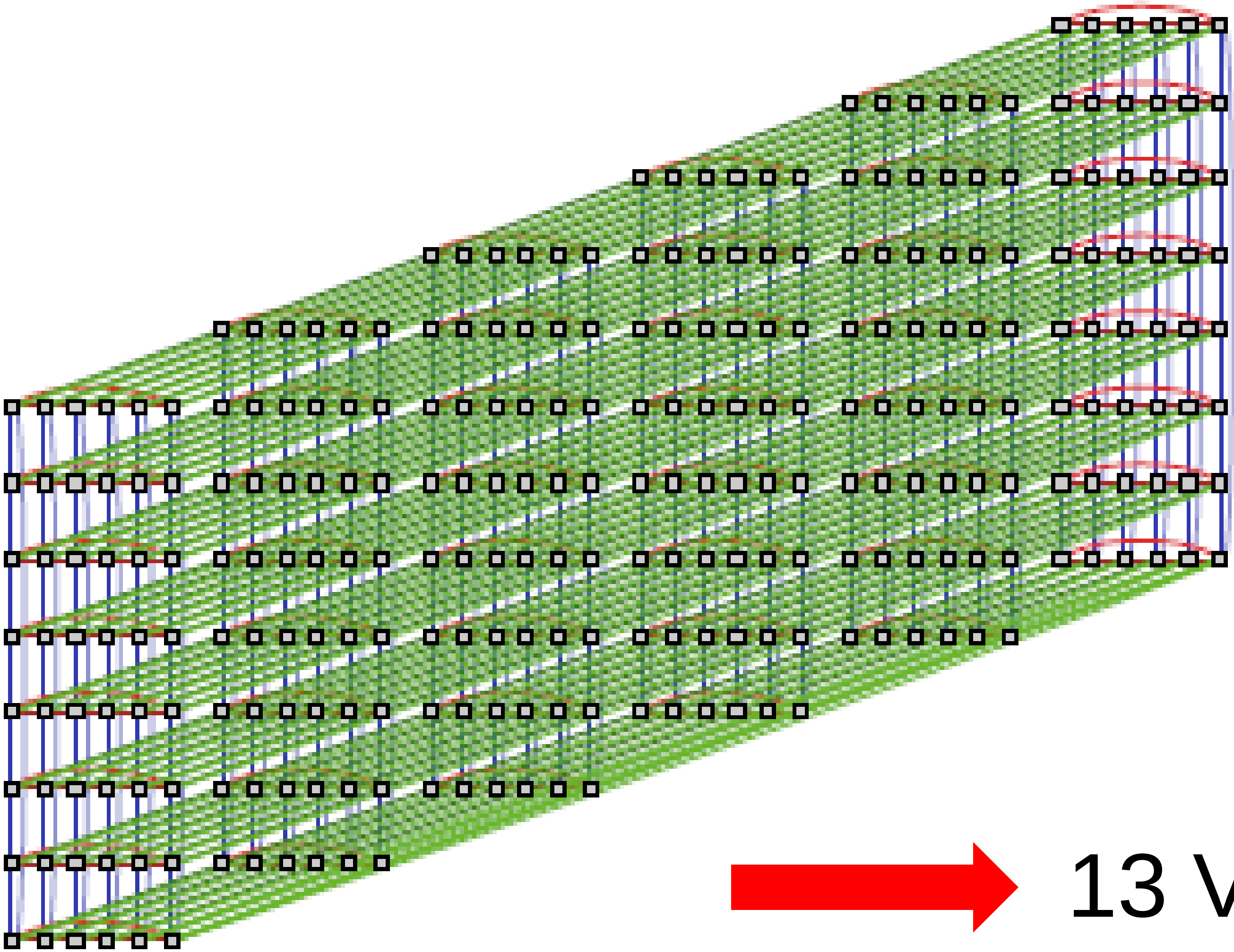


Currently only deadlock aware routing engine in opensm

Summary

- Shortest paths between points are assigned to lanes (SL=VL so no distinction) in an arbitrary order (greedy, depends on order found)
- Lane assignment is communicated through SA path records, nothing is changed in the switches, honor system
- Reverse paths must be in same lane (lane(A->B) = lane(B->A)) by IBA reqmt.
- As many lanes as required are used until you run out
- Algorithm limited by number of VLs (8 for ConnectX)

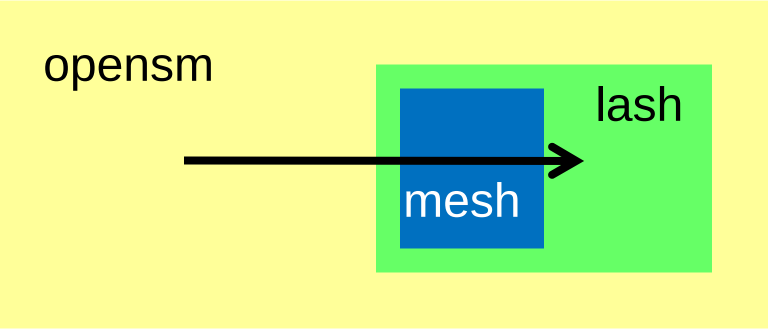
Lash for Redsky?



13 VLs!!

Mesh Patch for Lash

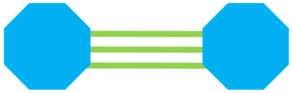
- Handle multiple links between switches
- Discover geometry automatically
- Re-sort links to DOR order with sign inversion
- Re-sort switches to dimension order

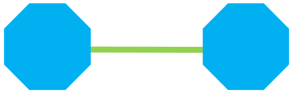


- Mostly upstream now

Mesh Algorithm (1)

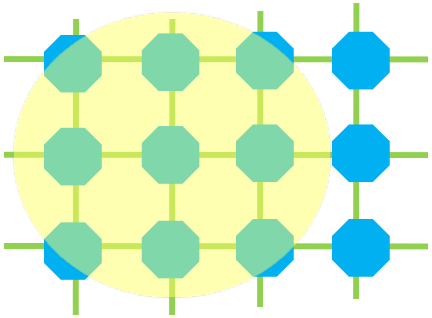
1.





(Link groups)

2.

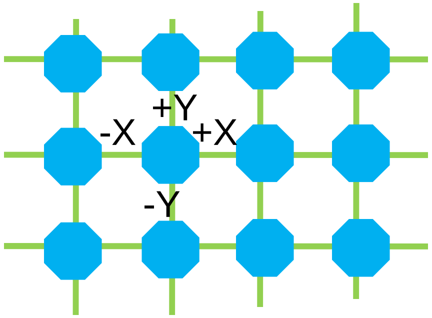


Local shapes
(char poly of local metric)

3. Local shapes

Dominant shape
(e.g. 6x6x8 torus)

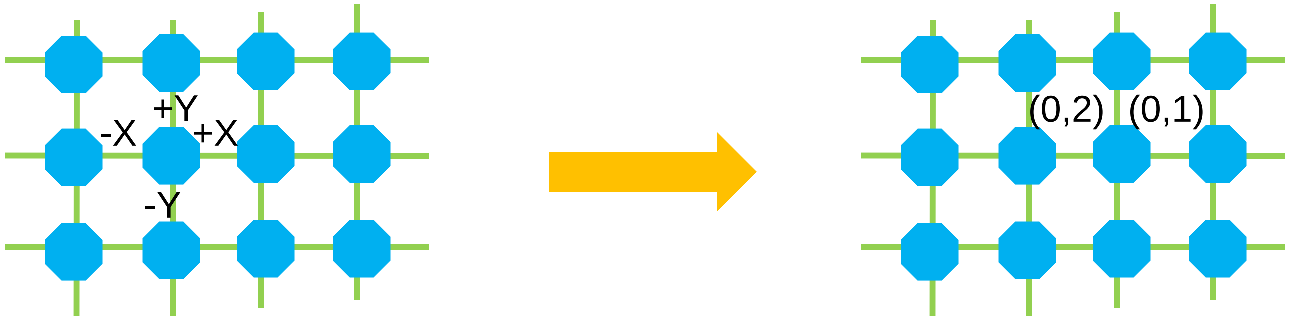
4. Shape+seed



Build
local
coordinates

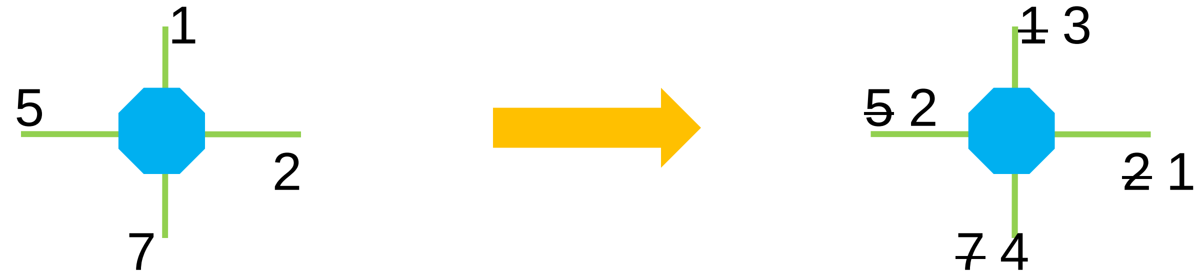
Mesh Algorithm (2)

5.



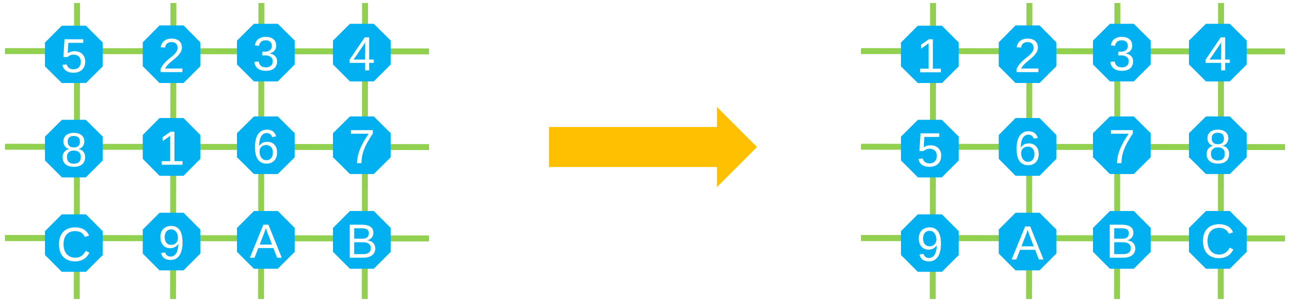
Coordinates & size

6.



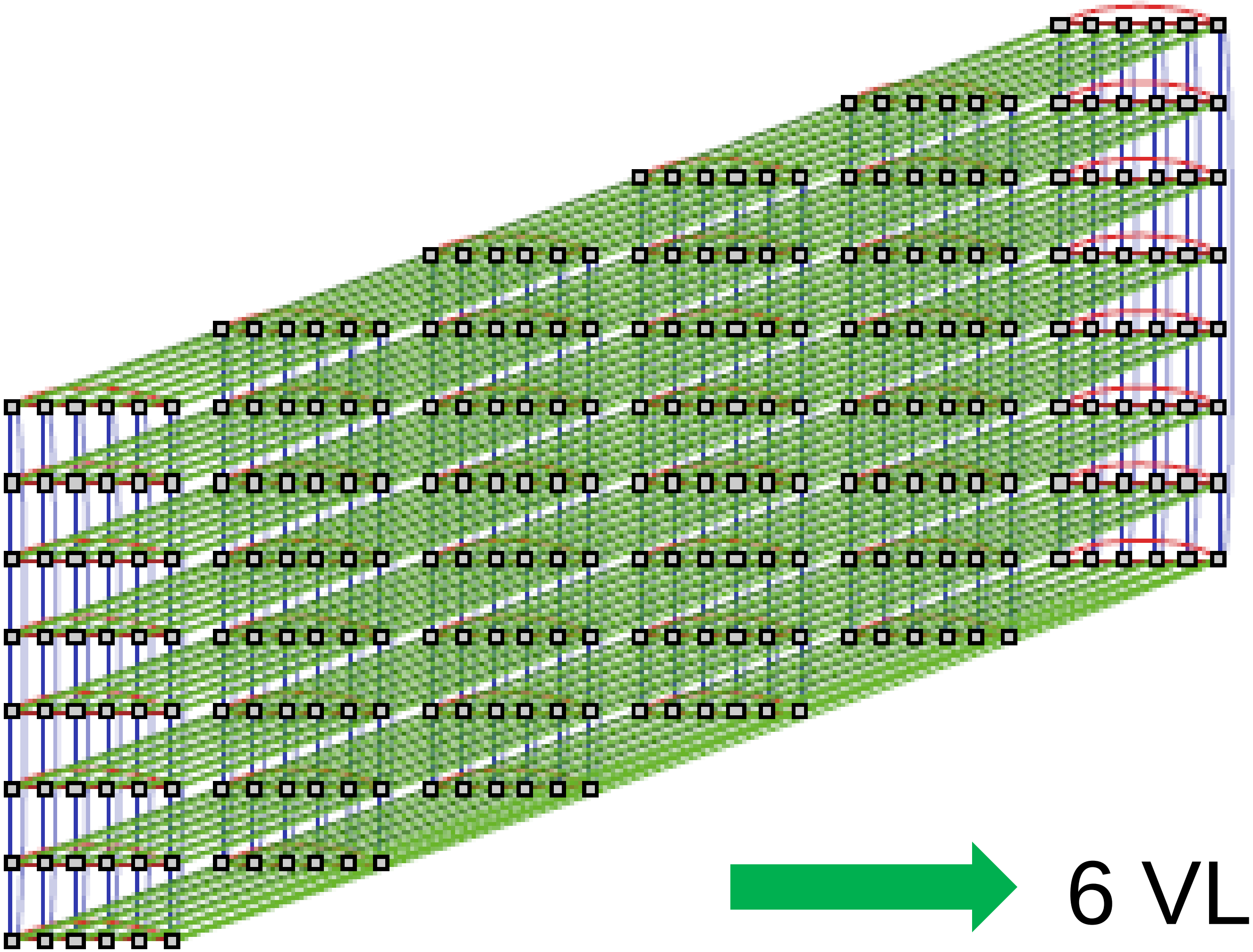
DOR relabeling of ports

7.



Sort switches in odometer order

Lash+Mesh for Redsky



6 VLs

Lash+Mesh with H/W Failures

- Current algorithm tested against:
  - All single/double link failures
  - All single switch & switch pair failures
- Result:
  - All single link failures & switch failures route in 7 VLs or less
  - 0.2% of double link failures route in 8 VLs
  - 0.04% of double link failures route in 9 VLs
- We are working to improve this (better seed choice)

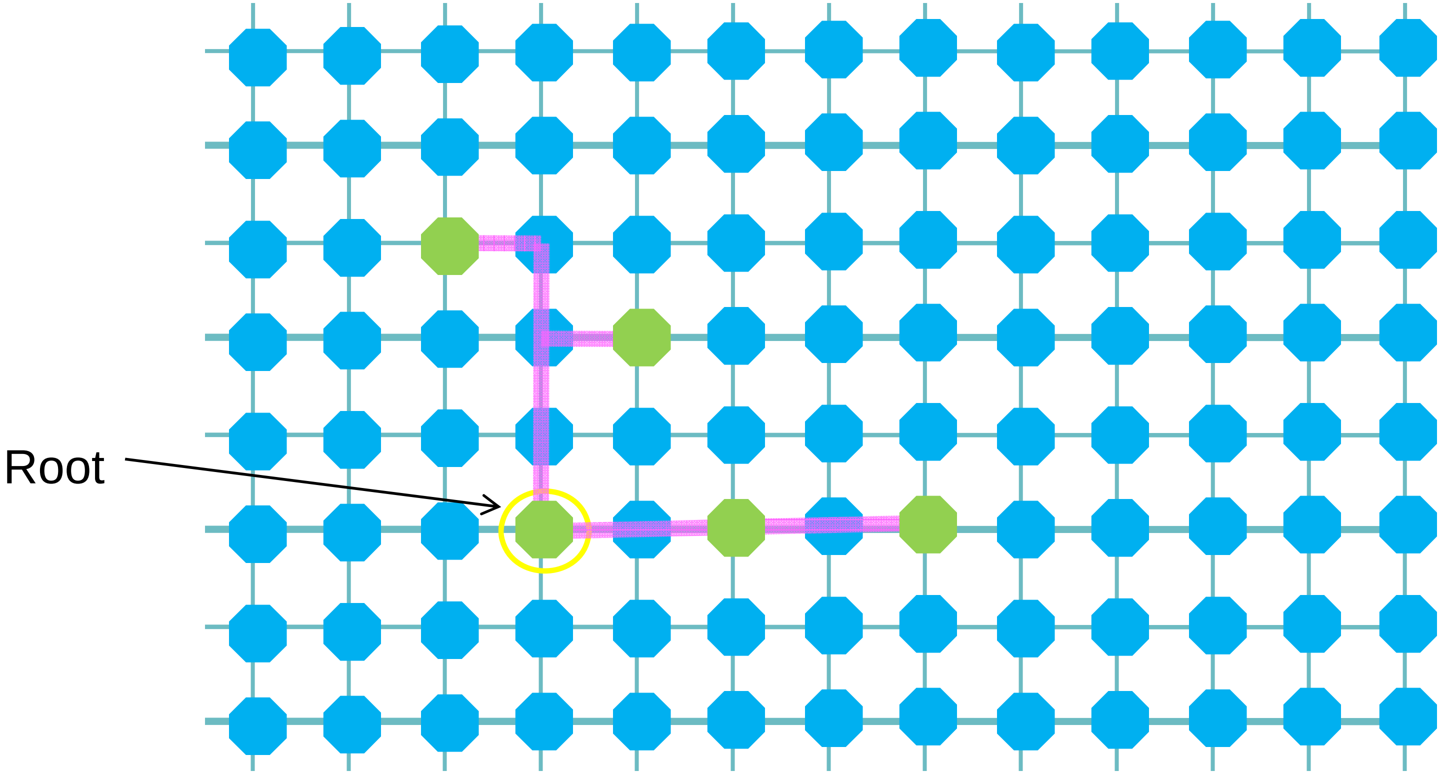
Lash+mesh size scaling issues

- As size of 3D torus increases number of VLs increases up to a limit
  - 2-3 X increases in size lead to 7 VLs
  - Really big 3D torus leads to 8 VLs
  - HW failures are extra!

MULTICAST DEADLOCK ISSUES

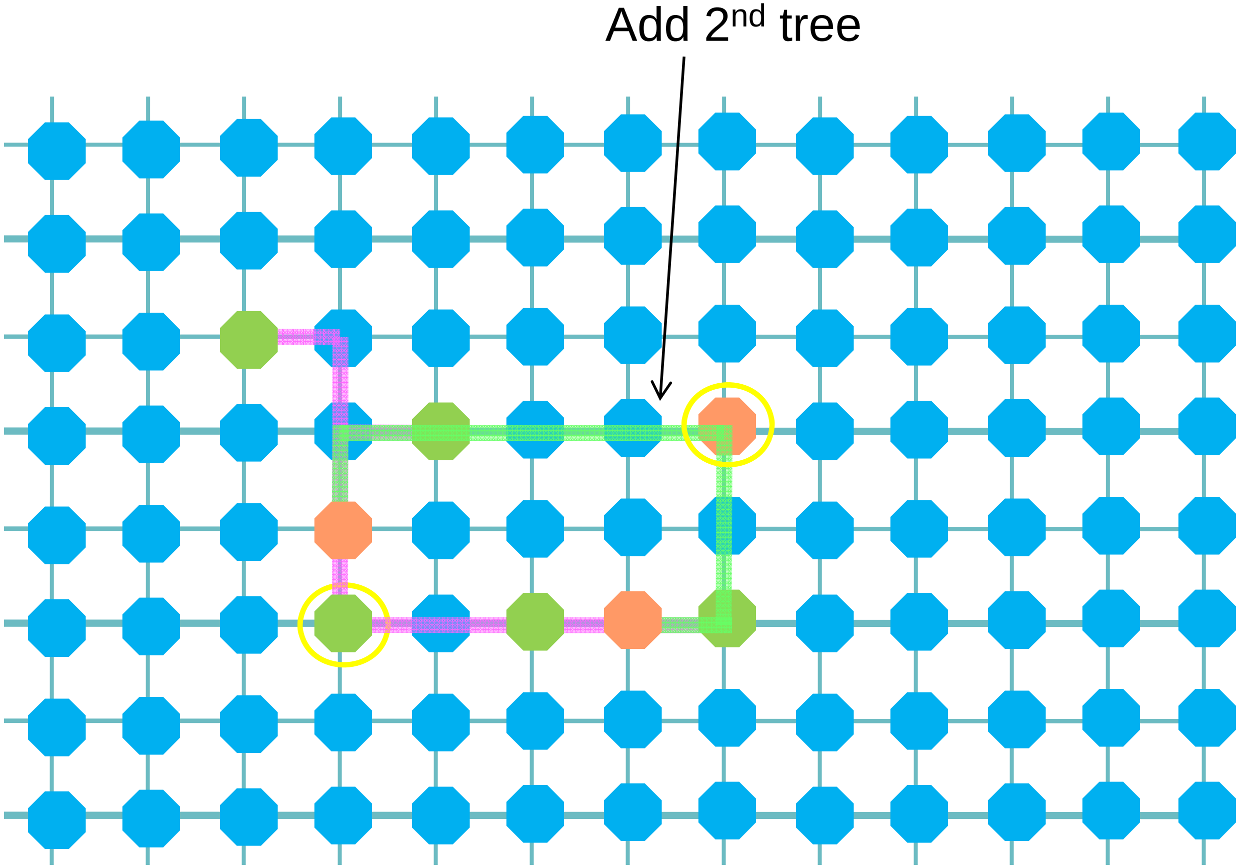
Multicast

(a la opensm)



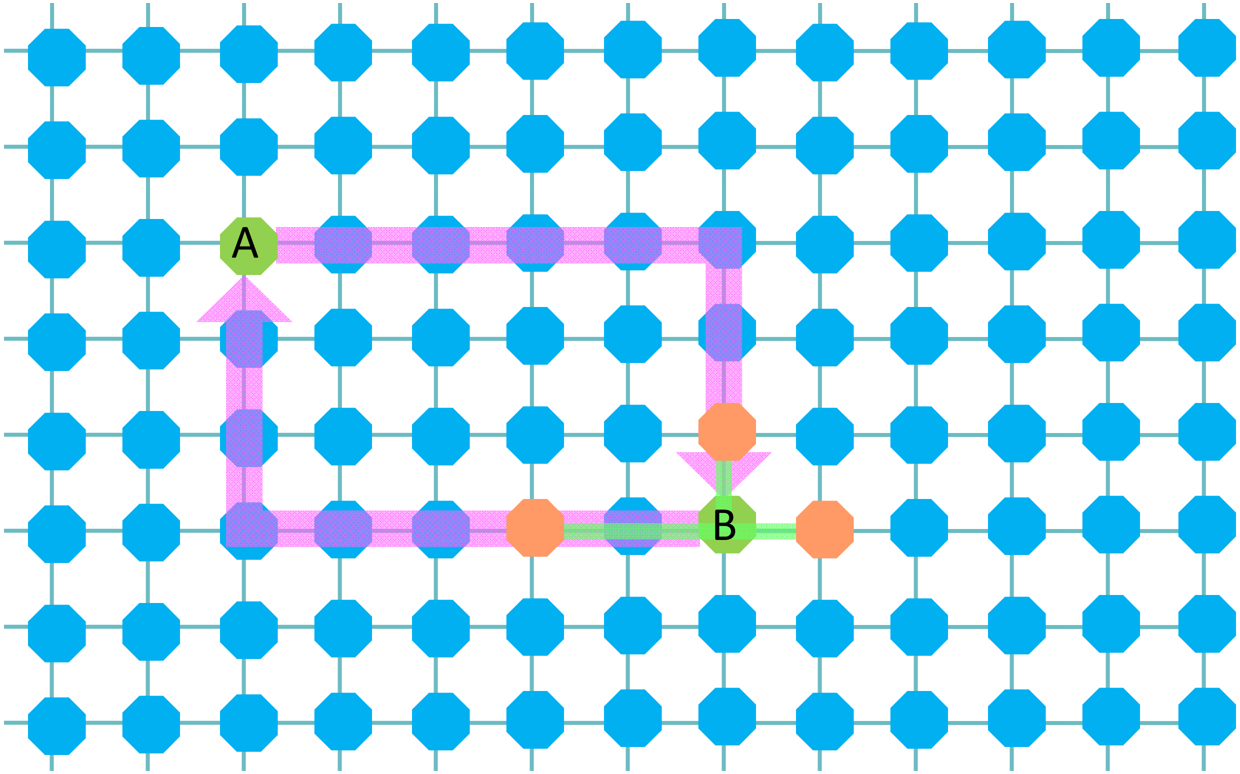
Path discovery is arbitrary based
On how the fabric is discovered

Multicast



Recall traffic is bi-directional on paths &
We have a closed credit loop!!

Multicast



Multicast

- Challenges
  - Multicast can deadlock against other multicast traffic
  - Multicast can deadlock against unicast traffic
- Solution ideas
  - Route all multicast groups using sub-trees of a single spanning tree, can choose spanning tree that does not deadlock against DOR!!
  - Collapse overlapping multicast groups into common spanning tree, (but IPoIB MC group spans every node so reduces to first answer)
  - Move multicast to separate SL/VL

TYING IT ALL TOGETHER

Day One Redsky Configuration

- Lash+Mesh for “Good” unicast traffic SL=VL=1-6/7
  - Good apps plays by the rules, uses CM for SL determination
  - Tolerates many failures
- SL=VL=0 for “Bad” unicast traffic
  - Bad apps do not, many examples in stack
  - Can map SL0 to VL15 which drops packets
- Multicast SL=8, VL=0

What Works

- All ULPs and mvapich and OpenMPI optionally are “good” apps
- Layout up and running on exemplar system

Work Remaining

- Change multicast routing algorithm to avoid deadlocks between MC groups
- Clean up remaining “Bad” apps
  - Many MAD based apps, SMSL usage, some point tools
- Improve link failure VLs in mesh
- Integrate Lash and QoS subsystems
  - Currently broken, needs more flexible management
- Graceful failure recovery
  - Lash incremental re-route, prevent large # of SL changes on HW failures
  - Implement unpath, repath or something else to kill paths on SL change

FUTURE DIRECTIONS

New Routing Algorithm

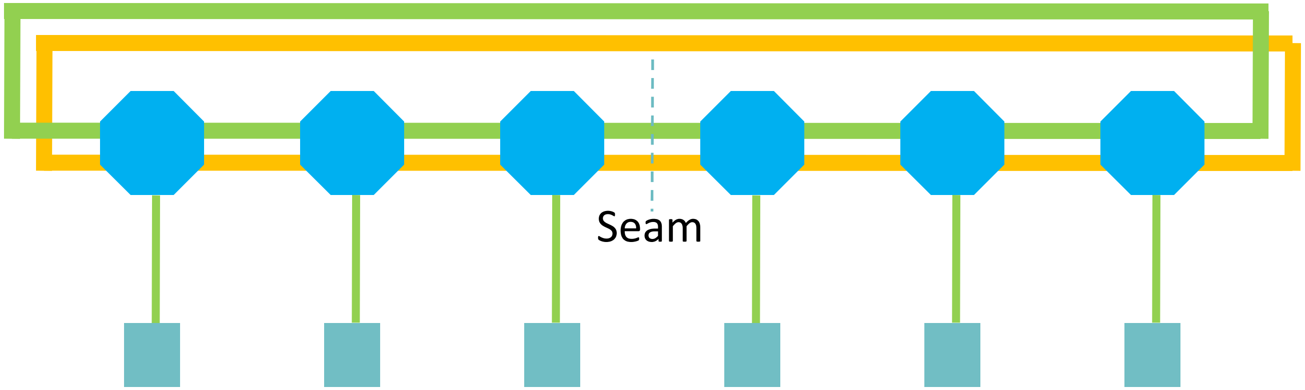
- 8 SLs and 2 VLs, explicit torus based algorithm
- Can create 2 QoS groups (uses all 16 SLs)
  - Use LMC=1 to create two DOR based routings, ‘dual rails on one fabric’
  - Compatible multicast spanning tree

Work being done by Eitan Zahavi, Jim Schutt, Sven Arne and others

BACKUP SLIDES

Algorithmic Approach

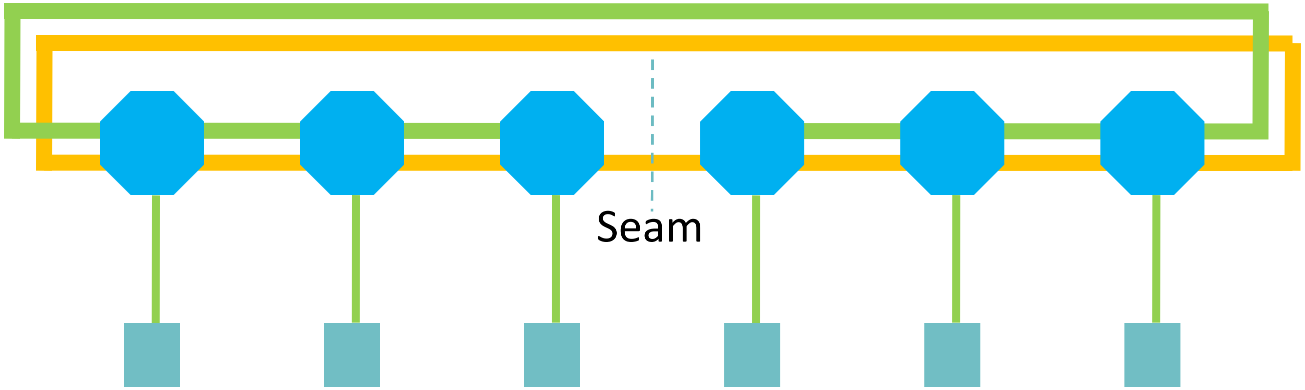
1D example



Take shortest paths to destination (left or right)
if path does not cross ‘seam’ take one VL
if path crosses ‘seam’ take the other VL

Algorithmic Approach

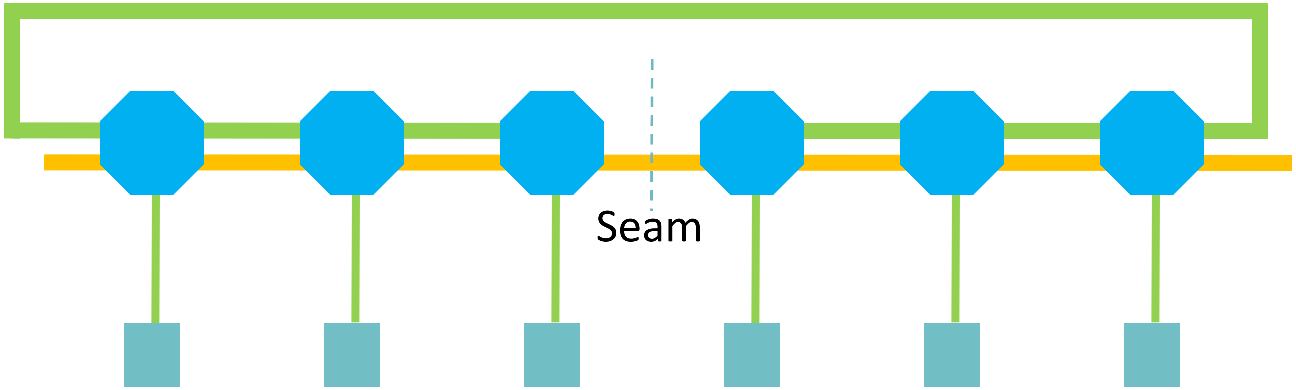
1D example



Take shortest paths to destination (left or right)
if path does not cross ‘seam’ take one VL
if path crosses ‘seam’ take the other VL

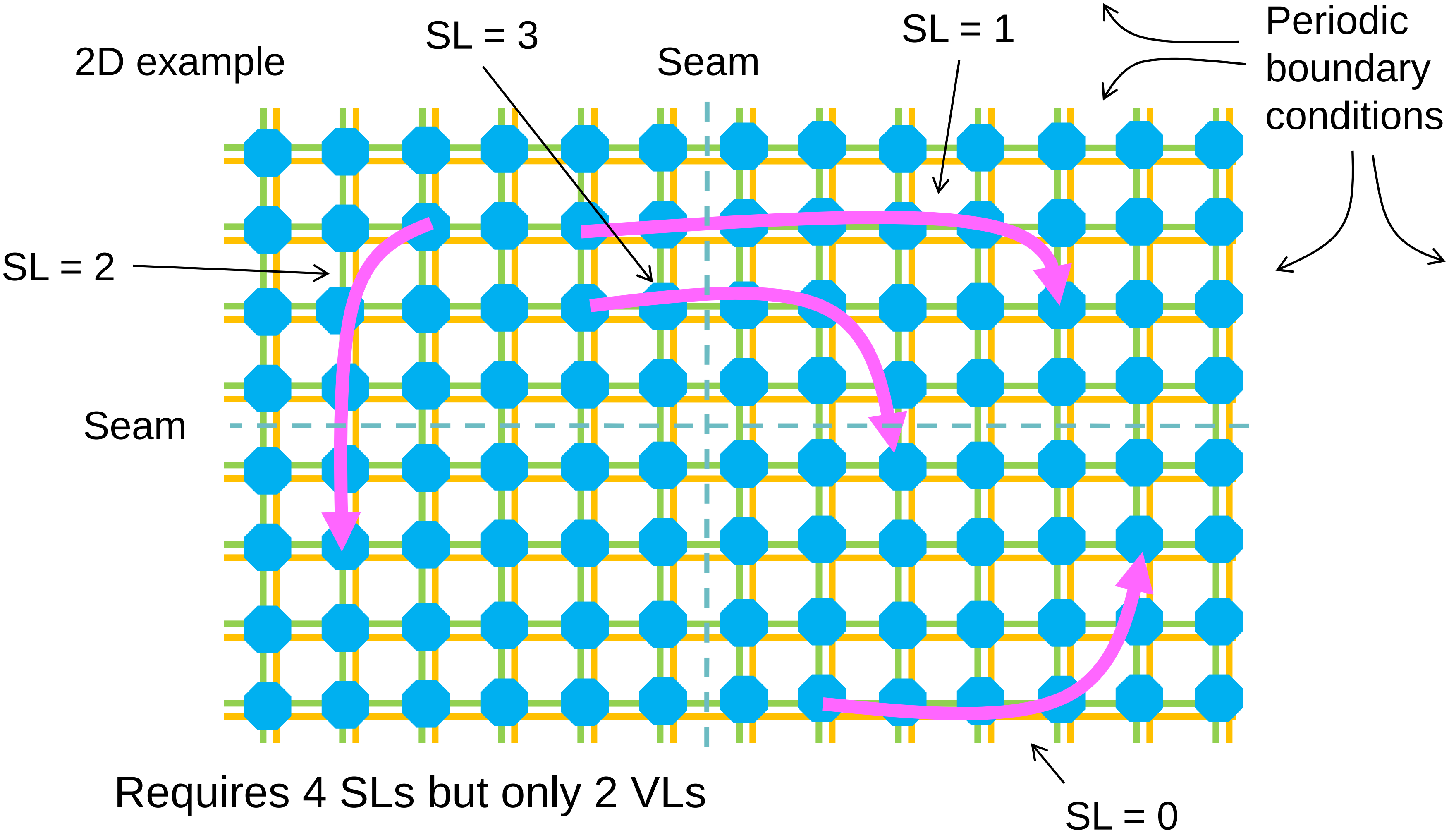
Algorithmic Approach

1D example



Take shortest paths to destination (left or right)
if path does not cross ‘seam’ take one VL
➡ if path crosses ‘seam’ take the other VL

Algorithmic Approach

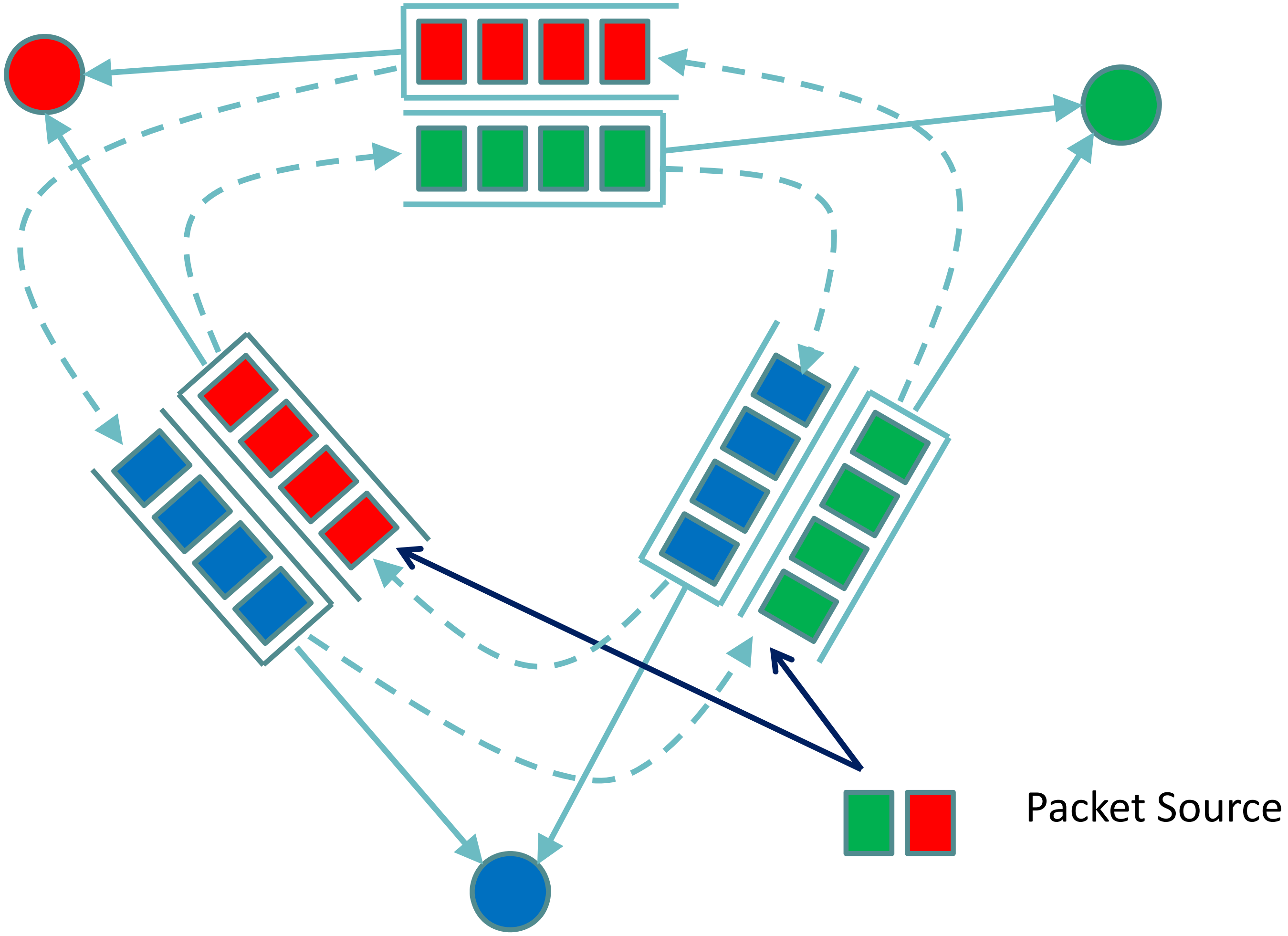


Requires 4 SLs but only 2 VLs

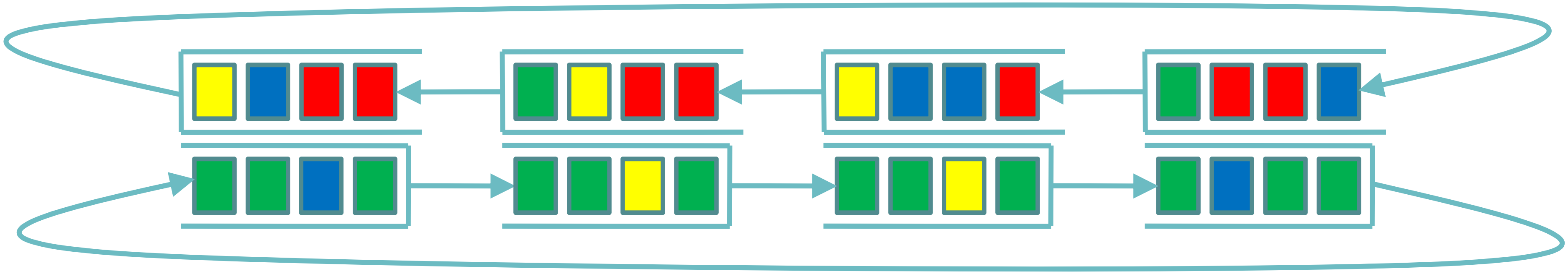
Algorithmic Approach

- Choice of whether to use alternate lane is encoded in SL. Requires 2^D SLs for a D dimensional torus
- Only 2 VLs are required!

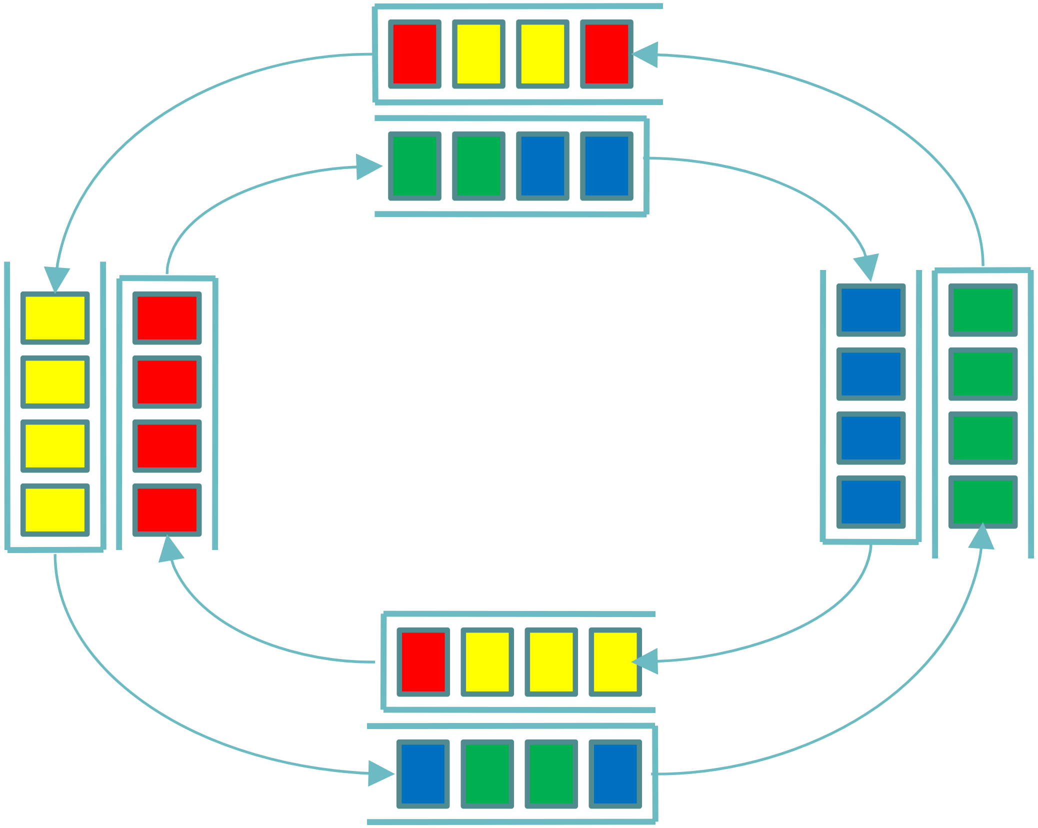
L=3 never deadlocks



L=4 isomorphic to 2x2



L=4 with DOR never deadlocks



L=4 isomorphic to 2x2

tori with L=4

- 2x4
- 4x5
- 4x4x7

Is equivalent to…

- 2x2x2
- 2x2x5
- 2x2x2x2x7