ECE 8823 A / CS 8803 - ICN

Interconnection Networks

Spring 2017

http://tusharkrishna.ece.gatech.edu/teaching/icn_s17/

Lecture 6: Deadlocks - II

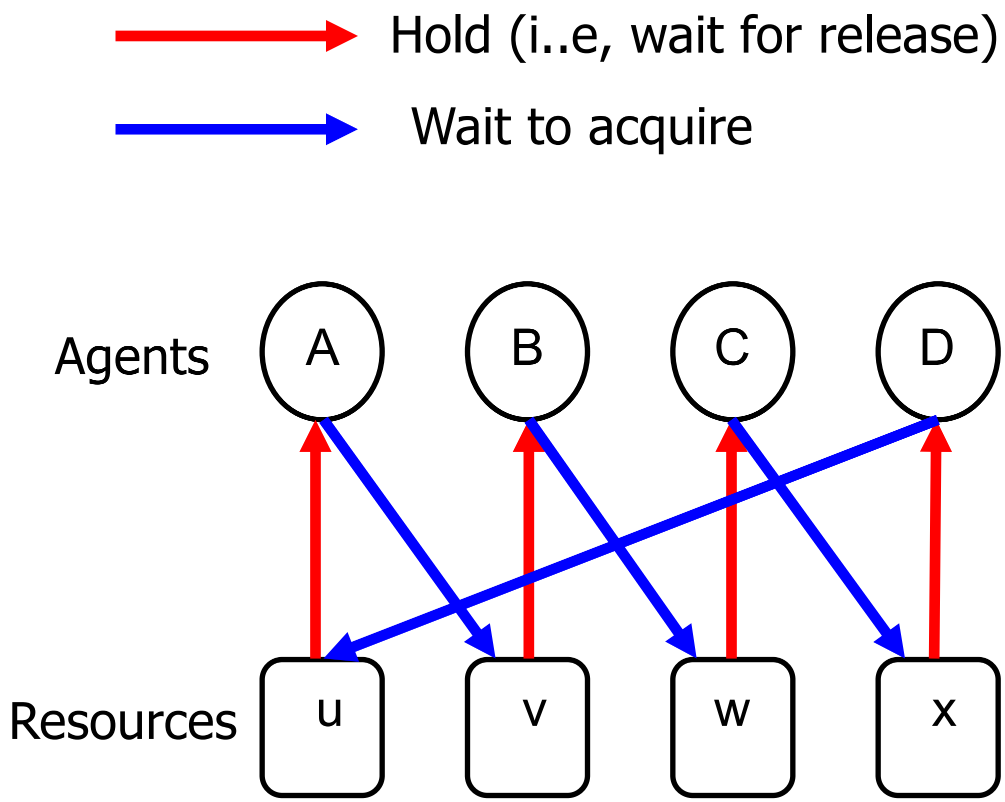
Tushar Krishna

Assistant Professor
School of Electrical and Computer Engineering
Georgia Institute of Technology

tushar@ece.gatech.edu

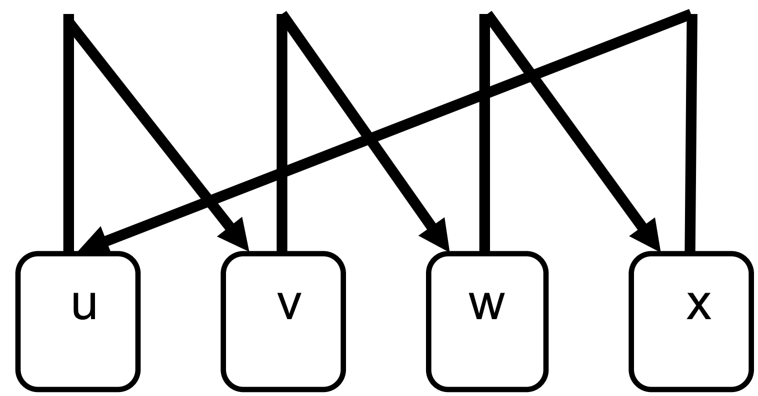
Recap: Resource Dependence

Resource A is ***dependent*** on resource B if it is possible for A to be *held-by* an agent X and it is also possible for X to *wait-for* B



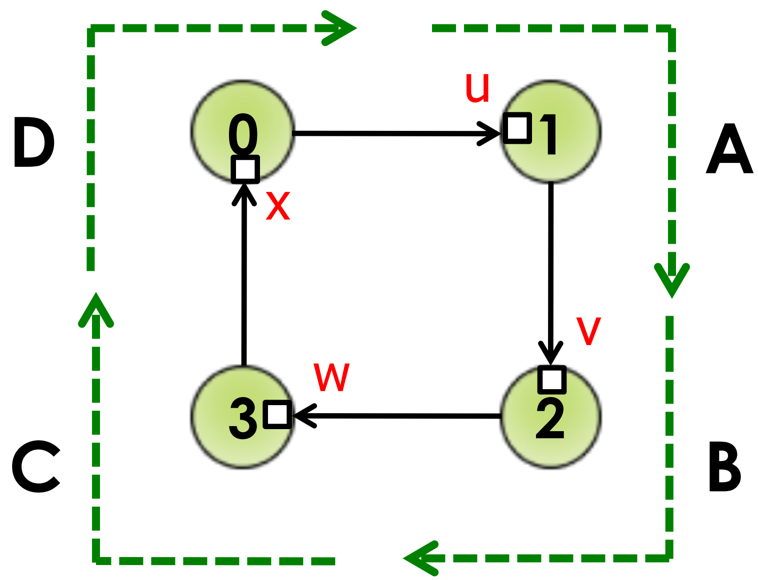
To avoid deadlocks, resource dependence graph should be acyclic

Resource Dependence Graph



Recap: Network Deadlock

- **Agent = packet or flit, Resource = buffer**



We will use the following convention:
channel == buffer at end of channel

Channel 01 blocked =>
Buffer at 1 not free

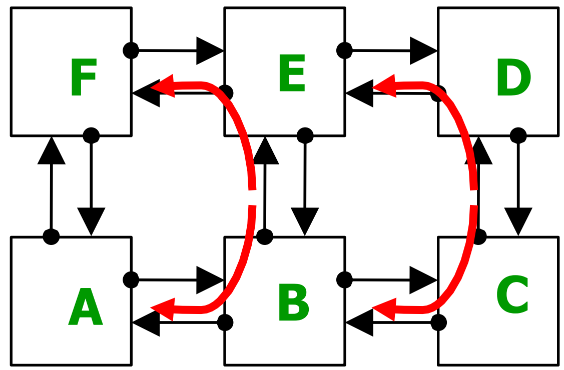
Channel 01 depends on Channel 12 =>
Buffer at 1 depends on Buffer at 2

- Packet A holds buffer u (in 1) and wants buffer v (in 2)
- Packet B holds buffer v (in 2) and wants buffer w (in 3)
- Packet C holds buffer w (in 3) and wants buffer x (in 0)
- Packet D holds buffer x (in 0) and wants buffer u (in 1)

Deadlock Avoidance

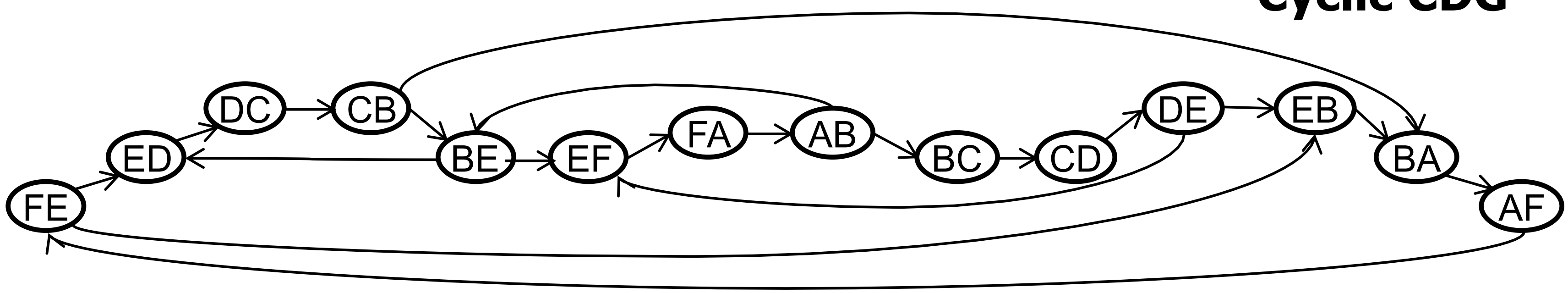
- Eliminate cycles in Resource Dependency Graph
  - Resource Ordering
    - Enforce a partial/total order on the resources, and insist that an agent acquire the resources in ascending order
    - Deadlock avoided since a cycle must contain at least one agent holding a higher numbered resource waiting for a lower-numbered resource which is not allowed by the ordering allocation
  - Implementation
    - Restrict certain routes so that a higher numbered resource cannot wait for a lower numbered resource
    - Partition the buffers at each node such that they belong to different **resource classes**. A packet only any route can only **acquire buffers in ascending order of resource class**

Example: Channel Dependency Graph for Mesh

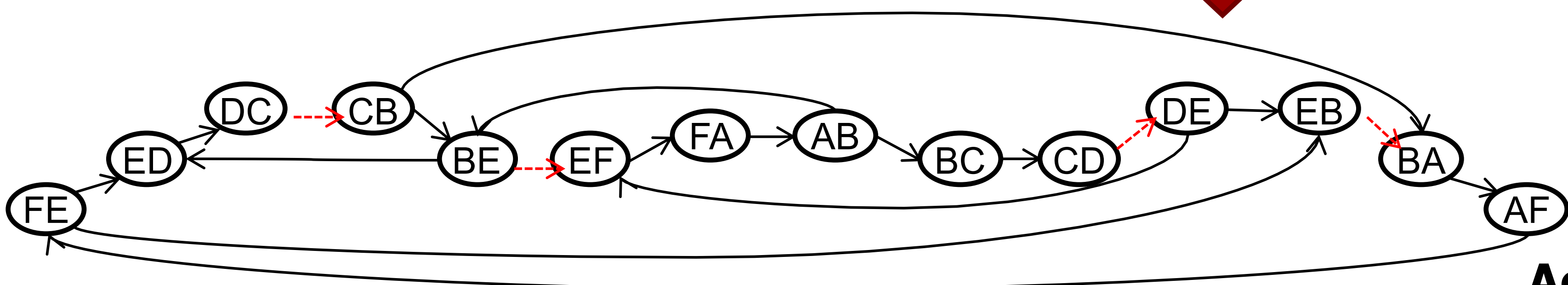


This is the West-first turn model!

Cyclic CDG

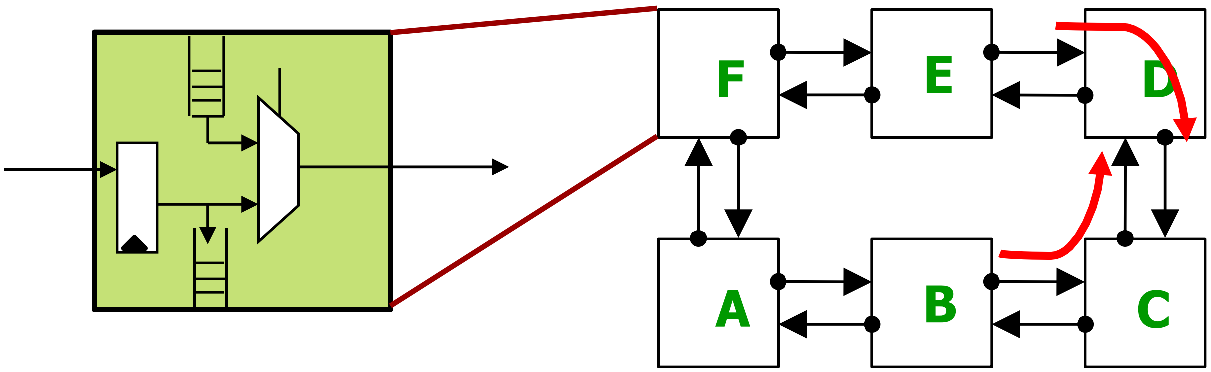


Disable certain edges



Acyclic CDG

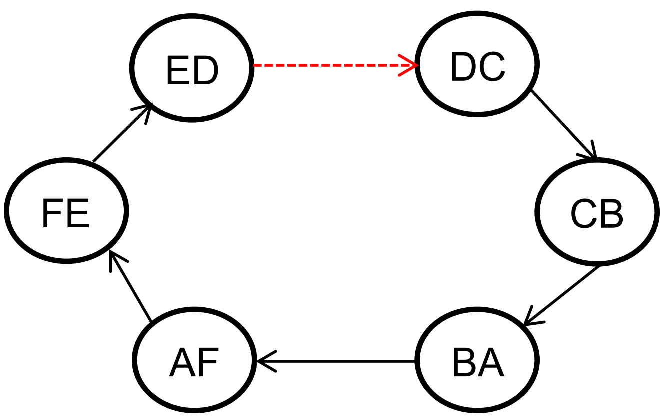
Acyclic CDG for a Ring



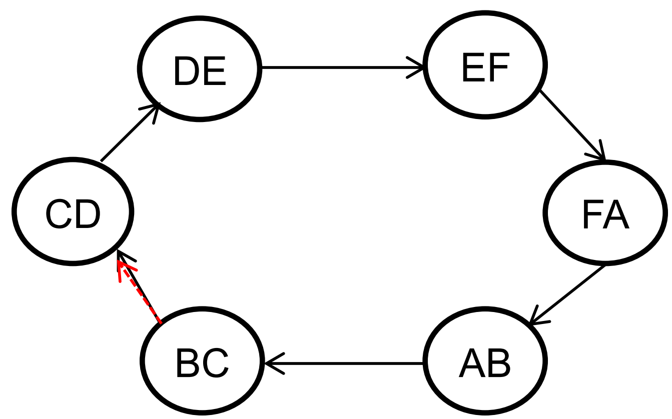
Route from E to C disabled
(E to D) and (D to C) allowed

Route from B to D disabled

Option 3



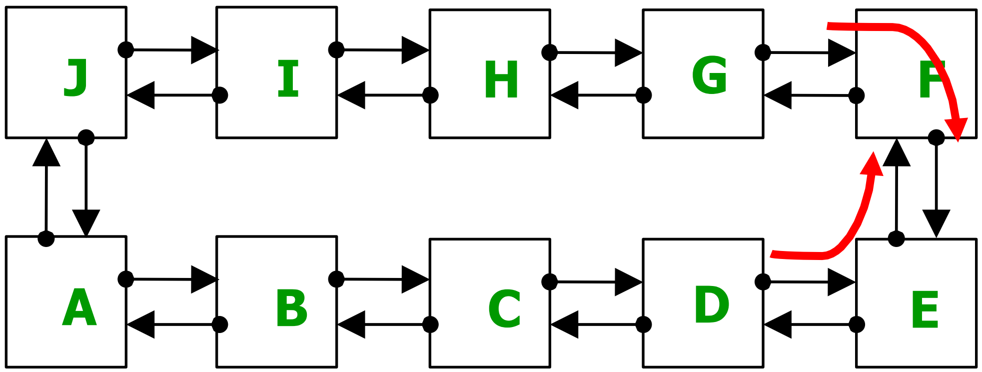
CDG



Acceptable CDG

Problem? E to C no longer minimal

Acyclic CDG for a Large Ring

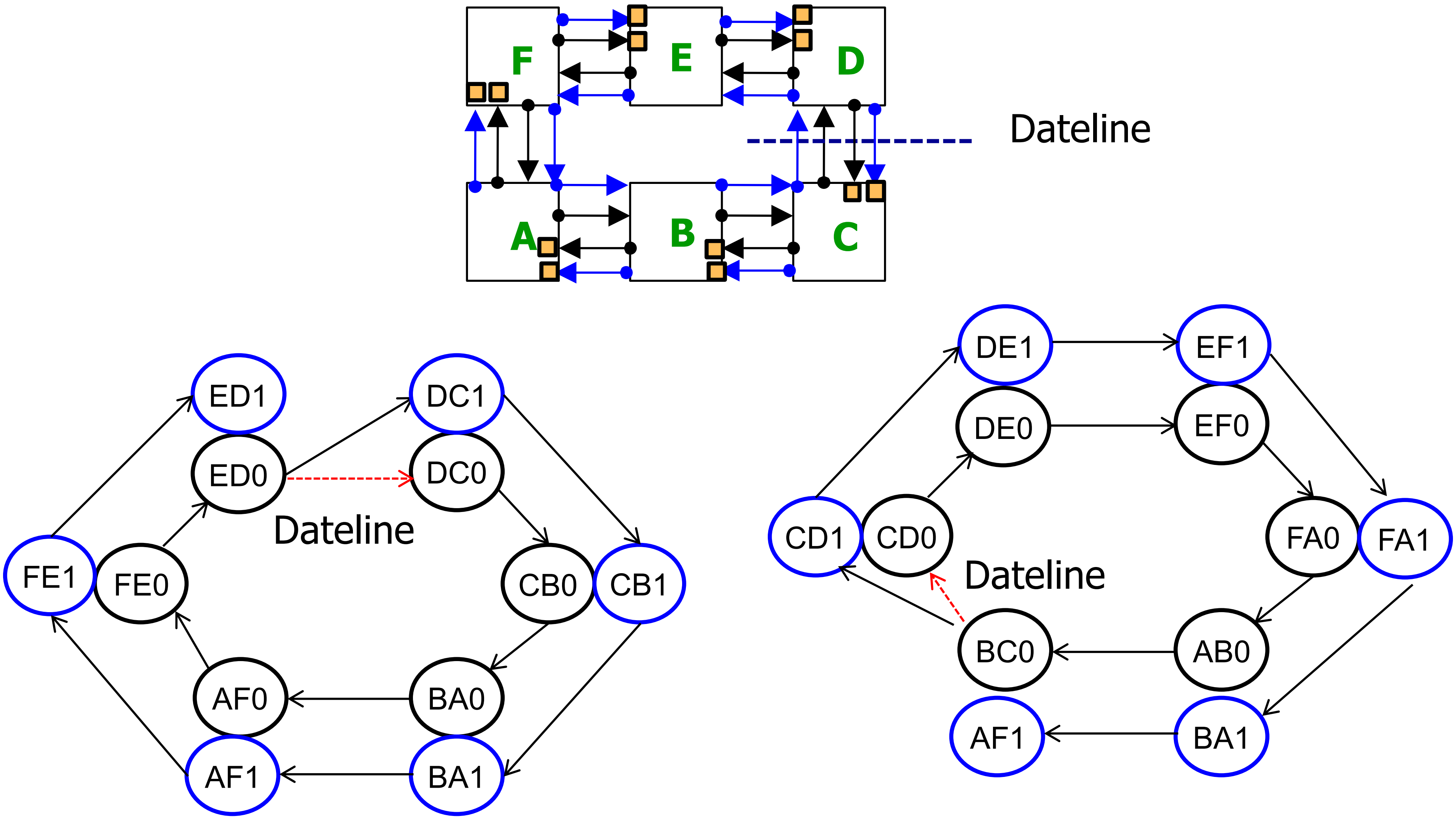


Problem?

G, H, I have to take non-minimal paths to reach E!

D, C, B have to take non-minimal paths to reach F

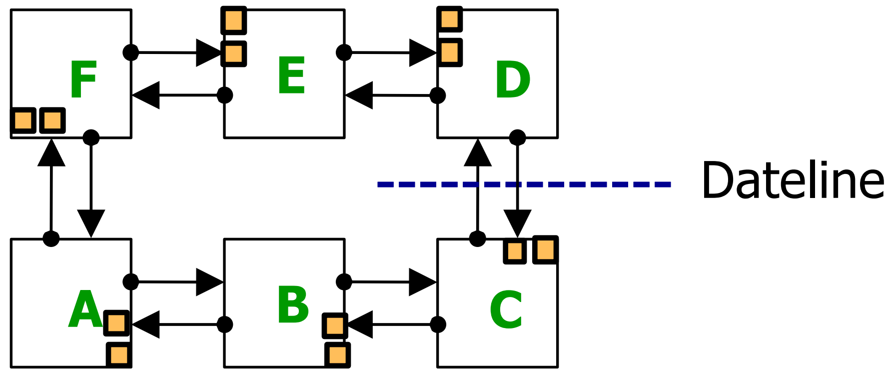
Suppose *two* channels

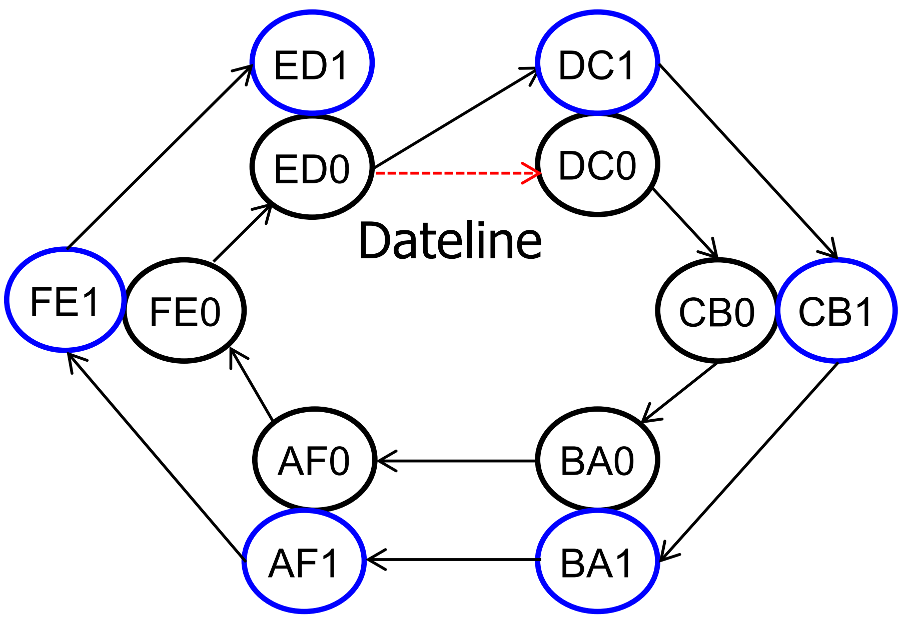


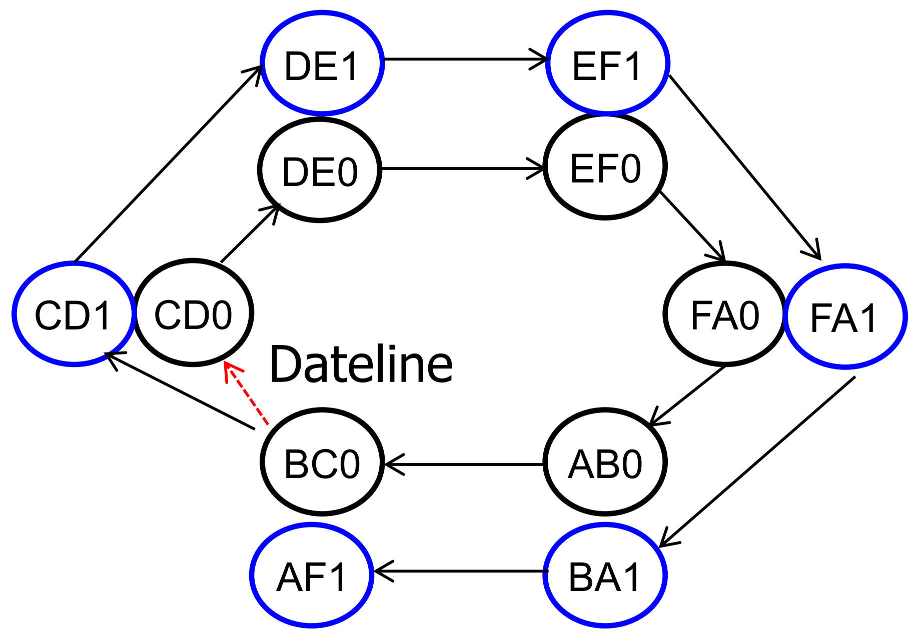
Need not be physical channels

Need at least 2 classes of buffers - called **"Virtual Channels"**

Start in VC in Class0
After Dateline, jump to VC in Class1







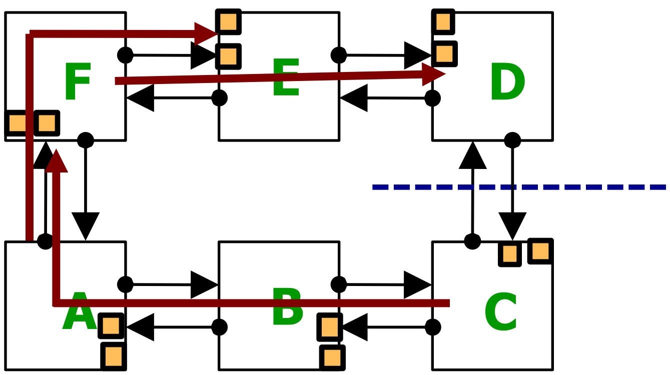
Deadlock Avoidance

- Eliminate cycles in Resource Dependency Graph
  - Resource Ordering
    - Enforce a partial/total order on the resources, and insist that an agent acquire the resources in ascending order
    - Deadlock avoided since a cycle must contain at least one agent holding a higher numbered resource waiting for a lower-numbered resource which is not allowed by the ordering allocation
  - Implementation
    - Restrict certain routes so that a higher numbered resource cannot wait for a lower numbered resource
    - Partition the buffers at each node such that they belong to different **resource classes**. A packet only any route can only **acquire buffers in ascending order of resource class**

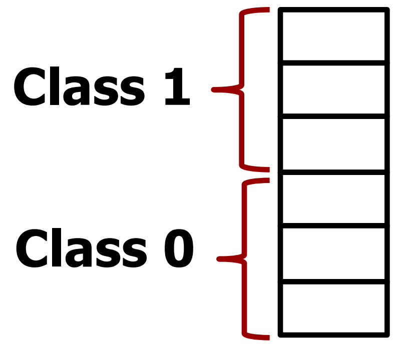
Using VCs for Deadlock Avoidance

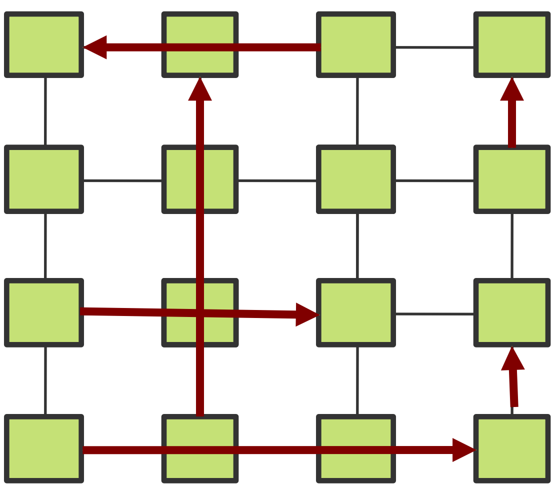
- Ring
  - Use VC from class 0 before dateline
  - Use VC from class 1 after dateline
- Fully-Oblivious (e.g., O1turn)
  - Use VC 0 for XY, VC 1 for YX
- Fully-Adaptive Routing (no turns restricted)
  - Use VC from class 0 before turning
  - Use VC from class 1 after turning
- Valiant's Routing Algorithm
  - DOR over VC in class 0 from source till intermediate
  - DOR over VC in class 1 from intermediate to destination

VC utilization



Ring with Dateline



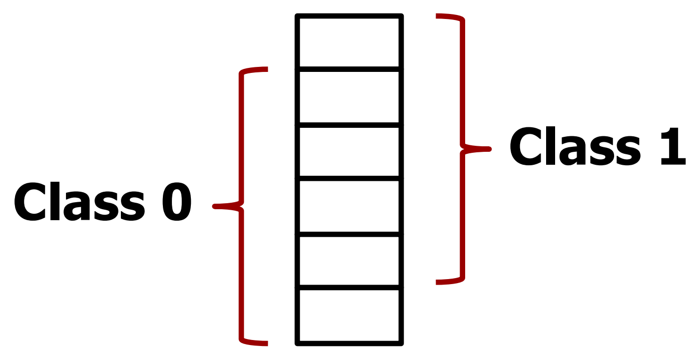


Mesh with O1Turn

Problem? Packet on Ring never crosses dateline
Packet on Mesh does not make any turns

VC from Class 1 never used!

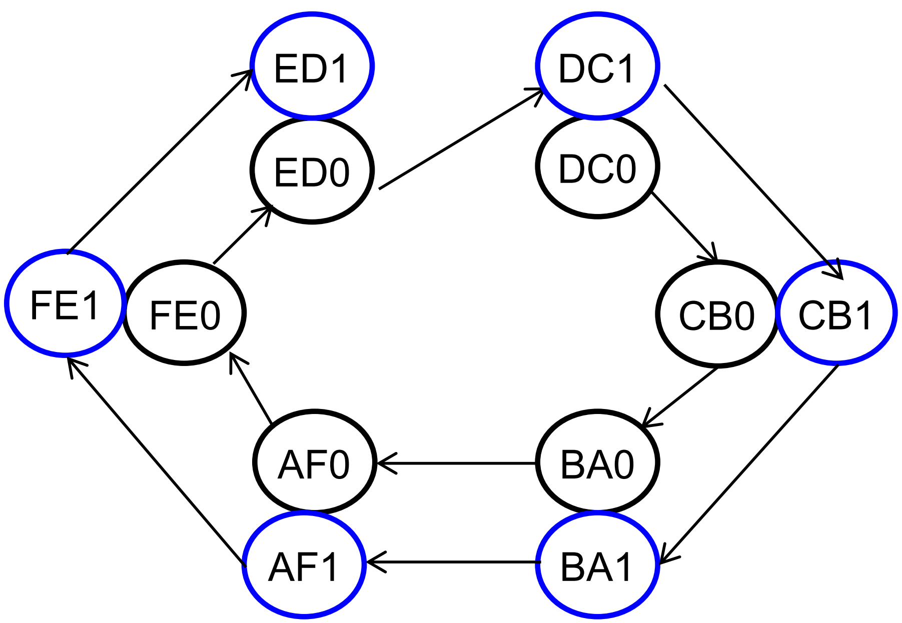
Solution: Overlapping Resource Classes

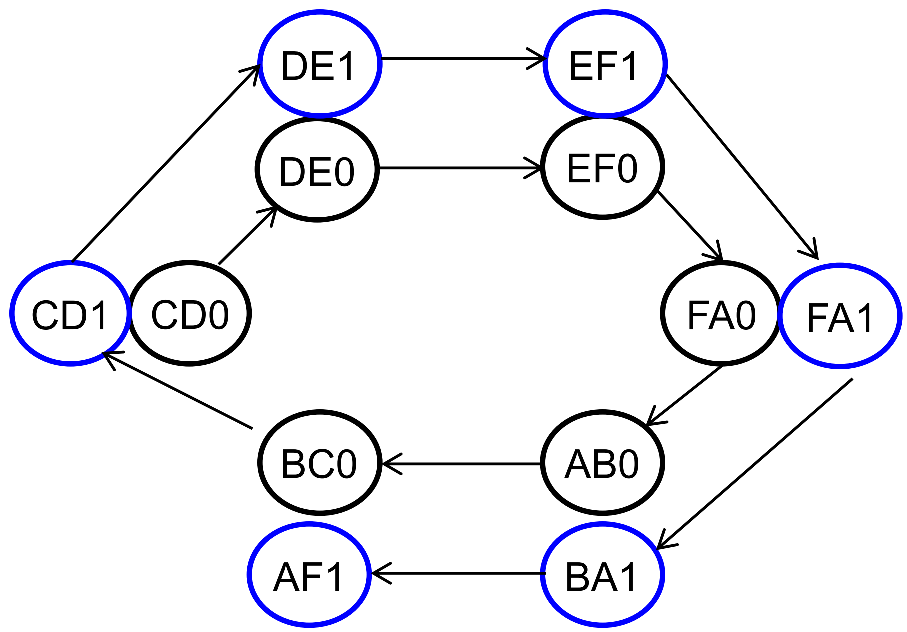


As long at least one buffer per class, can guarantee deadlock freedom

Deadlock Avoidance

- So far, we said deadlock is avoided if cycles eliminated in Channel Dependence Graph
  - Remove cycles via turn restriction
  - Convert cyclic CDG into a spiral using VCs
    - Called *extended* CDG





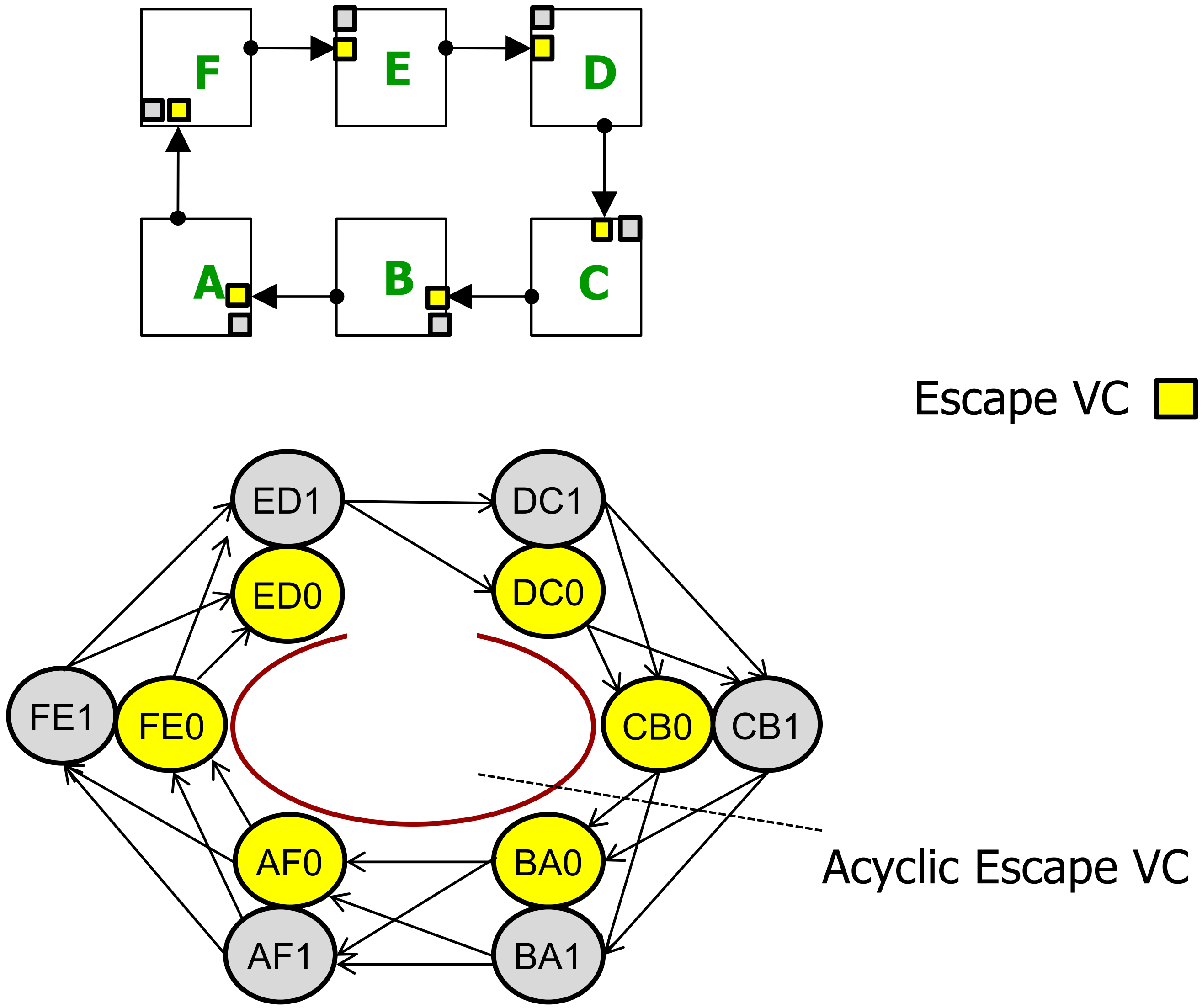
Deadlock Avoidance

- So far, we said deadlock is avoided if cycles eliminated in Channel Dependence Graph
  - Remove cycles via turn restriction
  - Convert cyclic CDG into a spiral using VCs
    - Called *extended* CDG

- However, it is possible for a (extended) CDG to have cycles and still be deadlock-free (Duato*, 1993)
  - As long as the cycle connects to some sub-graph within the (extended) CDG that is acyclic
  - Known as the *escape path* or *escape VC*

*José Duato. A new theory of deadlock-free adaptive routing in wormhole networks. IEEE Transactions on Parallel and Distributed Systems, 4(12):1320–1331, December 1993.

CDG for Escape VCs



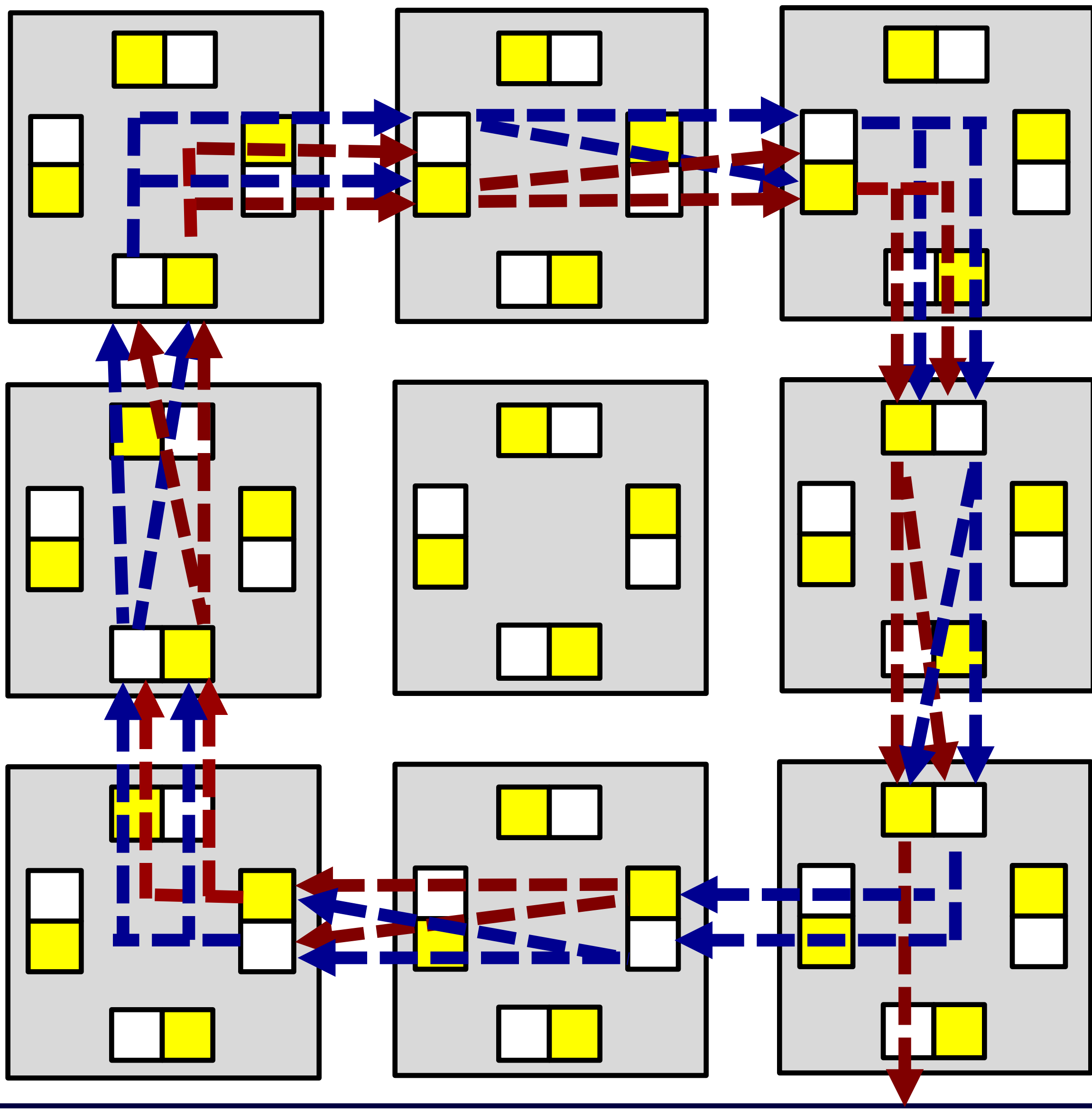
Why escape VCs work

- Intuitively, at least one packet in the cycle has an option to take an acyclic route
  - Packets should not wait on any *specific* channel
  - If allocation is fair, escape VCs guaranteed to show up eventually
- Use of escape channels by a message is not unidirectional
  - If a message enters the escape network it can move back to the adaptive network, and vice versa, if minimal* routes
    - *for non-minimal routes, message has to continue on escape VC once it gets in, without going back to the adaptive VCs

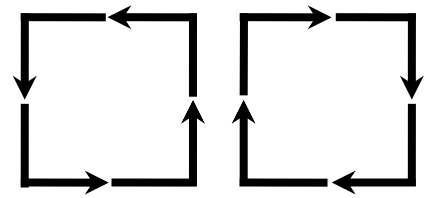
Example

- Consider a 2D Mesh with 8 VCs and minimal routing
  - VC 1-7 can use any arbitrary minimal routing
    - Cyclic CDG
  - VC 0 (escape VC) is restricted to DOR (provides escape path)
    - Acyclic CDG
  - As long as a packet can allocate all VCs fairly, there will always be an escape path available in case the network deadlocks

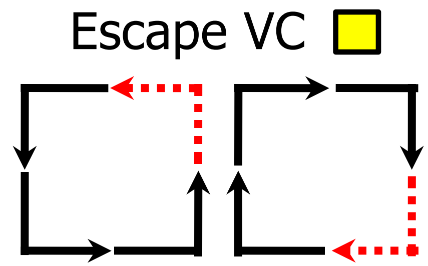
Example



Regular VCs



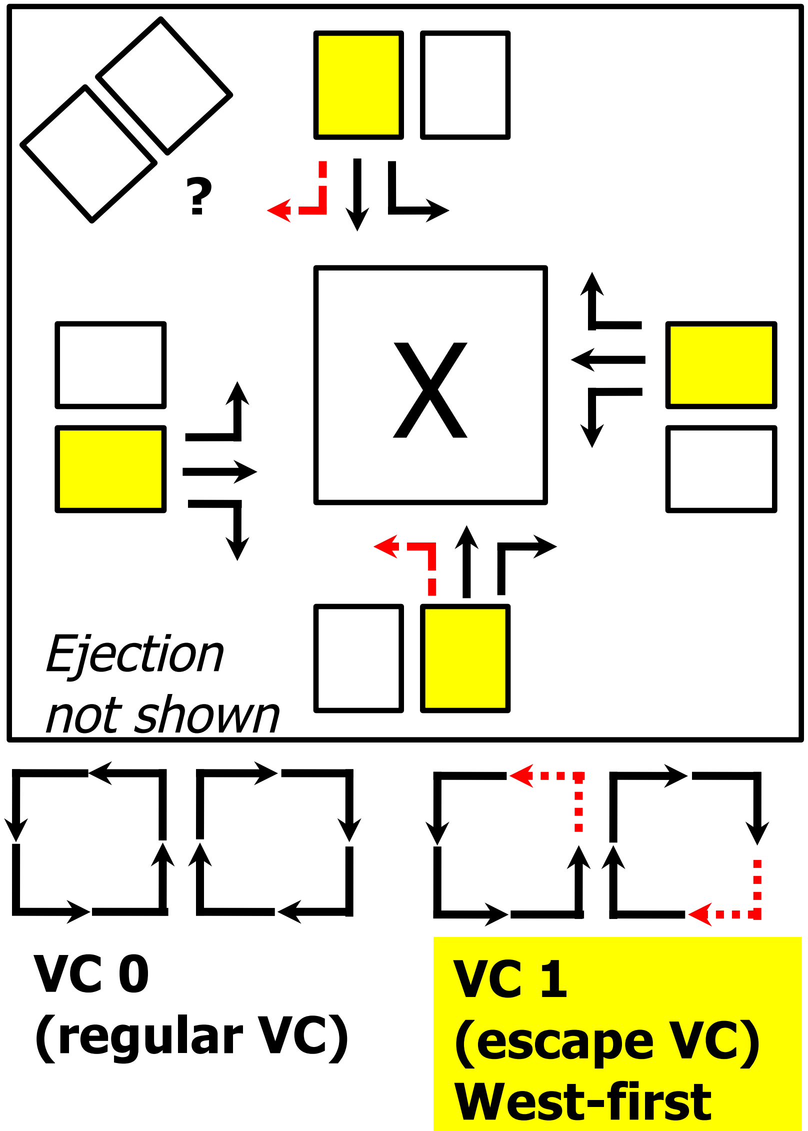
Escape VC



West-first

Deadlock-free
escape path

Rules for getting in/out of escape VCs



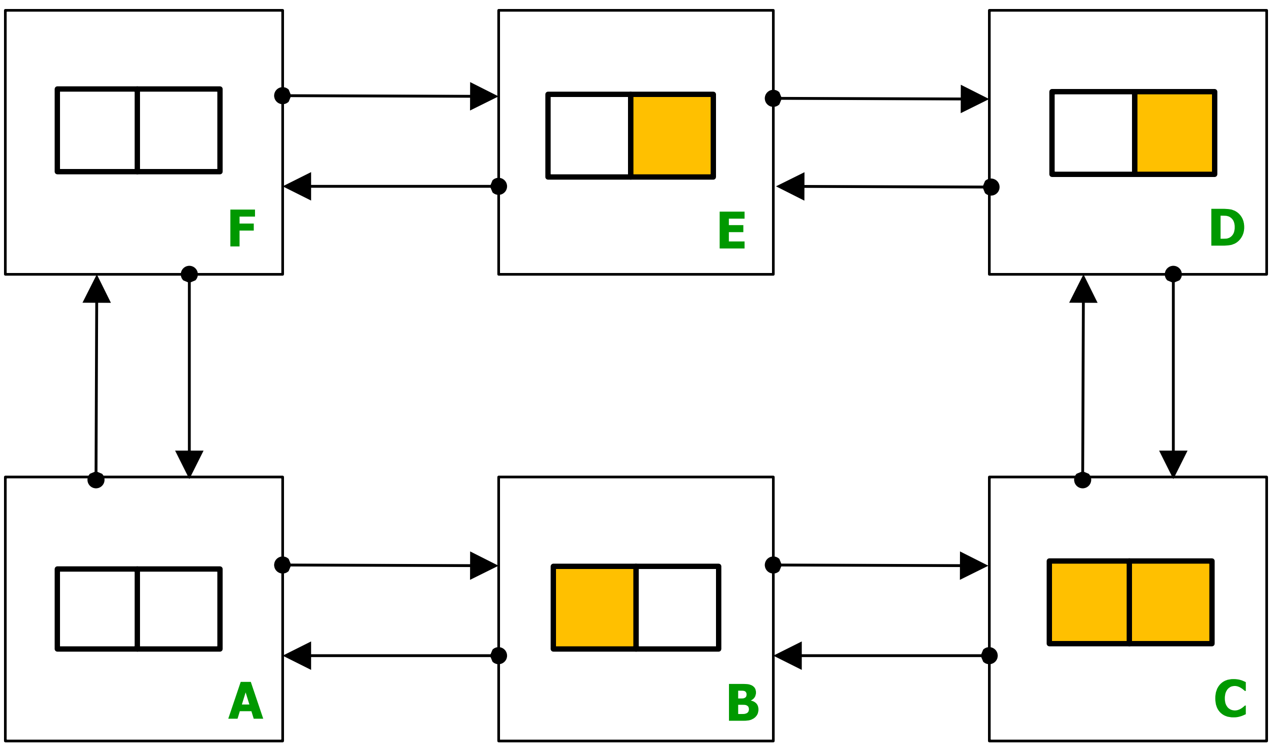
- The escape VC should always makes forward progress!
  - A flit that is going NW or SW should never enter a router from the S or N port in escape VC, else S→W or N→W turn is inevitable
    - How to guarantee this?
      - When selecting VC at previous router
      - Lab 3!

Deadlock Avoidance

- Routing
  - Turn Model in a Mesh
    - Acyclic CDG
- Buffer Assignment
  - Acquire VCs in ascending order
    - Acyclic extended CDG
  - Escape VCs
    - Cyclic CDG + Acyclic sub-graph
- Flow Control
  - Control traffic flow such that deadlock does not occur
    - Cyclic CDG + injection control to avoid dependence cycle

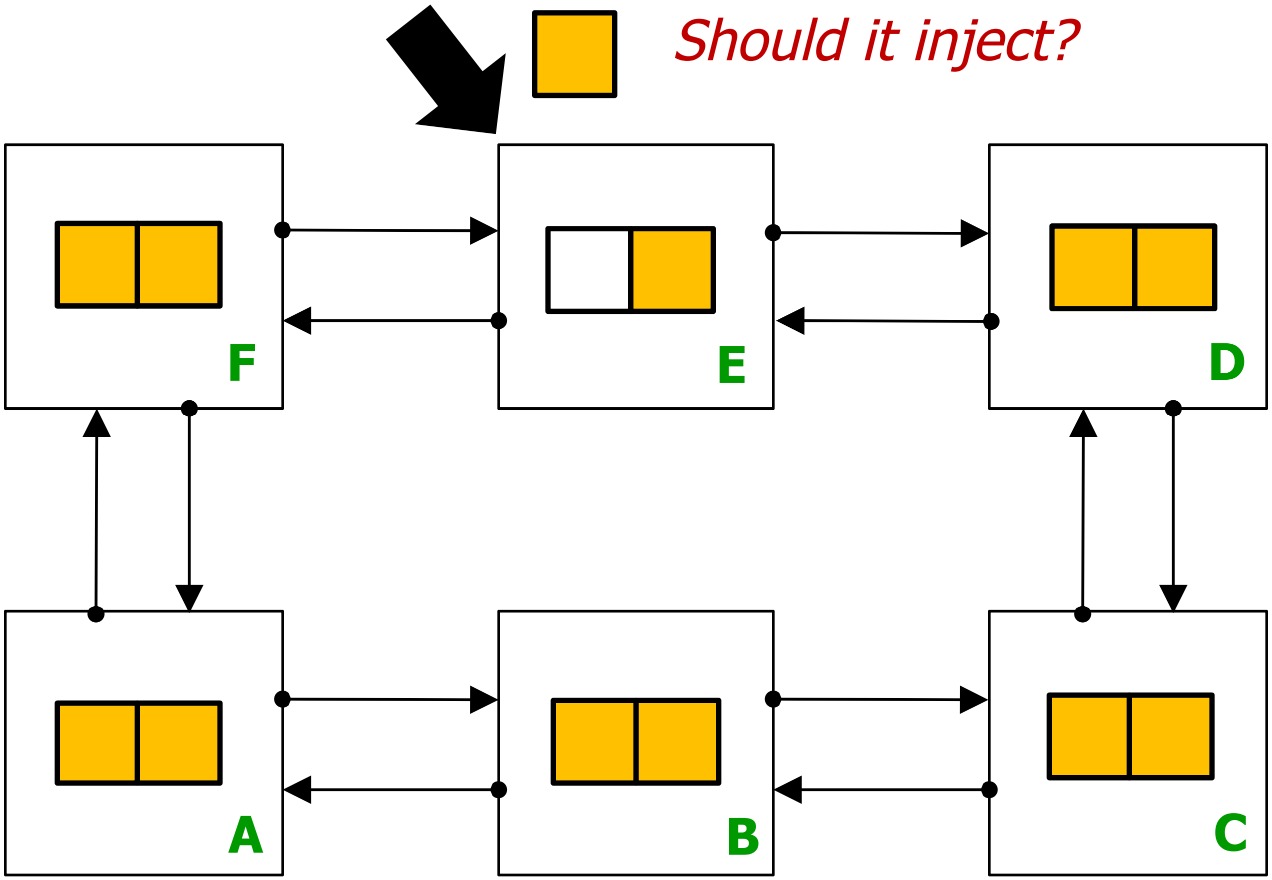
Bubble Flow Control

Ring Traversal Rule:
traverse if one bubble free



*V. Puente et al. **The adaptive bubble router.** Journal of Parallel and Distributed Computing, 2001.*

Bubble Flow Control

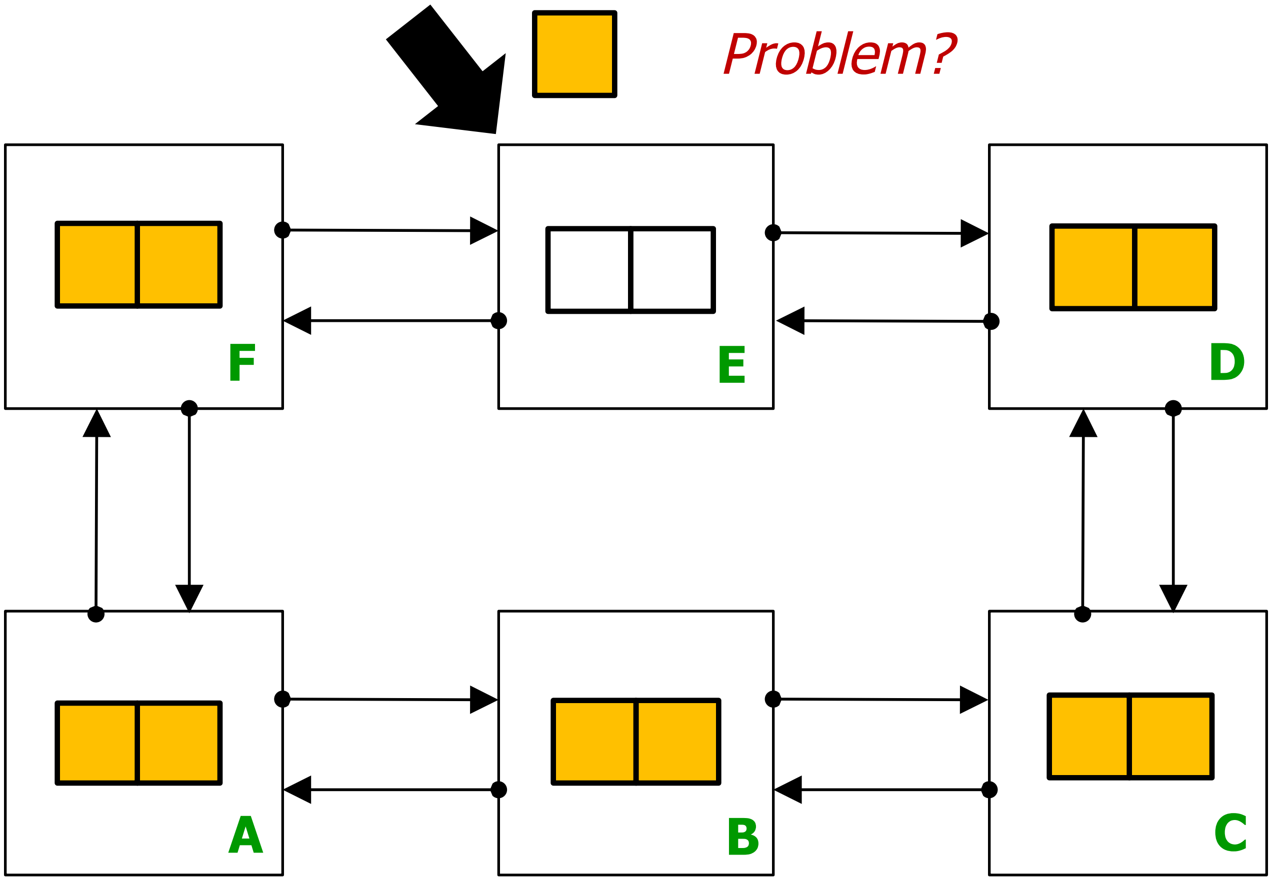


Ring Traversal Rule:
traverse if one bubble free

BFC Injection Rule:
only inject if 2 bubbles free.

V. Puente et al. ***The adaptive bubble router.*** *Journal of Parallel and Distributed Computing, 2001.*

Bubble Flow Control

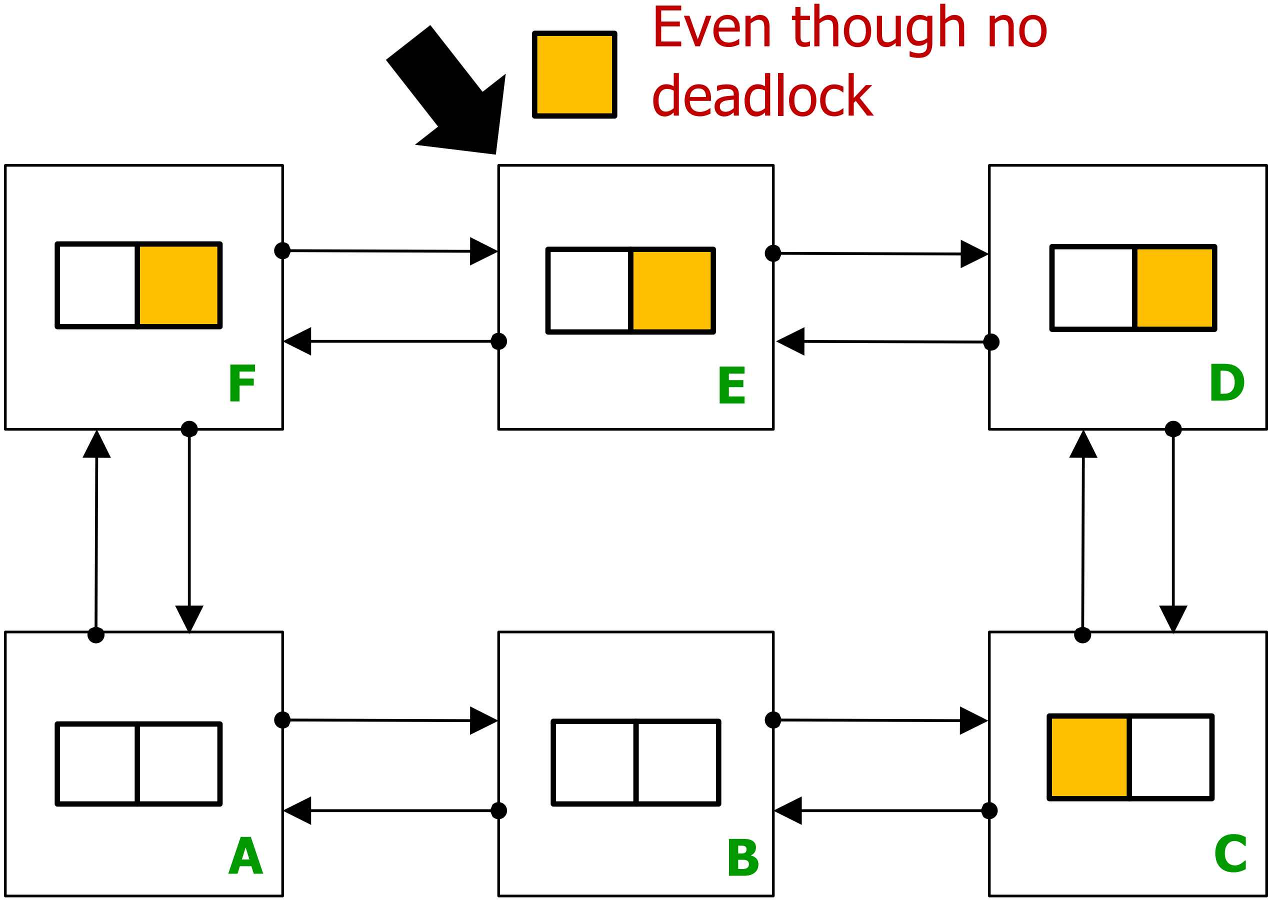


Ring Traversal Rule: traverse if one bubble free

BFC Injection Rule: only inject if 2 bubbles free.

V. Puente et al. ***The adaptive bubble router.*** *Journal of Parallel and Distributed Computing, 2001.*

Bubble Flow Control

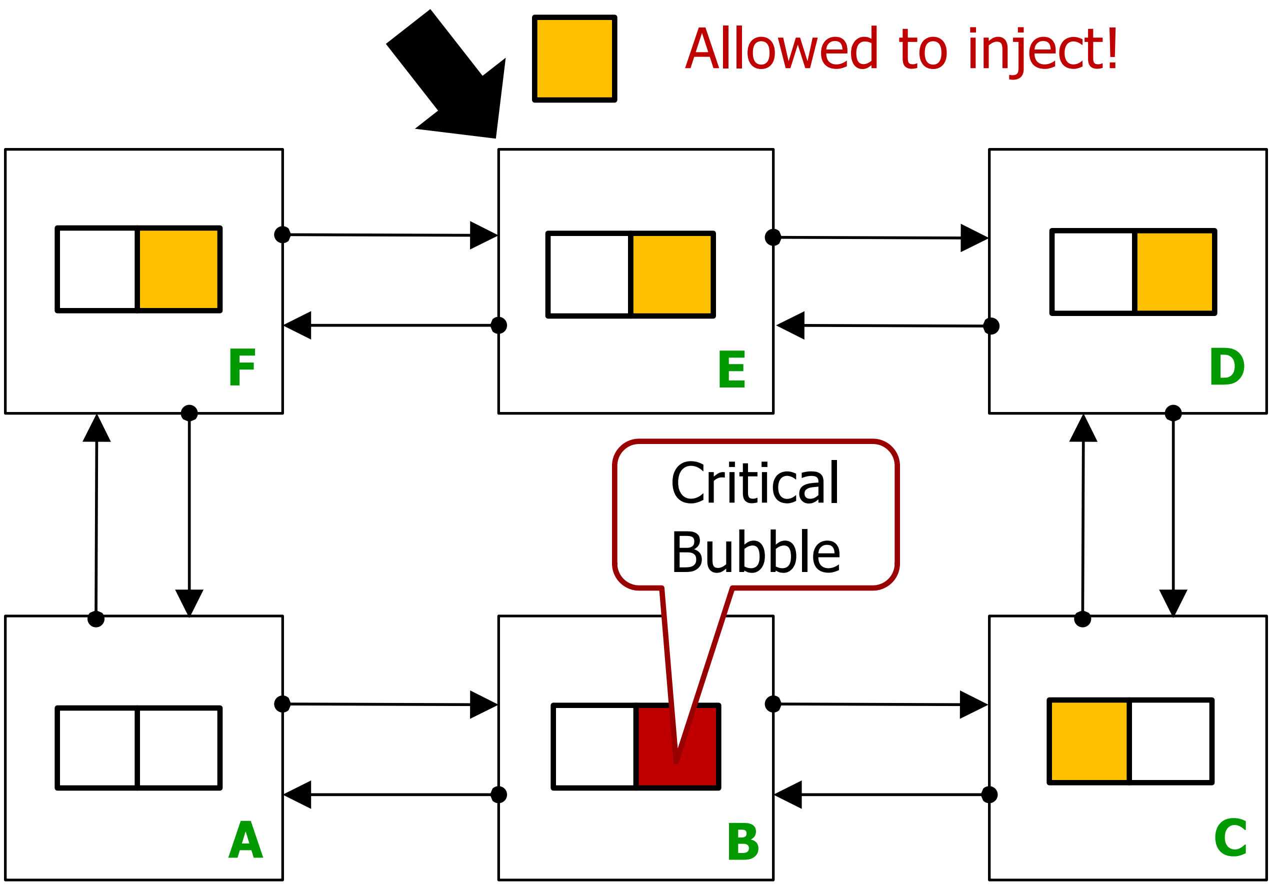


Ring Traversal Rule: traverse if one bubble free

BFC Injection Rule: only inject if 2 bubbles free.

V. Puente et al. ***The adaptive bubble router.*** *Journal of Parallel and Distributed Computing, 2001.*

Critical Bubble Flow Control



Ring Traversal Rule:
traverse if one bubble free

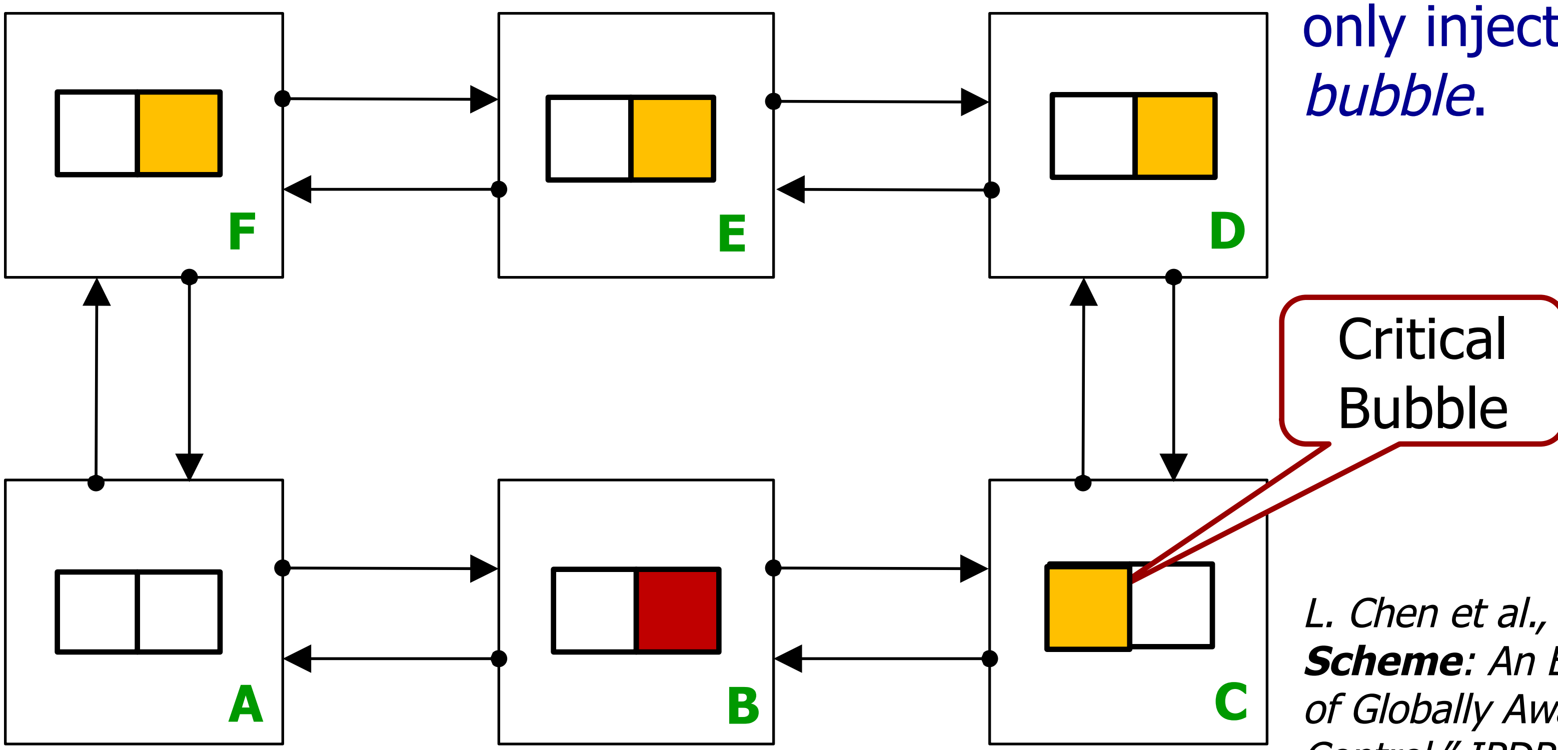
CBFC Injection Rule:
only inject if not *critical bubble*.

*L. Chen et al., "**Critical Bubble Scheme**: An Efficient Implementation of Globally Aware Network Flow Control," IPDPS 2011*

Critical Bubble Flow Control

How does critical bubble move?

If flit moves into critical bubble, its own buffer becomes new critical bubble



Ring Traversal Rule: traverse if one bubble free

CBFC Injection Rule: only inject if not *critical bubble*.

*L. Chen et al., "**Critical Bubble Scheme**: An Efficient Implementation of Globally Aware Network Flow Control," IPDPS 2011*

Dealing with Deadlocks

- **Avoidance**
  - Guarantee that the network will never deadlock
  - Almost all modern networks use deadlock avoidance

- **Recovery**
  - Detect deadlock and correct

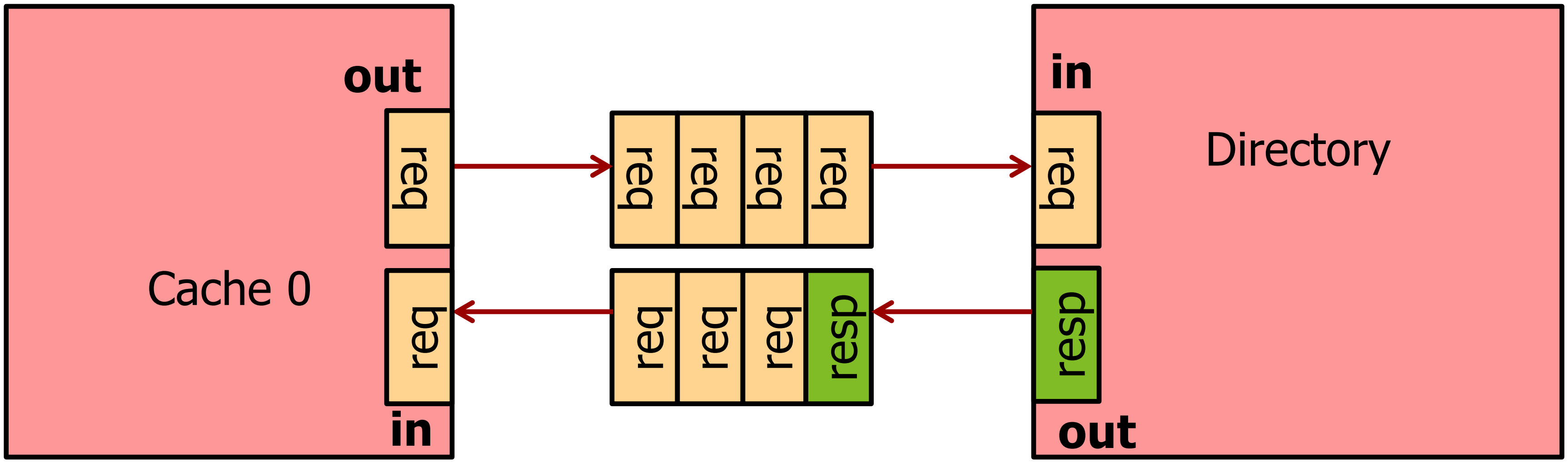
Deadlock Recovery

- When might this make sense?
  - Cannot accept performance degradation required to avoid deadlocks
  - Average-case performance more important
- Two phases
  - **Detection:**
    - E.g., timeouts attached with each resource
      - Can lead to false positives
  - **Recovery:**
    - Regressive – remove packets/connections that are deadlocked
      - E.g., drop packets after timeout
    - Progressive – recover without removing packets/connections
      - E.g. shared escape buffer to drain deadlocked packets

K. V. Anjan and T. M. Pinkiston, "An efficient fully adaptive deadlock recovery scheme: DISHA", ISCA 1995

Another kind of deadlock: Protocol Deadlock

Cache / Directory can process a request only if there is space in its output queue to send a response



Deadlock, even though network is deadlock-free

Need separate Virtual Channels* for requests and responses (called Virtual Networks)

Responses should always be drained ("consumption assumption")