UNIVERSITY OF CAMBRIDGE

Exo: Atomic Broadcast for the Rack-Scale Computer

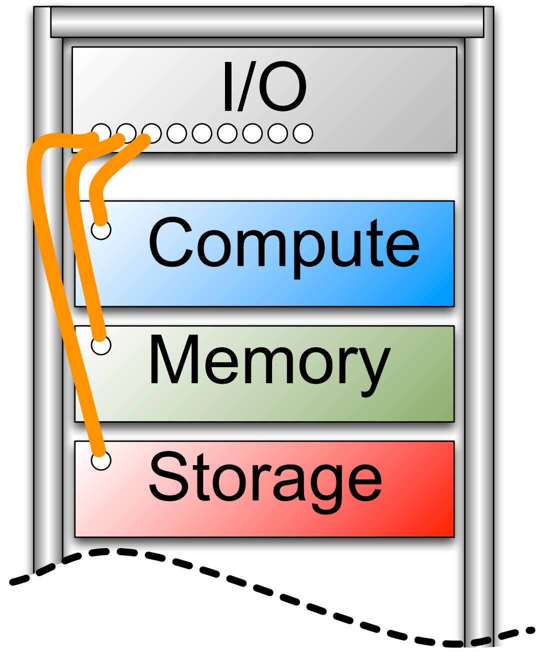
Matthew P. Grosvenor Marwan Fayed Andrew W. Moore

What is a Rack-Scale Computer

Today

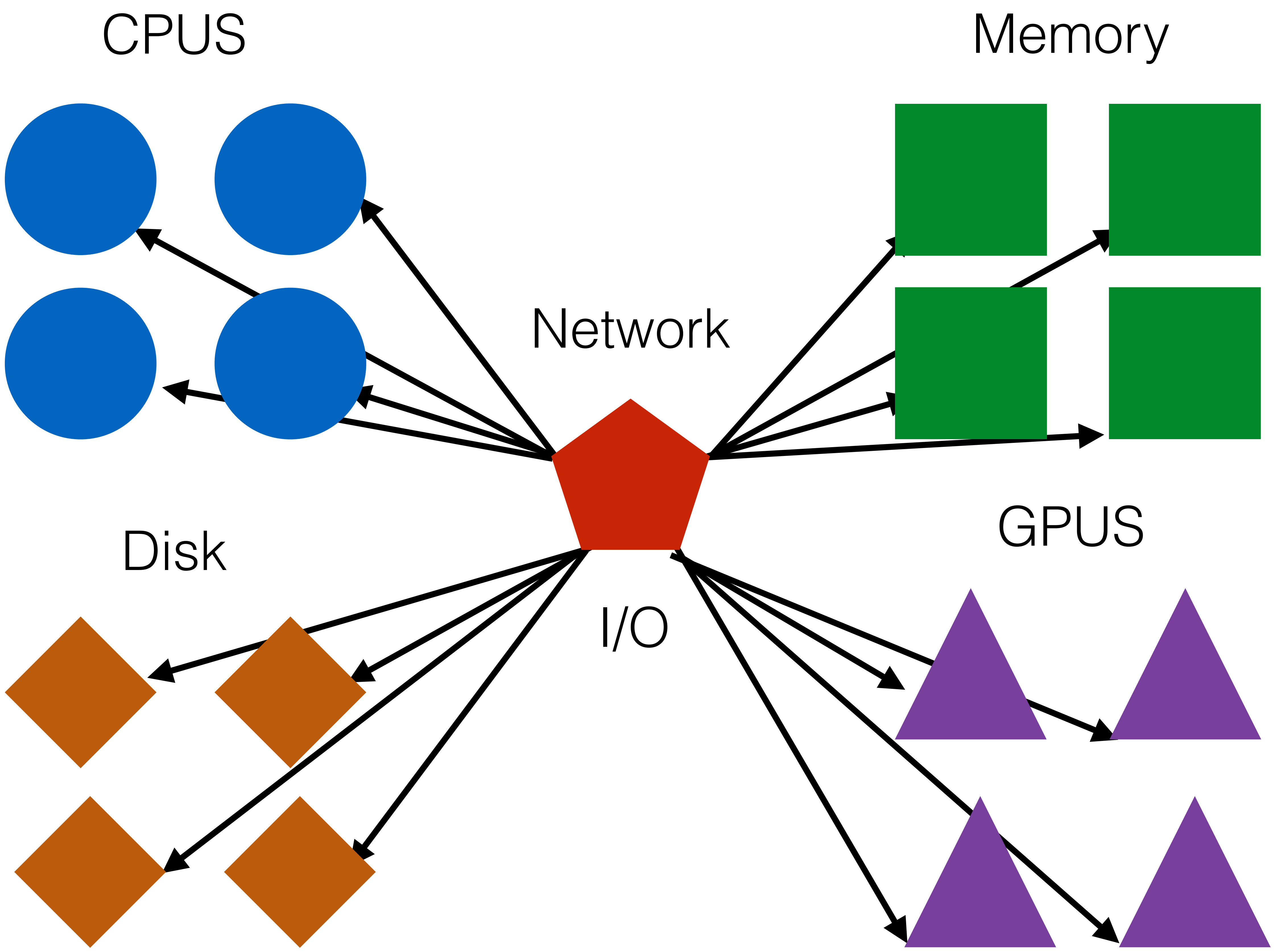
- 50 - 200 machines
- Commodity hardware
- Commodity network

Tomorrow



- 500-2000 “nodes”
- Disaggregated hardware
- Custom (photonic?) interconnect(s)

The problem with disaggregation



Other Types of Coordination

Single Machines

- Coordination using simple MESI/MOSI protocols
- Specialised hardware over a reliable, low latency interconnect.

✓ High performance

X Not failure tolerant

X Doesn't scale well

Other Types of Coordination

Single Machines

- Coordination using simple MESI/MOSI protocols
- Specialised hardware over a reliable, low latency interconnect.

✓ High performance

X Not failure tolerant

X Doesn't scale well

Cluster Systems

- Coordination using software protocols
- General purpose, high latency, error prone network

X Low performance

✓ Fault Tolerant

X Doesn't scale well

Other Types of Coordination

Single Machines

- Coordination using simple MESI/MOSI protocols
- Specialised hardware over a reliabl… interco…

Cluster Systems

- Coordination using software protocols
- General purpose, high latency, error … ork

Something in the middle ???

✓ High performance

X Not failure tolerant

X Low performance

✓ Fault Tolerant

X Doesn't scale well

X Doesn't scale well

Other Types of Coordination

Single Machines

- Coordination using simple MESI/MOSI protocols
- Specialised hardware over a reliabl… interc…

Cluster Systems

- Coordination using software protocols
- General purpose, high latency, error … ork

Something in the middle ???

✓ High performance

X Not failure tolerant

X Low performance

✓ Fault Tolerant

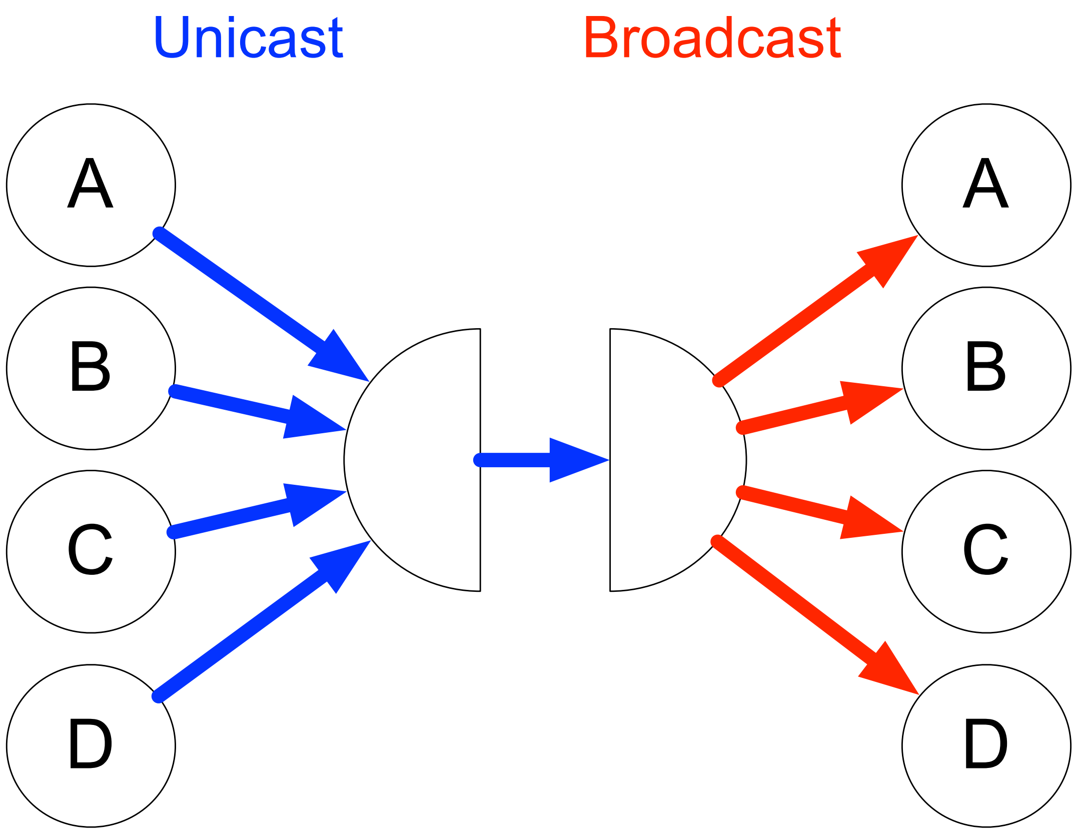
X Doesn't scale well

X Doesn't scale well

That scales?

Building a Coordination Network (Tomorrow)

- ✓ Silicon Photonic Interfaces on chip
- X High radix optical switches?
- ✓ Burst mode transceivers
- ✓ Passive all optical interconnect (PON)
- ✓ Network broadcast as a primitive for building Atomic Broadcast



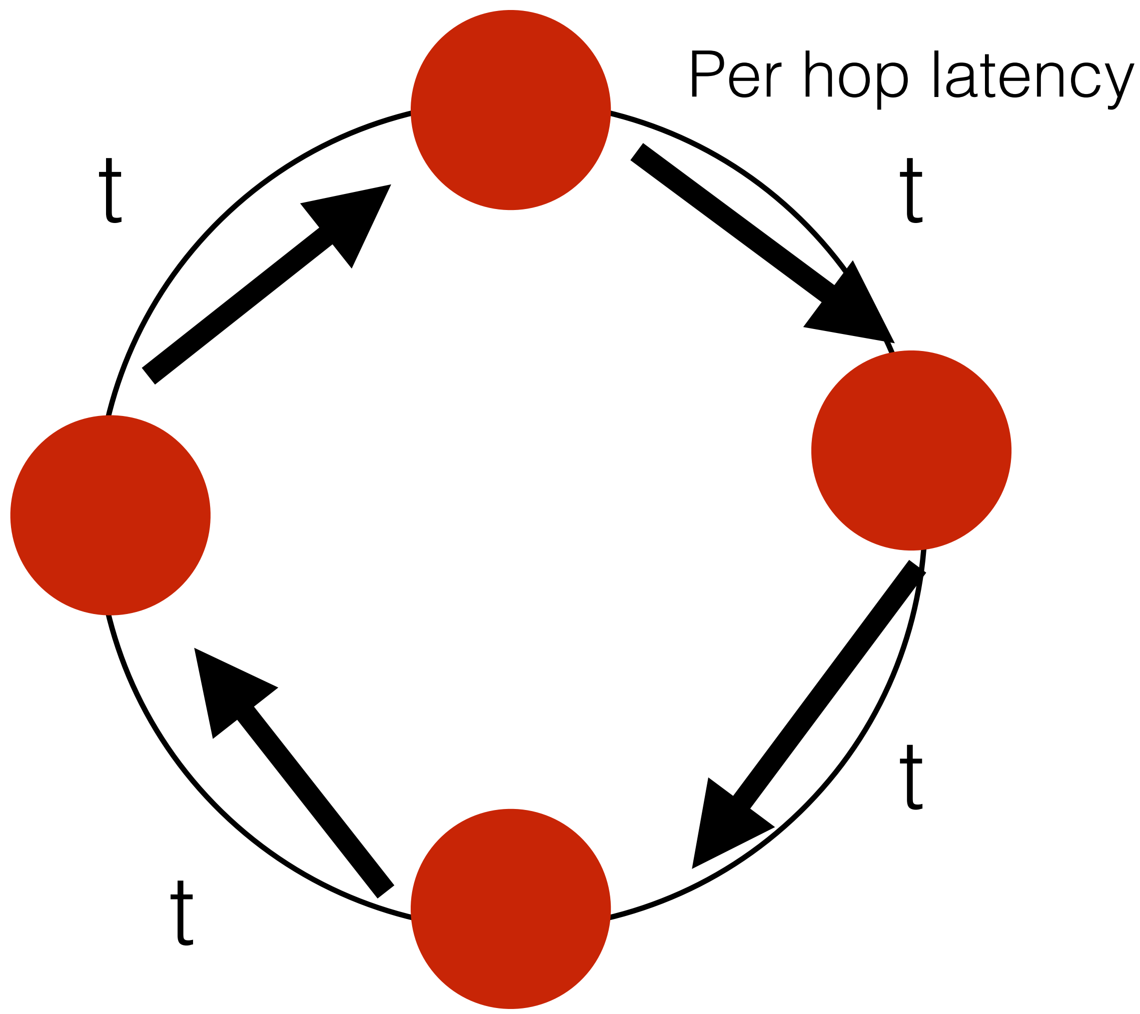
Media Access and Agreement

- How do we mediate access to the network?
- What agreement protocol do we use?

- Use token rings
    - Two birds with one stone
        - ✓ Well established MAC protocol
        - ✓ Well established agreement protocol. Very high (optimal) throughput performance.

The Problem with Token Passing

- Token passing is latency bound
- Each host must wait at least n-1 hops for agreement = n x per-hop latency (t)
- Per hop latency must be minimised

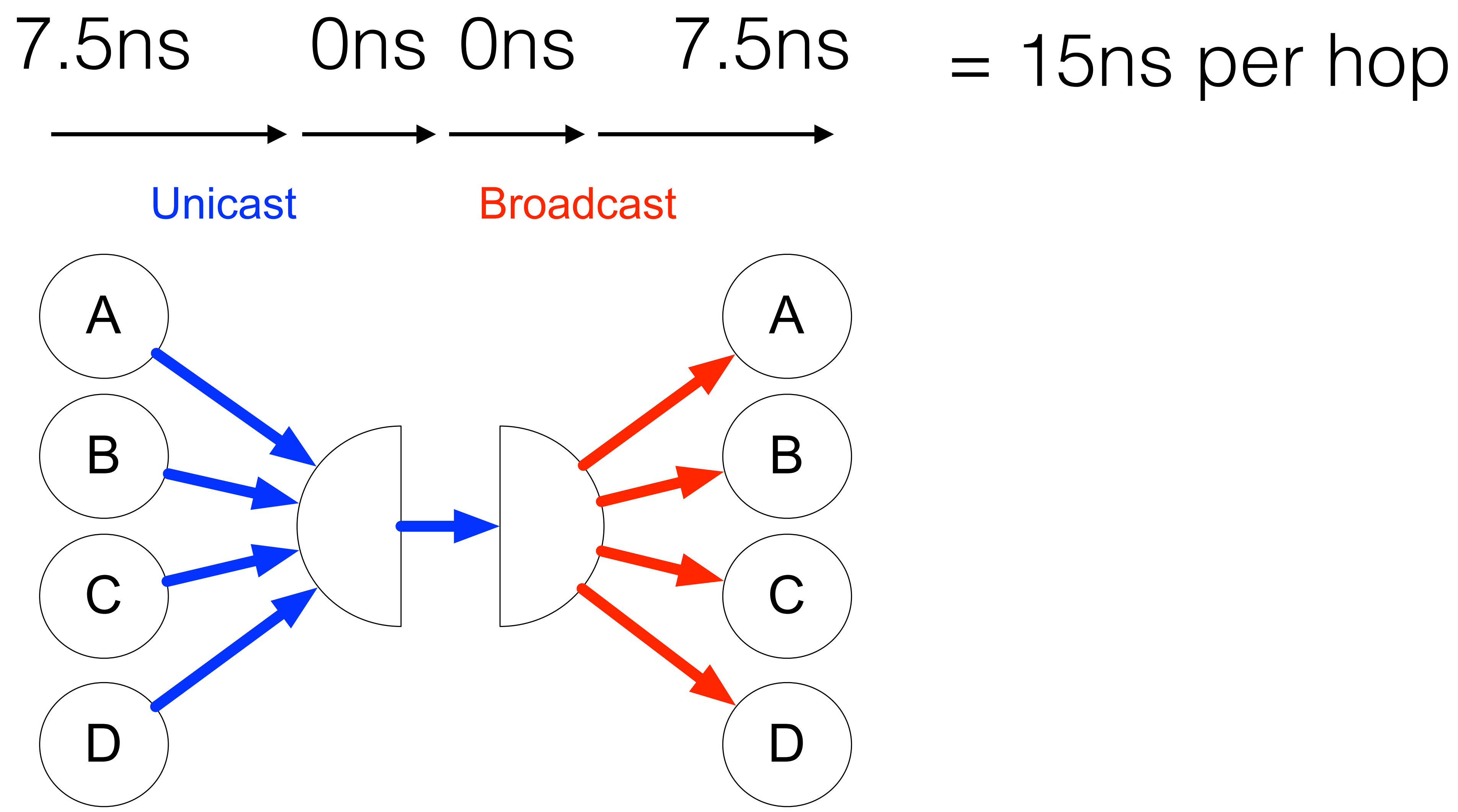


Total latency = t + t + t + t = 4t

Throughput (per host) ≈ 1/ 4t

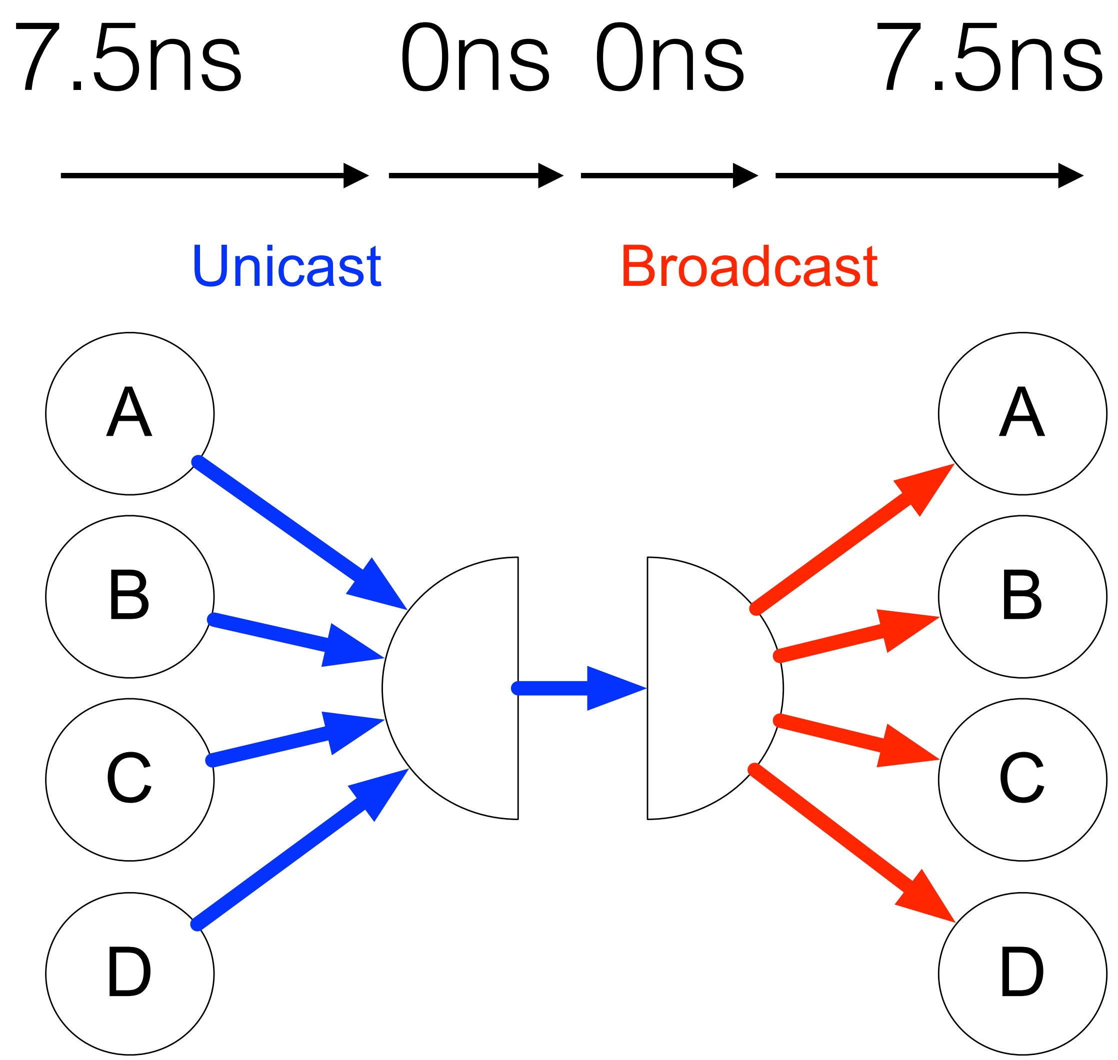
What's the latency?

Assuming 3m of fibre @ 0.75C



What's the latency?

Assuming 3m of fibre @ 0.75C



= 15ns per hop

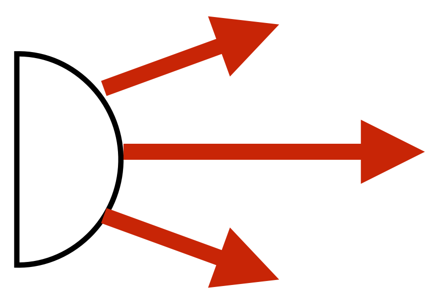
@10G 64B frames

= 14.4M msg/sec

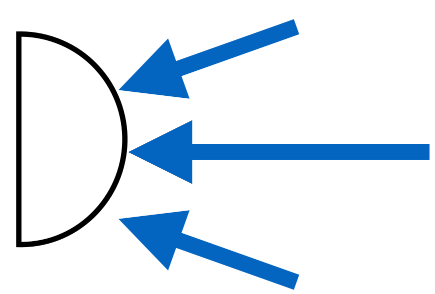
Building a Coordination Network (Today)

Use Exalink Fusion Switch to emulate all optical network

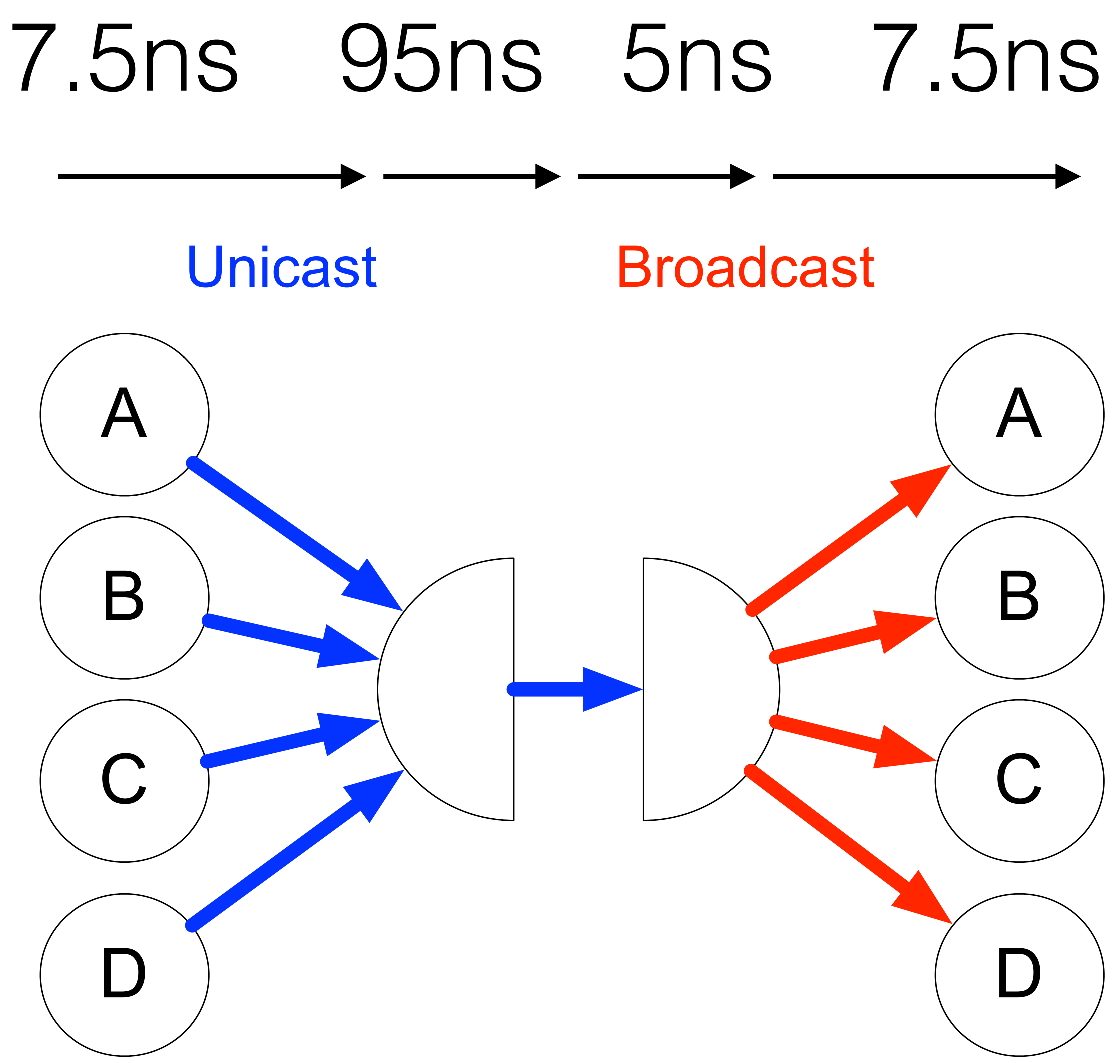
- Broadcast - 5ns



- Aggregation - 95ns



Building a Coordination Network (Today)



= 115ns per hop

= 8.6M msg/sec

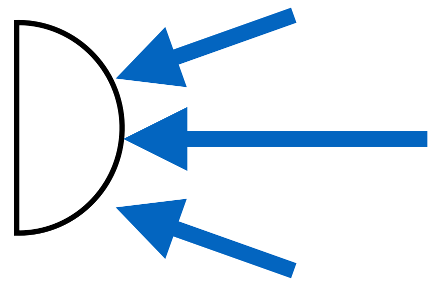
Fusion Aggregate

Fusion Broadcast

Building a Coordination Network (In the lab)

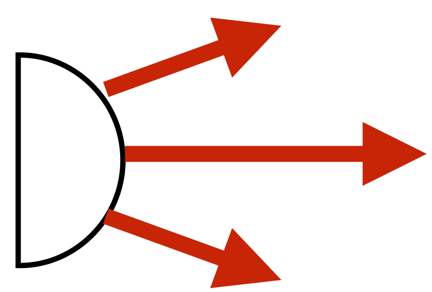
- Use ethernet switch and matrix switch to emulate Exablaze Fusion

- Arista 7124FX



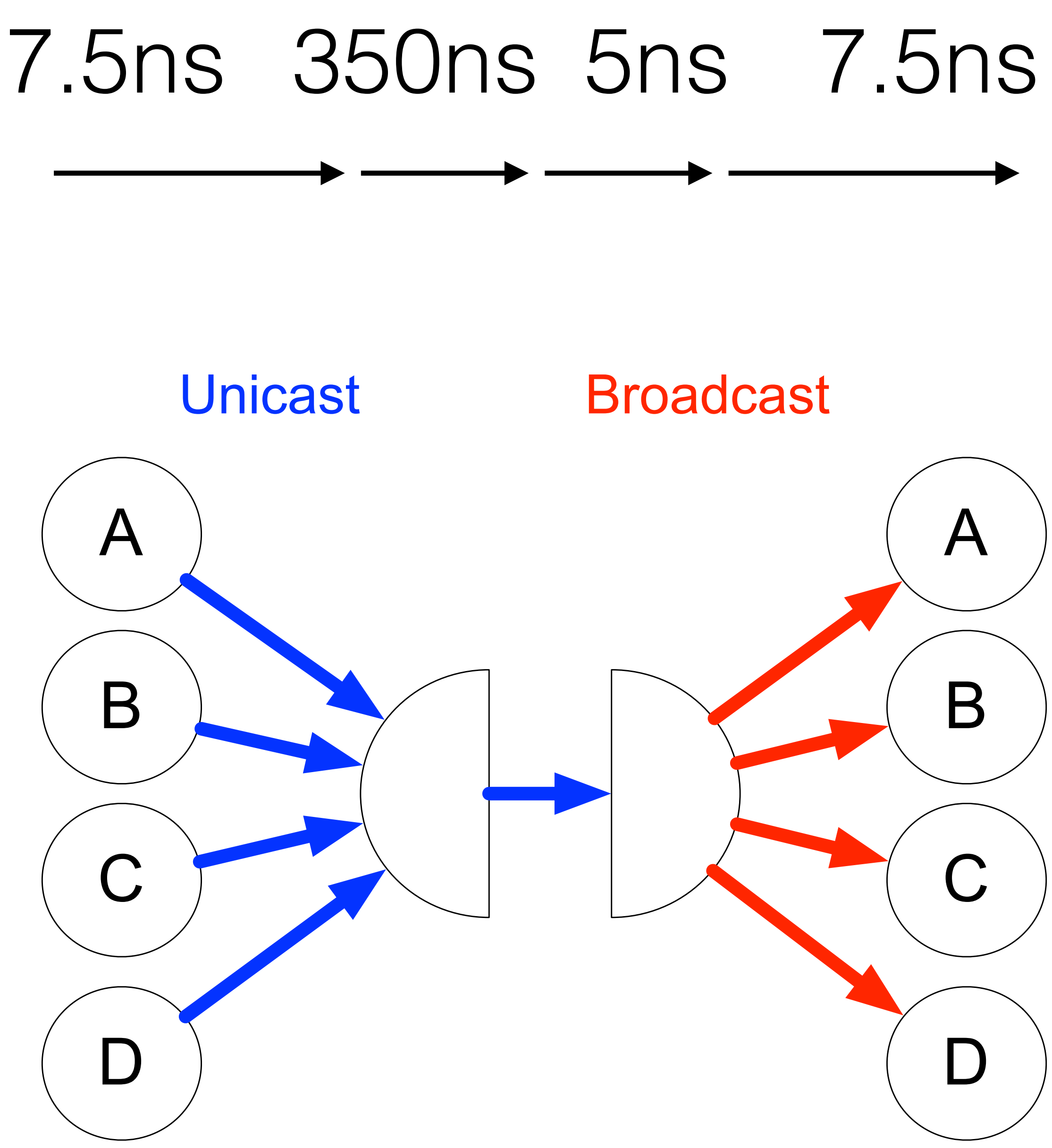
  - Aggregation/switching 350ns

- Exablaze ExaLink 50



  - Broadcast <5ns

Building a Coordination Network (in the lab)



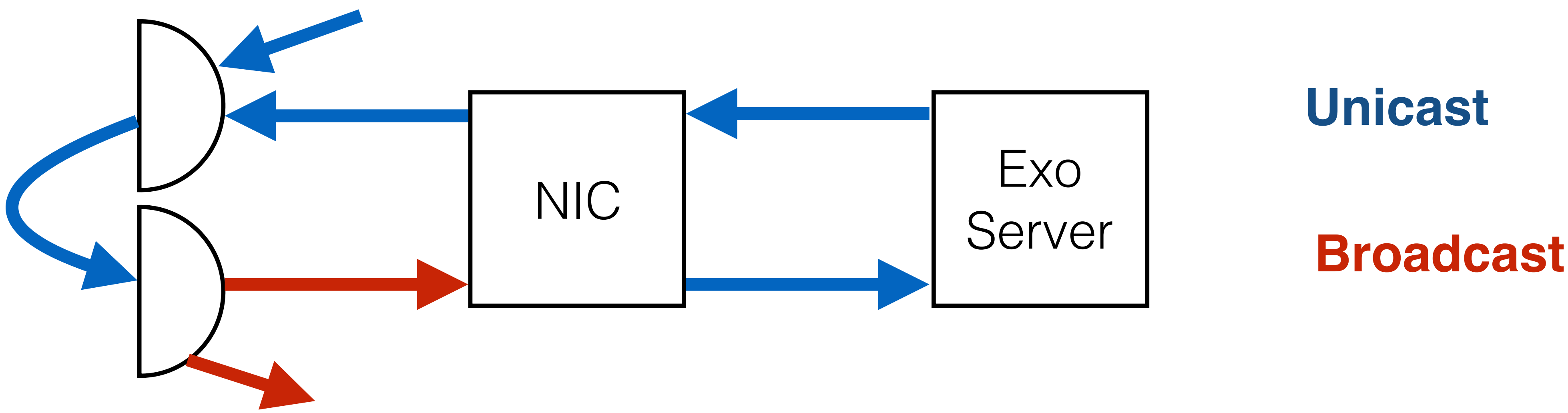
= 370ns per hop

= 2.7M msg/sec

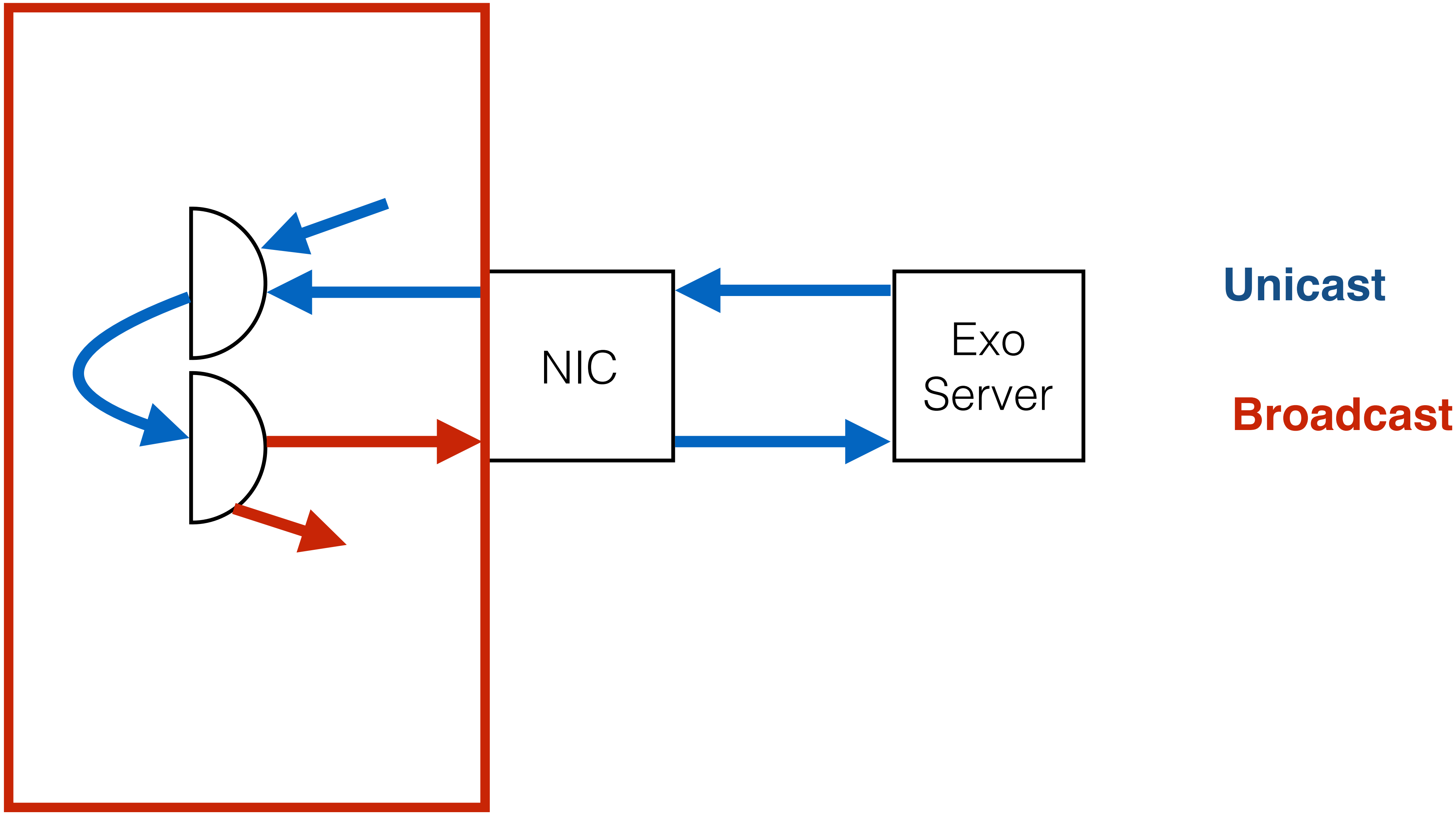
Artist 7124FX

Exablaxe ExaLink50

But wait, there's more… (latency)



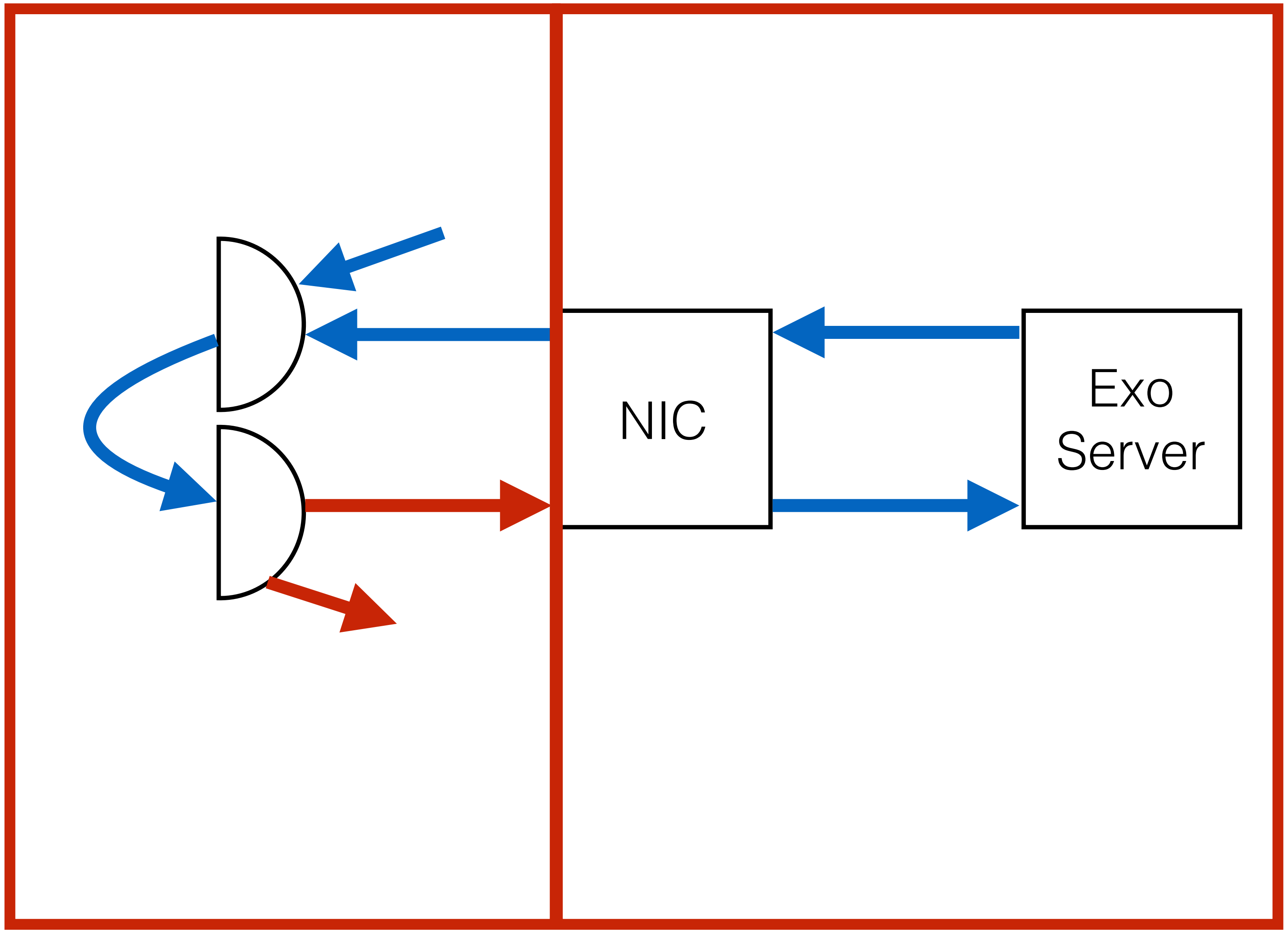
But wait, there's more… (latency)



Network Latency

370ns

But wait, there's more (latency)



Unicast

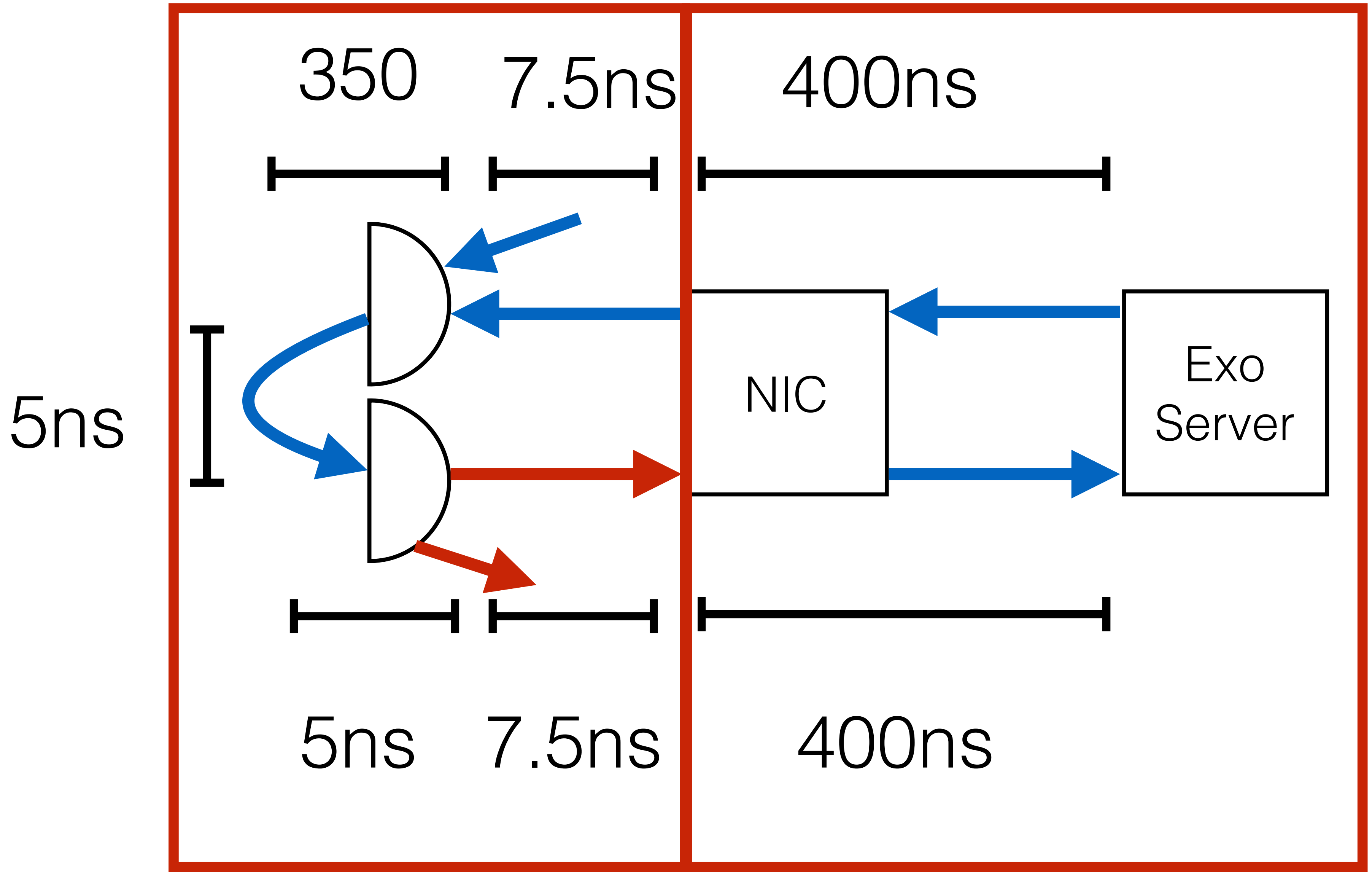
Broadcast

Network Latency

PCIe Latency

370ns

But wait, there's more… (latency)



Unicast

Broadcast

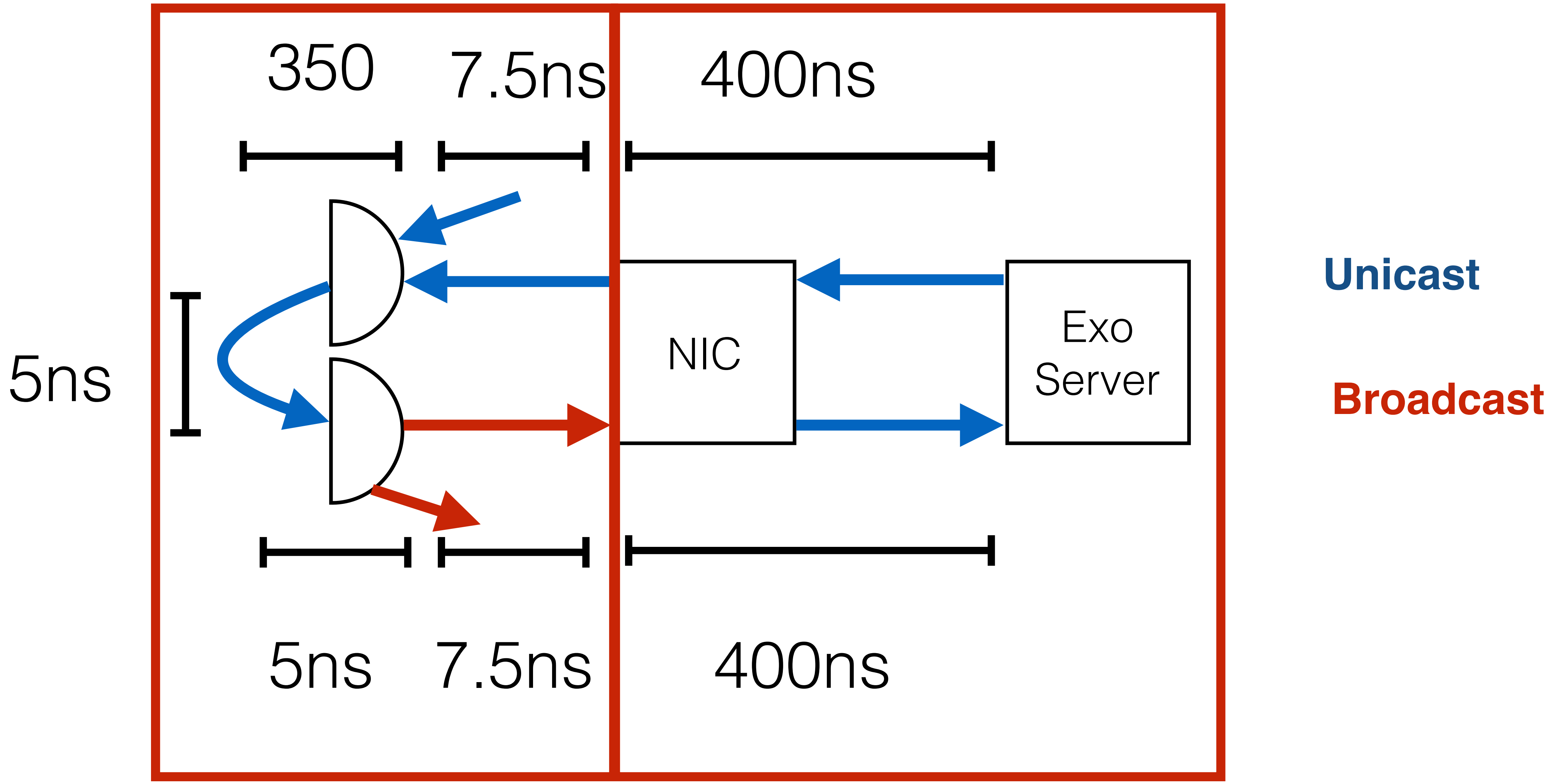
Network Latency

370ns

PCIe Latency

>800ns

But wait, there's more… (latency)



- total per-hop-latency ≈ 1us
- 200 x 1us ≈ 200us
  ≈ 5000 msgs per host per second.

Dealing with latency

1. Make it go away
   - e.g use a Fusion
   - e.g use all optical

2. Hide it
   - Introduce pipelining

Dealing with latency

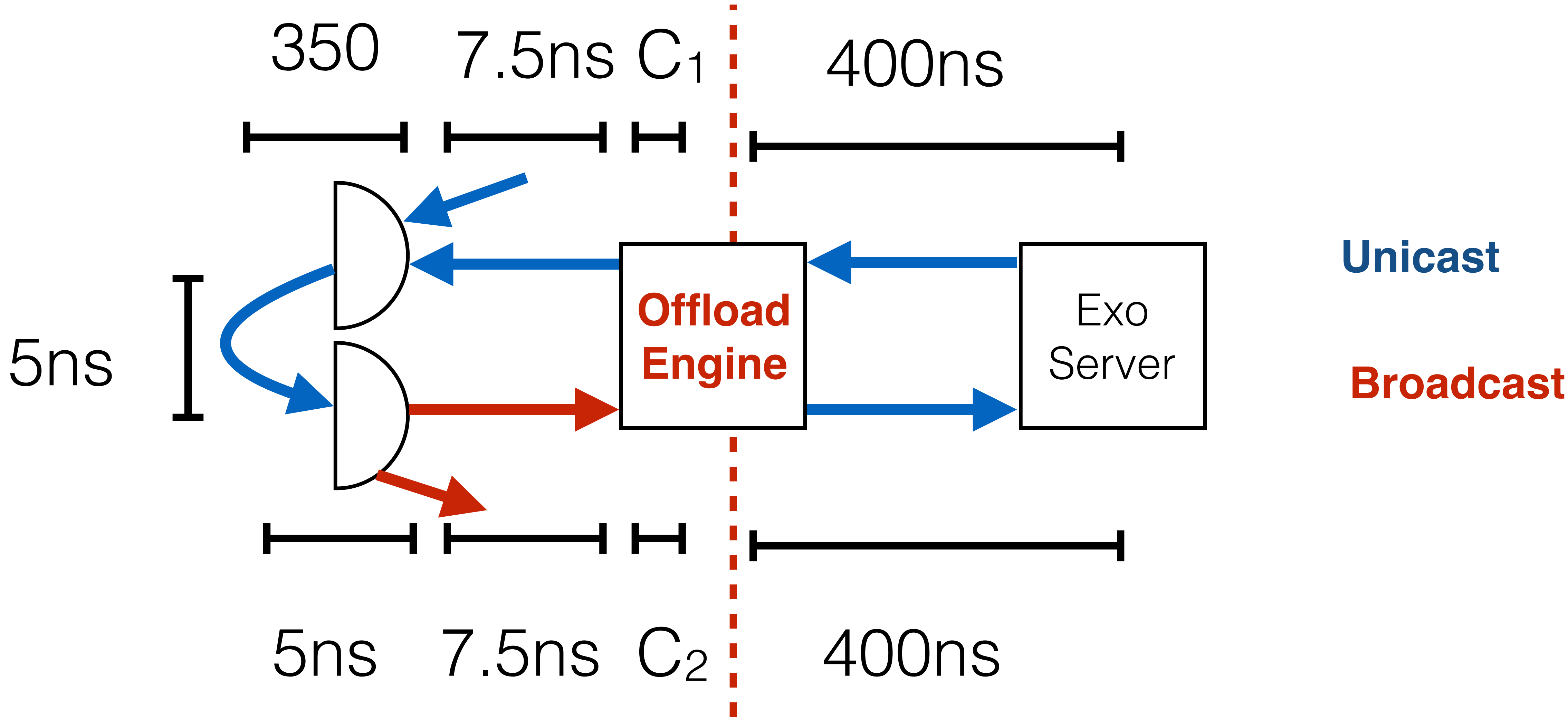
1. Make it go away

   - e.g use a Fusion
   - e.g use all optical

2. **Hide it**

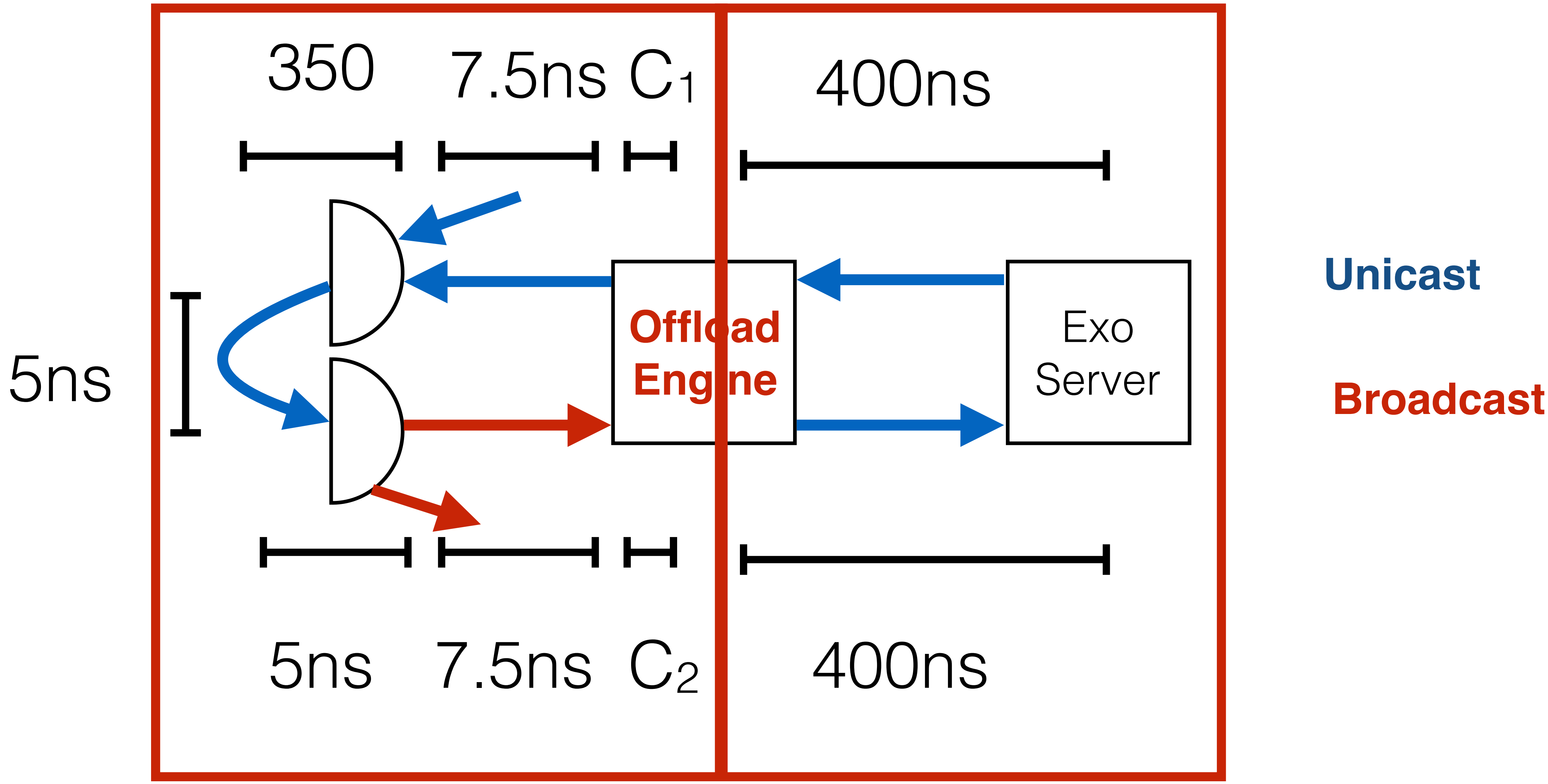
   - **Introduce pipelining**

Hiding Latency



- Introduce pipelining to hide latency
  - Host to NIC and NIC to network transmissions run in parallel

Hiding Latency



- In network latency ≈ 400ns
- Per-hop latency ≈ 1us (pipelined for 1 message per 400ns)
- 200 x 400ns ≈ 80us
  ≈ 12,500 messages per second / per host.

Let’s Talk About Failure

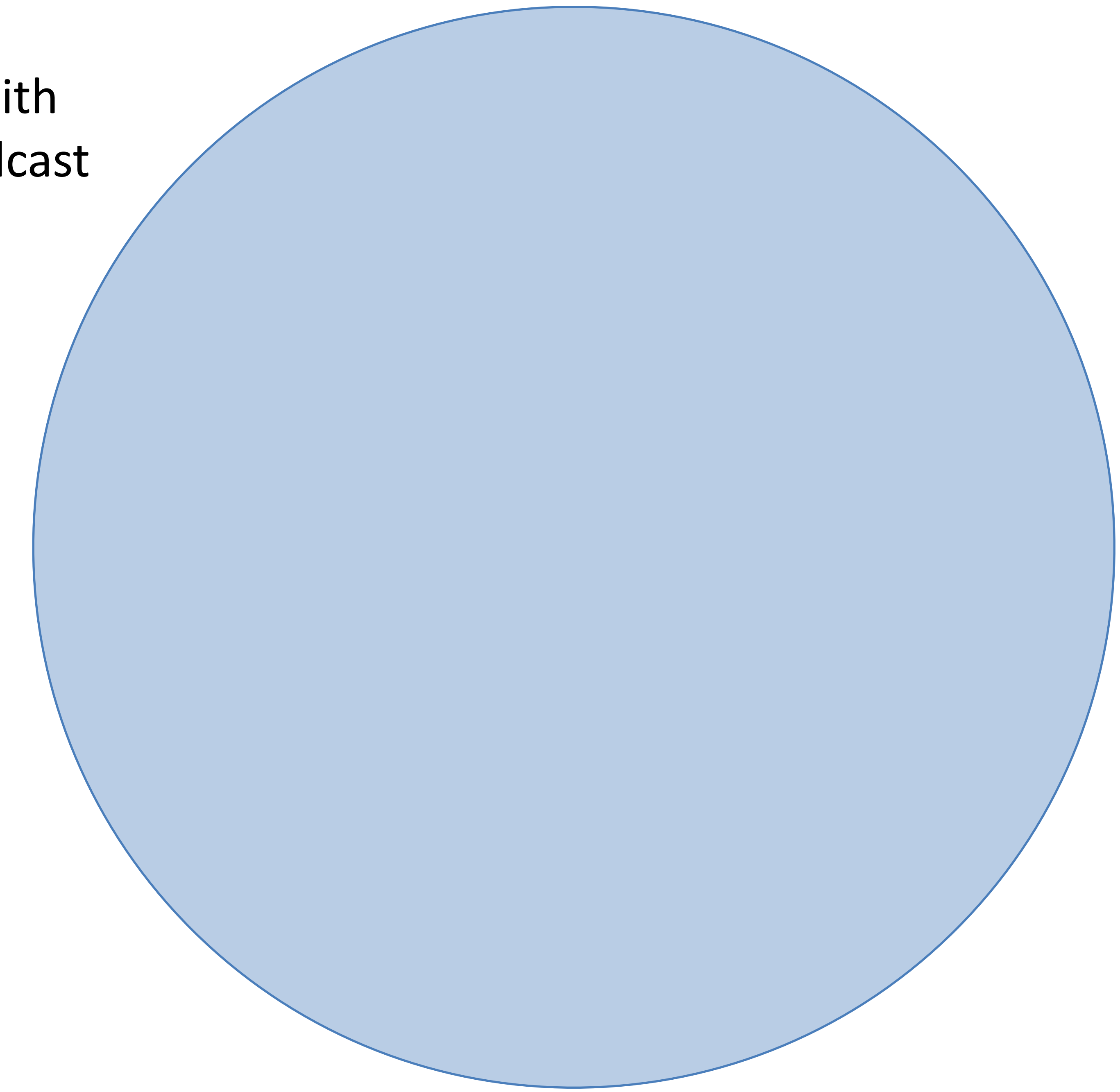
- Failure is very unlikely.
  - No packet loss due to congestion
  - Bit errors on the wire 1/10^12 … 1/10^14
  - Partitioning - Single chip? Can it half fail? Byzantine?
    - 1/10^20 ??
  - Partitioning - requires n-way x m - correlated failure
    - (1/10^12)^n * m
- Therefore, optimise for the common case

The Exo Protocol

- Protocol is implemented correctly on all hosts
- Hosts may stop, crash, restart, loose packets or become partitioned
- The is a fixed upper number of hosts (n) in the network. Host may leave and come back, but more hosts may not be added.

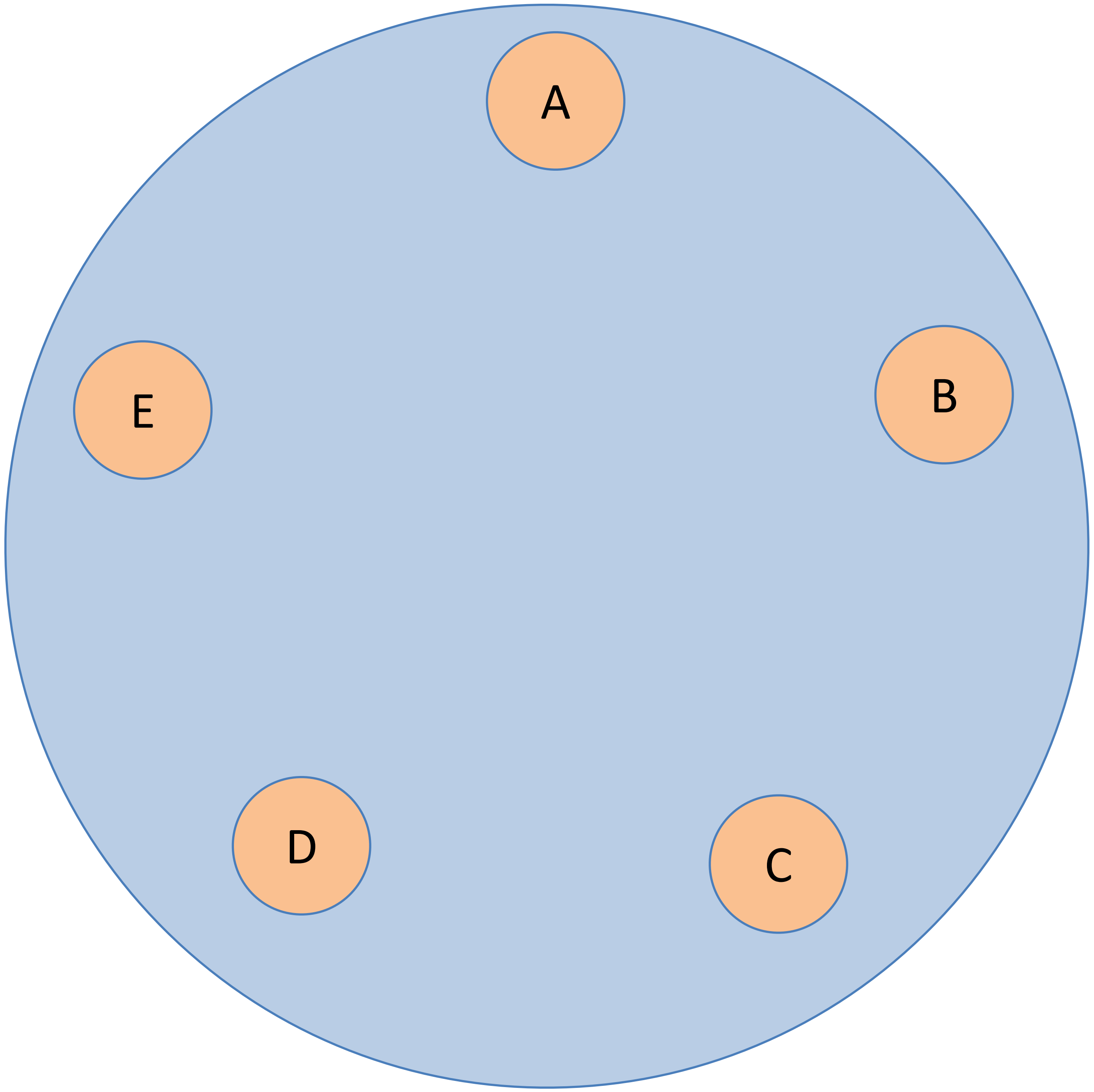
Anatomy of the Exo Protocol

Assume a network with fast and cheap broadcast messaging



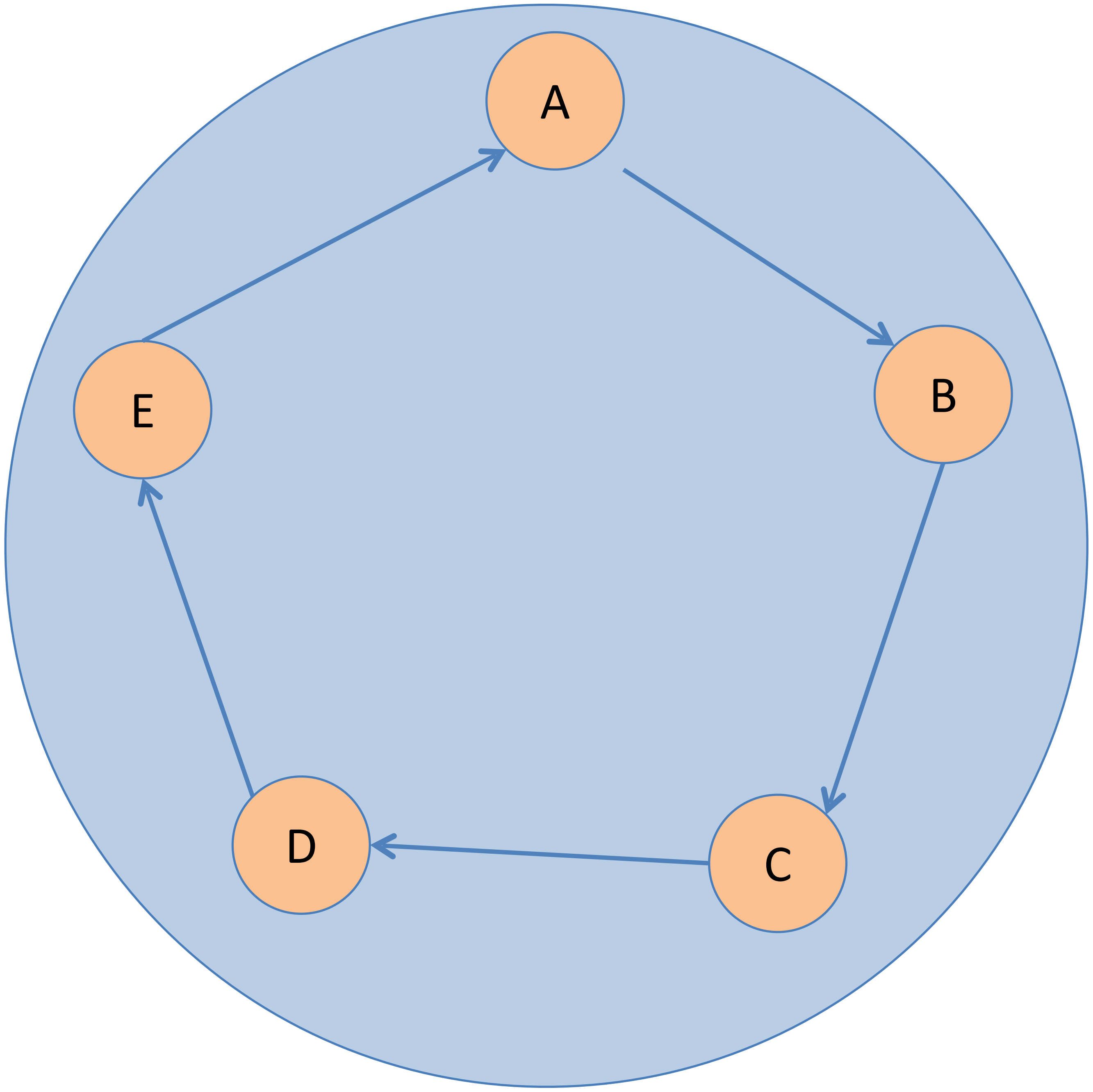
Anatomy of the Exo Protocol

And some maximum number of hosts ***n*** (5) labeled **A-E**



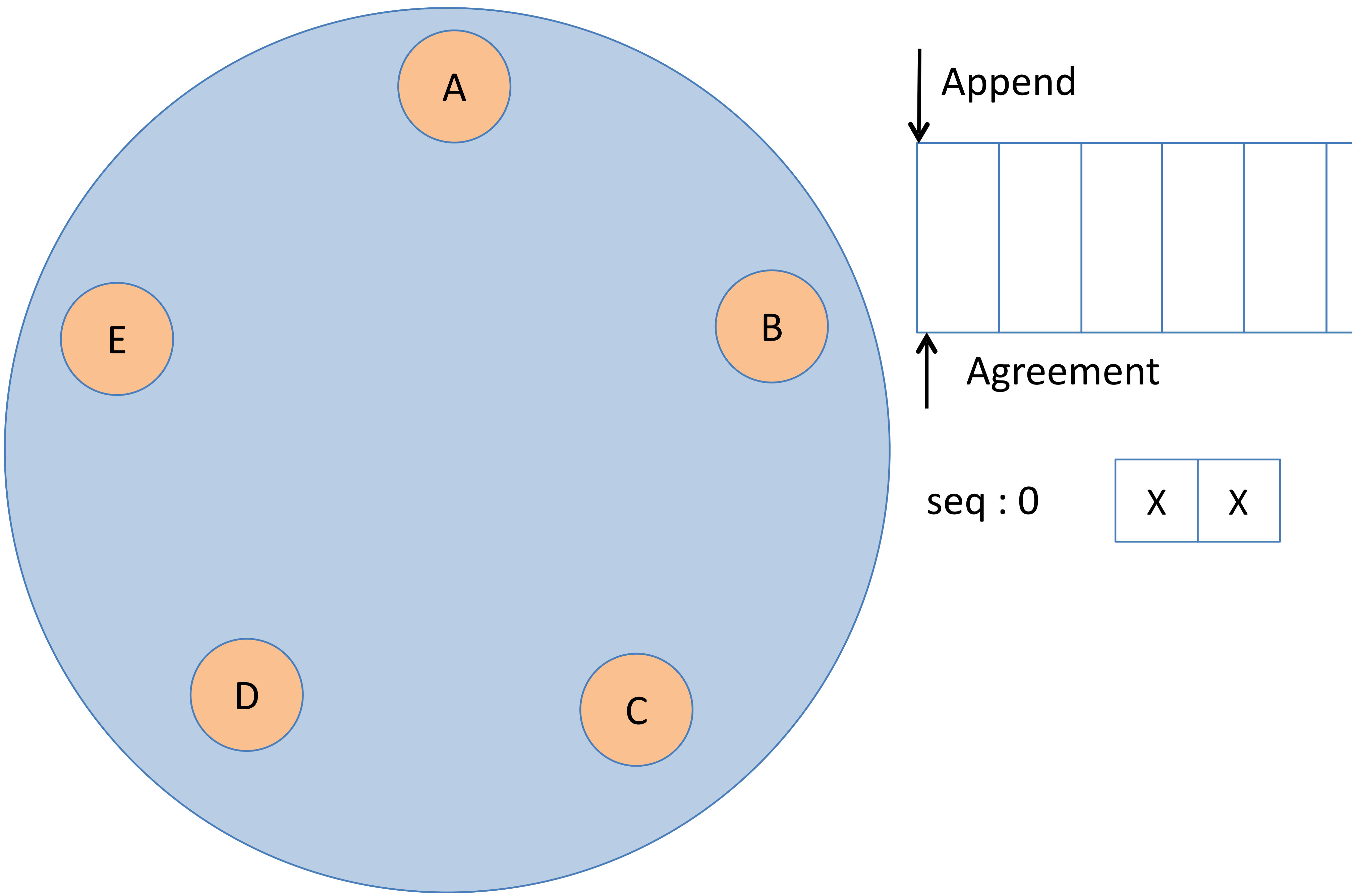
Anatomy of the Exo Protocol

And some maximum number of hosts ***n*** (5) labeled **A-E,** in a fixed, predetermined order



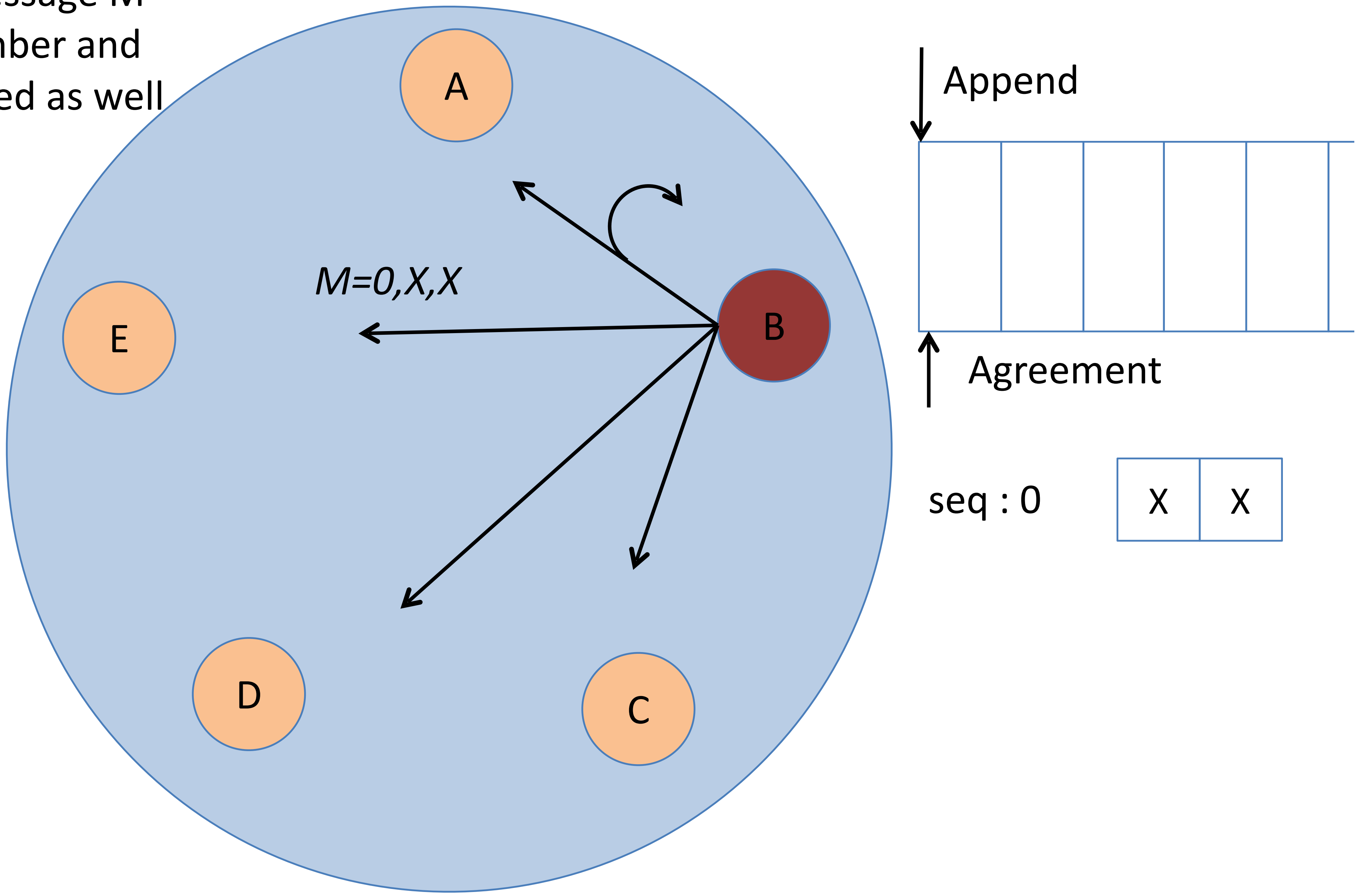
Anatomy of the Exo Protocol

Each host keeps the following state:
-append only log
-sequence number
-**n**/2 message history



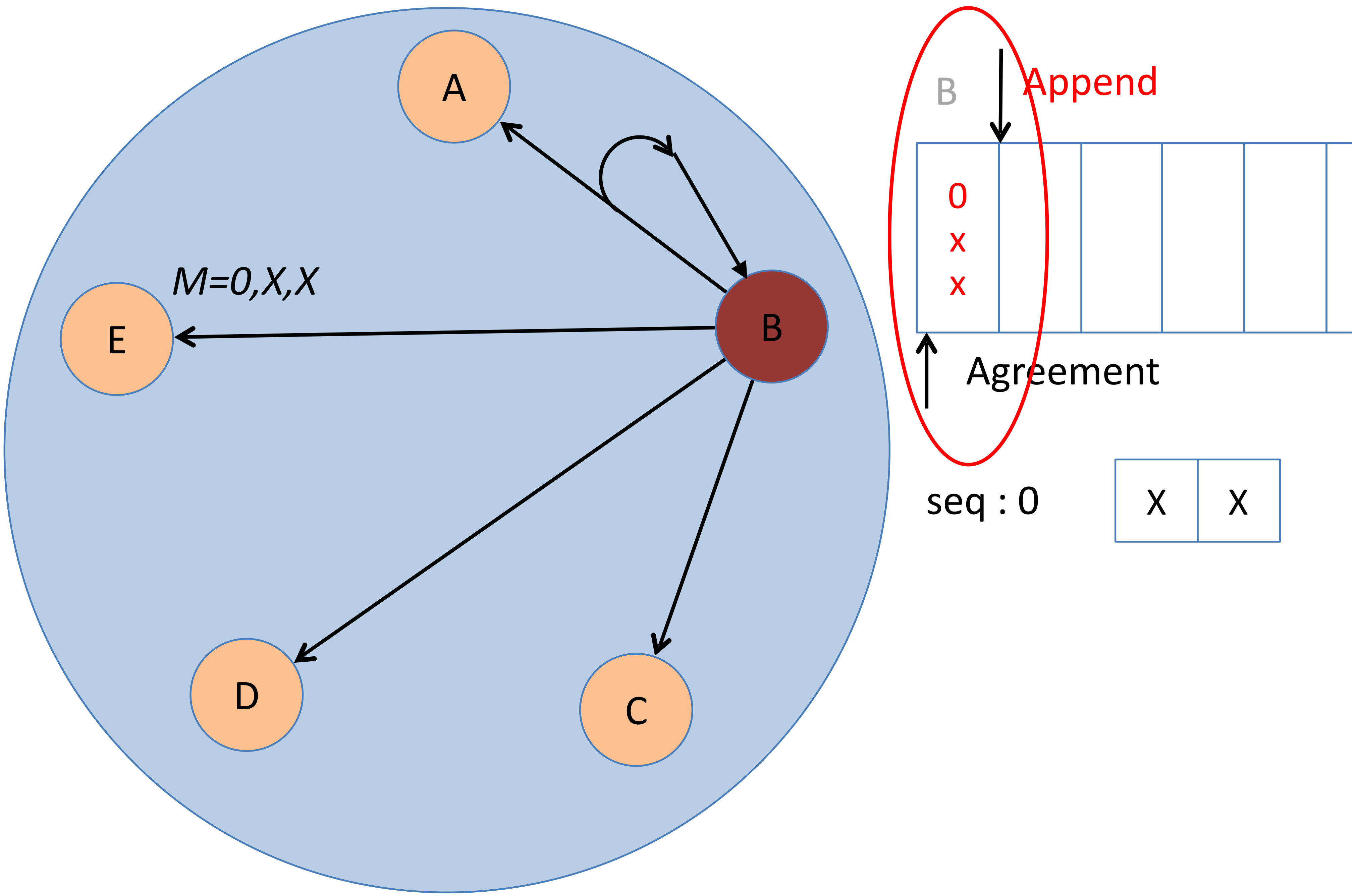
Anatomy of the Exo Protocol

Host *B* broadcasts a message M with the sequence number and message history included as well as any data (optional)



Anatomy of the Exo Protocol

Message from B arrives at all hosts, causes a:
-log append

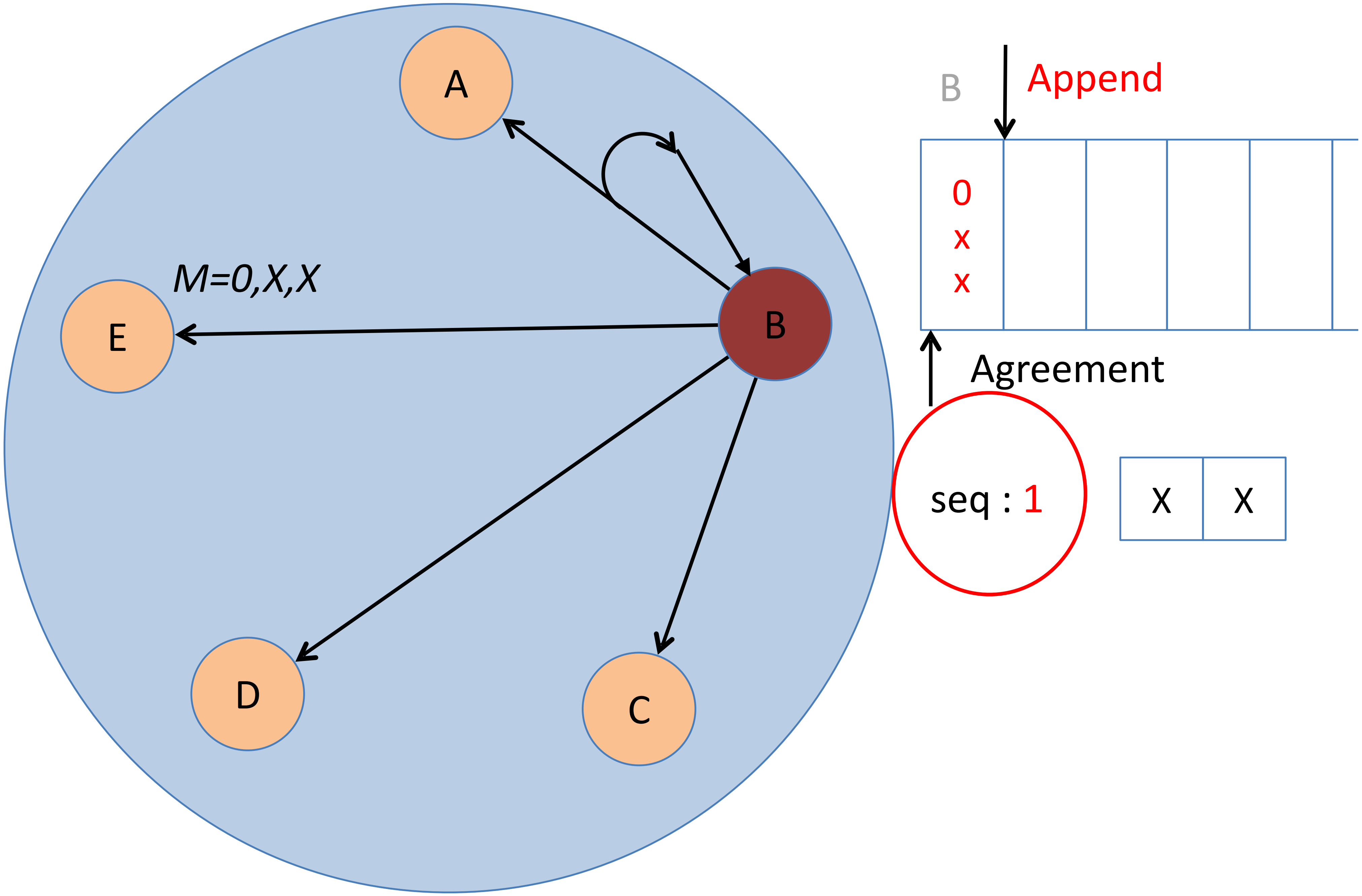


Anatomy of the Exo Protocol

Message from B arrives at all hosts, causes a:

-log append

-sequence number update



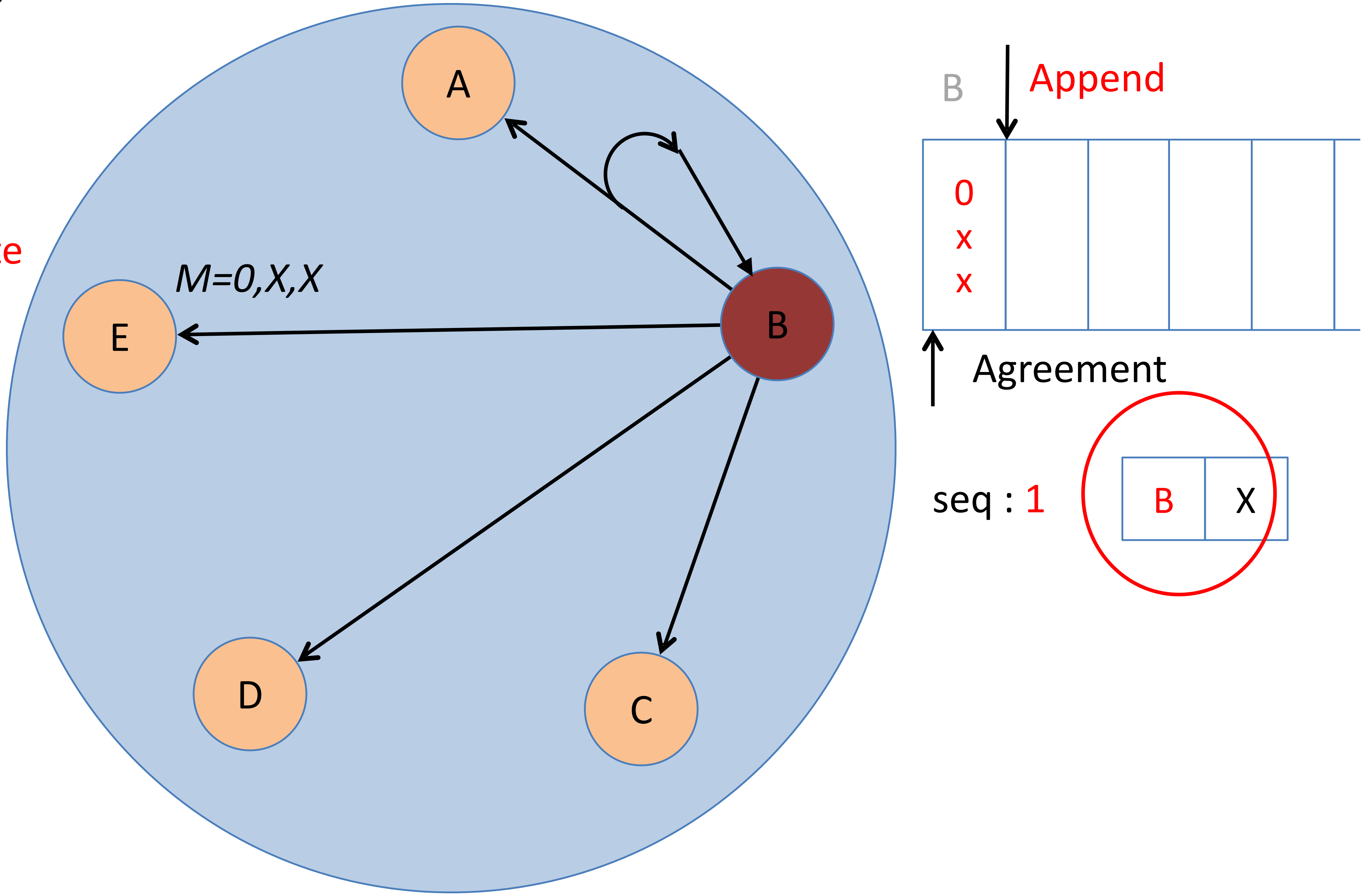
Anatomy of the Exo Protocol

Message from B arrives at all hosts, causes a:

-log append

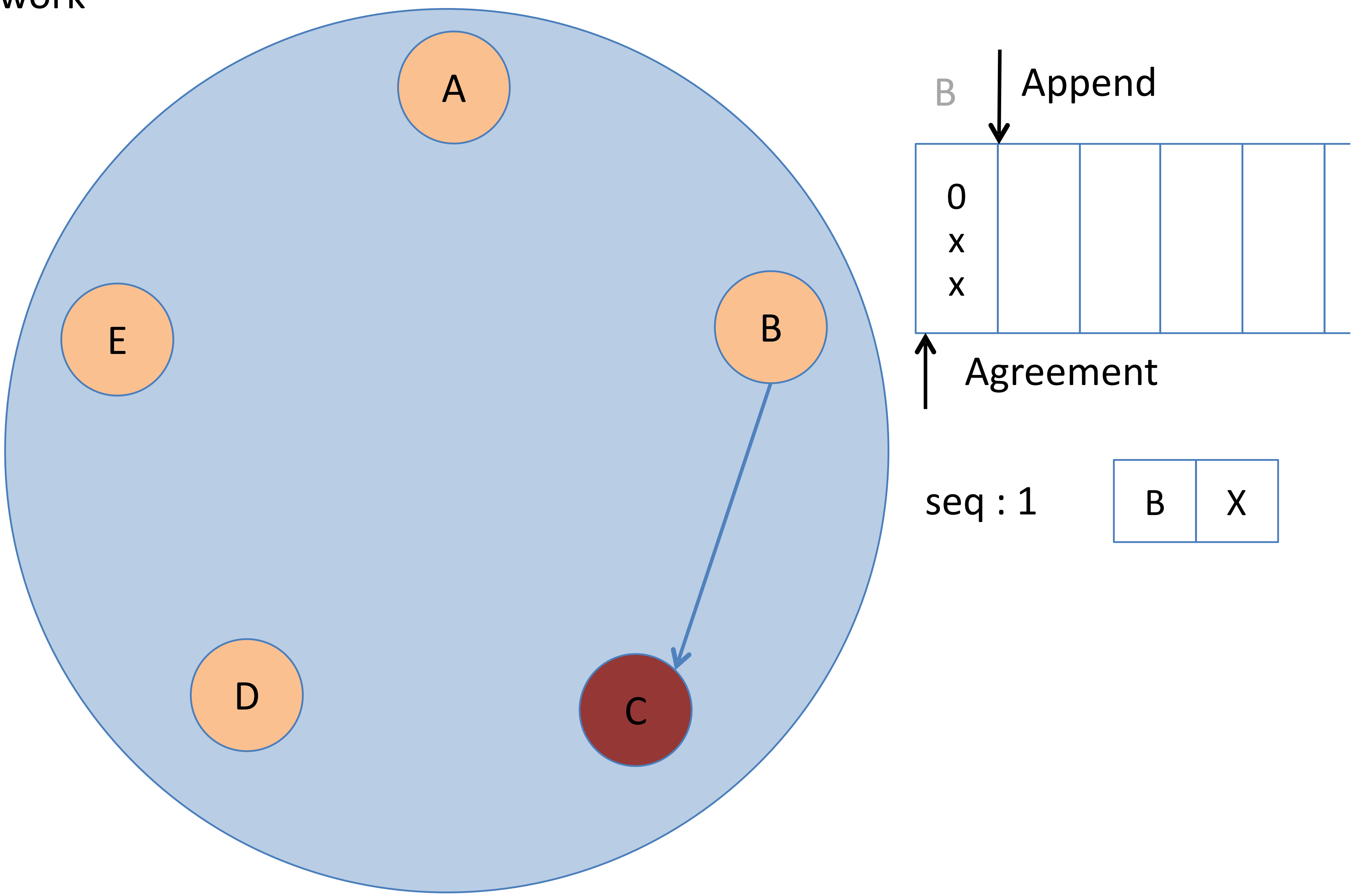
-sequence number update

-message history update



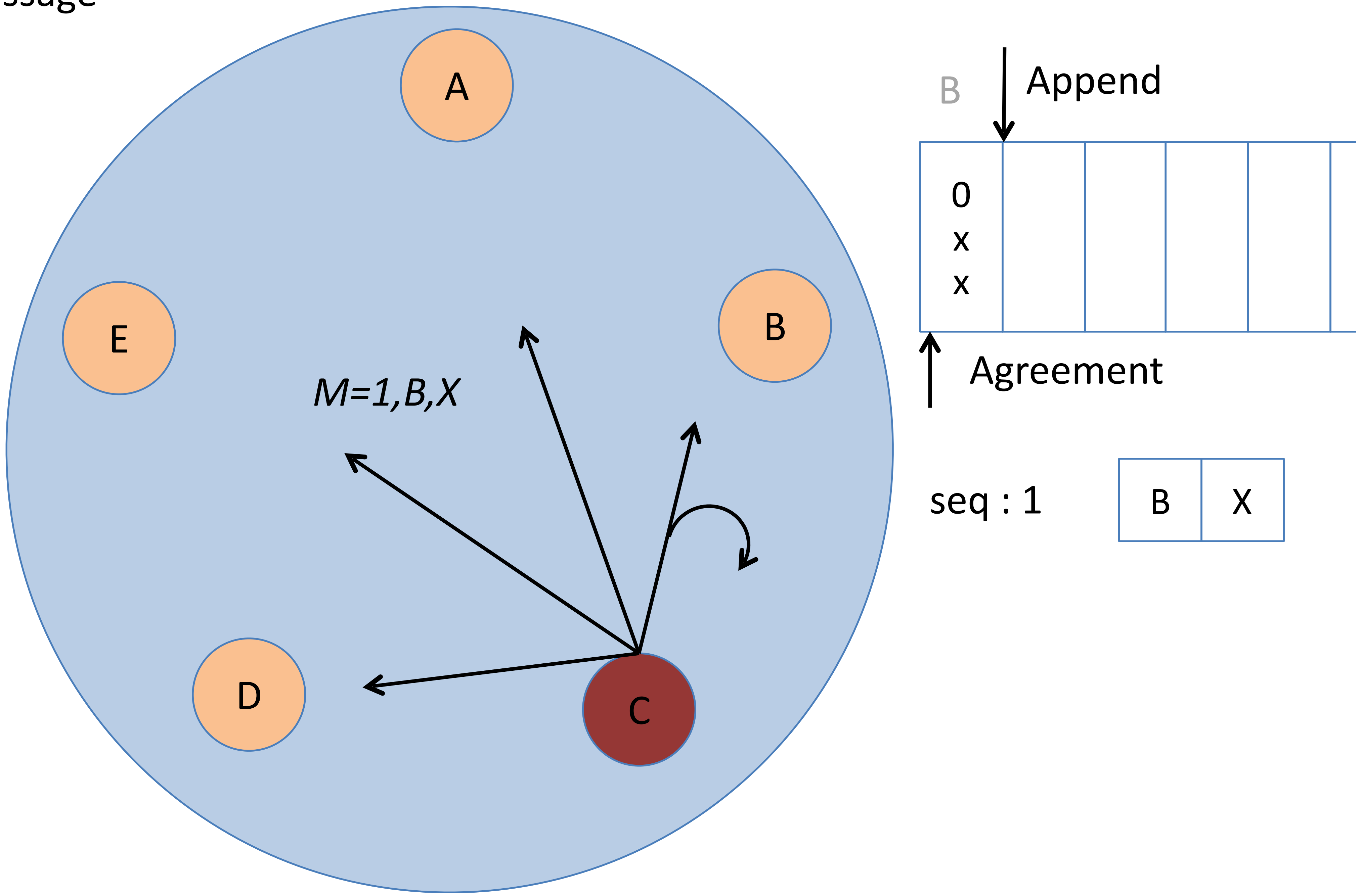
Anatomy of the Exo Protocol

Host C now has the network



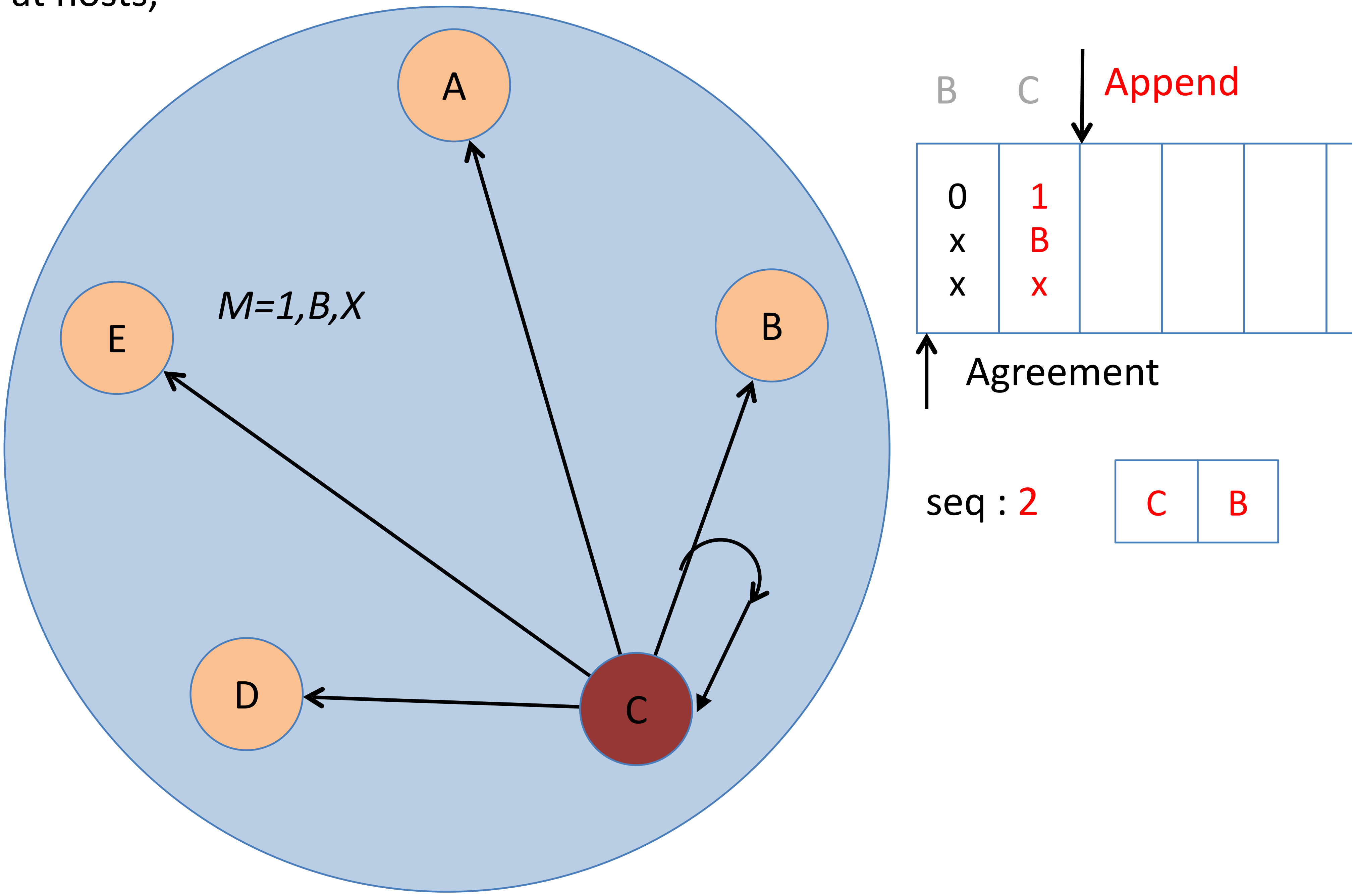
Anatomy of the Exo Protocol

Host *C* broadcasts a message



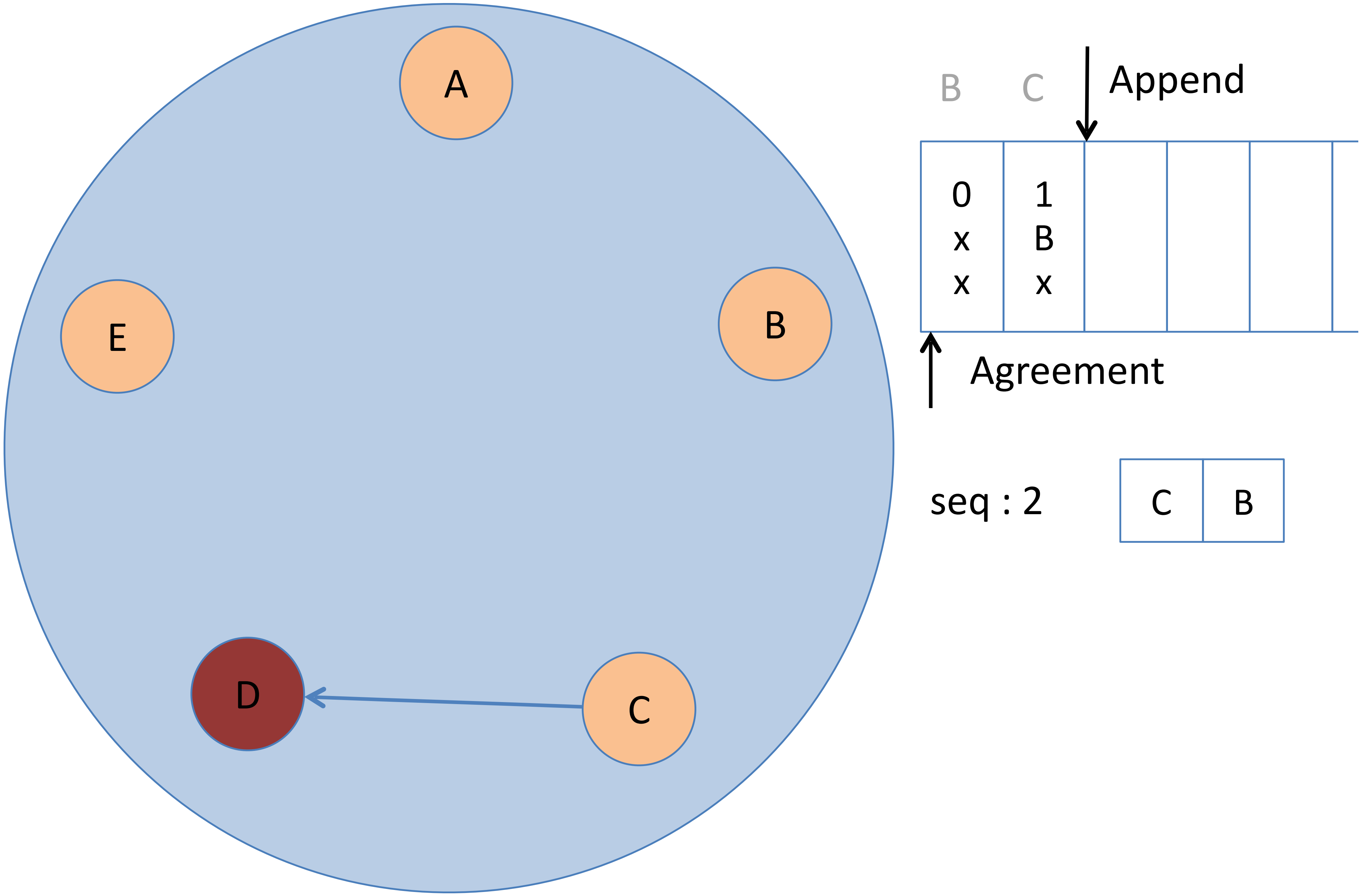
Anatomy of the Exo Protocol

Message from C arrives at hosts, causing update



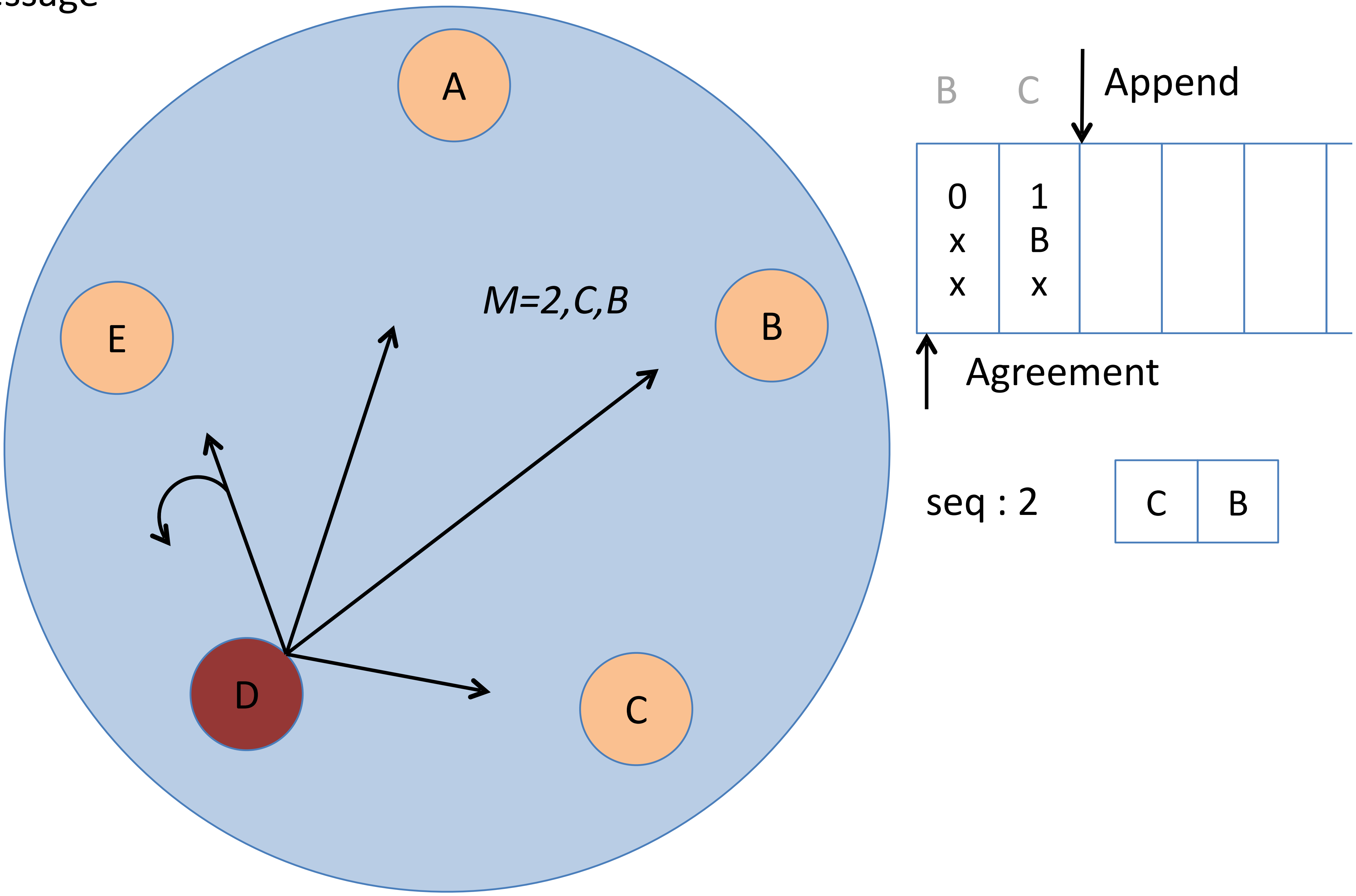
Anatomy of the Exo Protocol

D now has the token



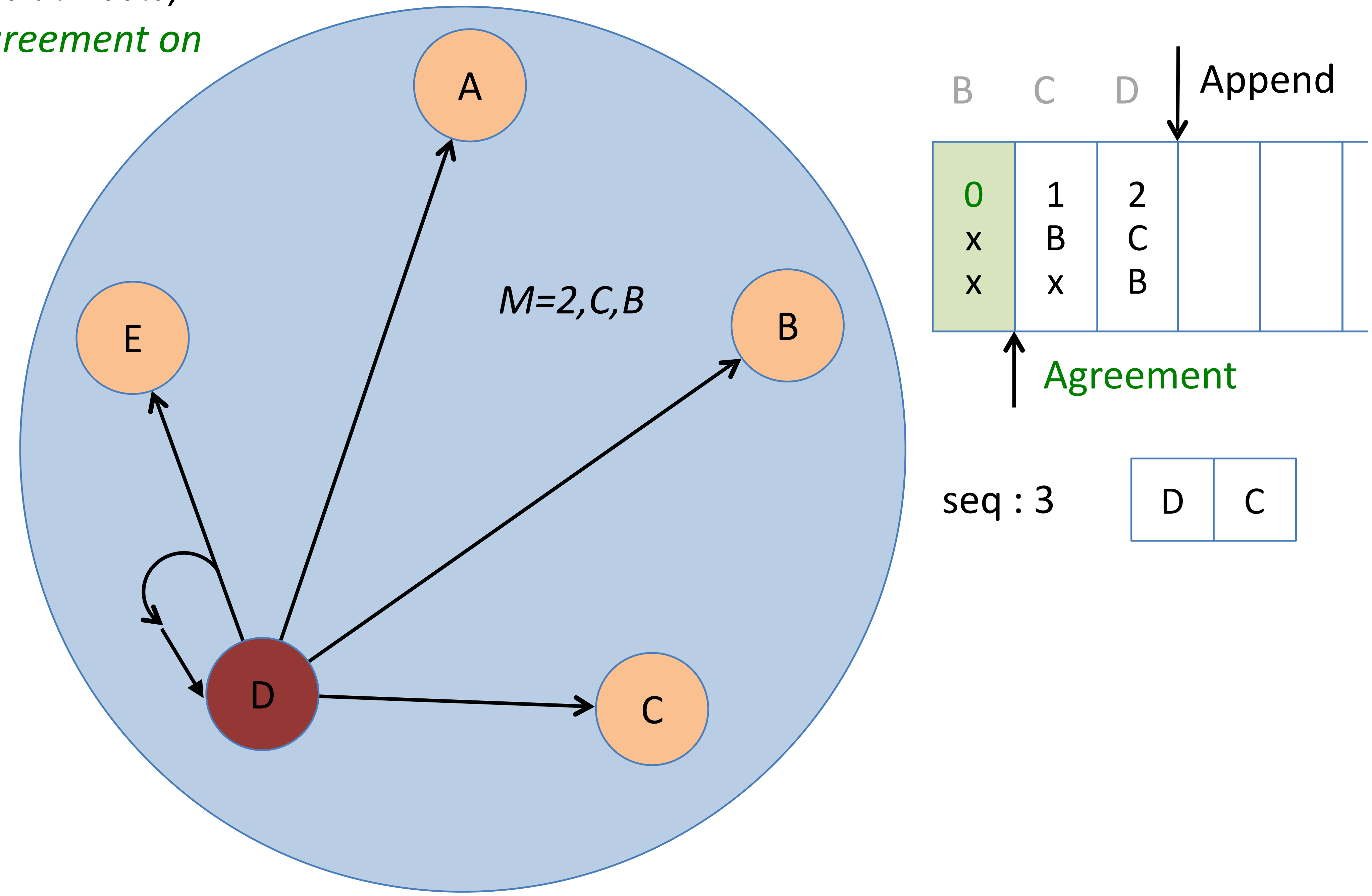
Anatomy of the Exo Protocol

Host *D* broadcasts a message



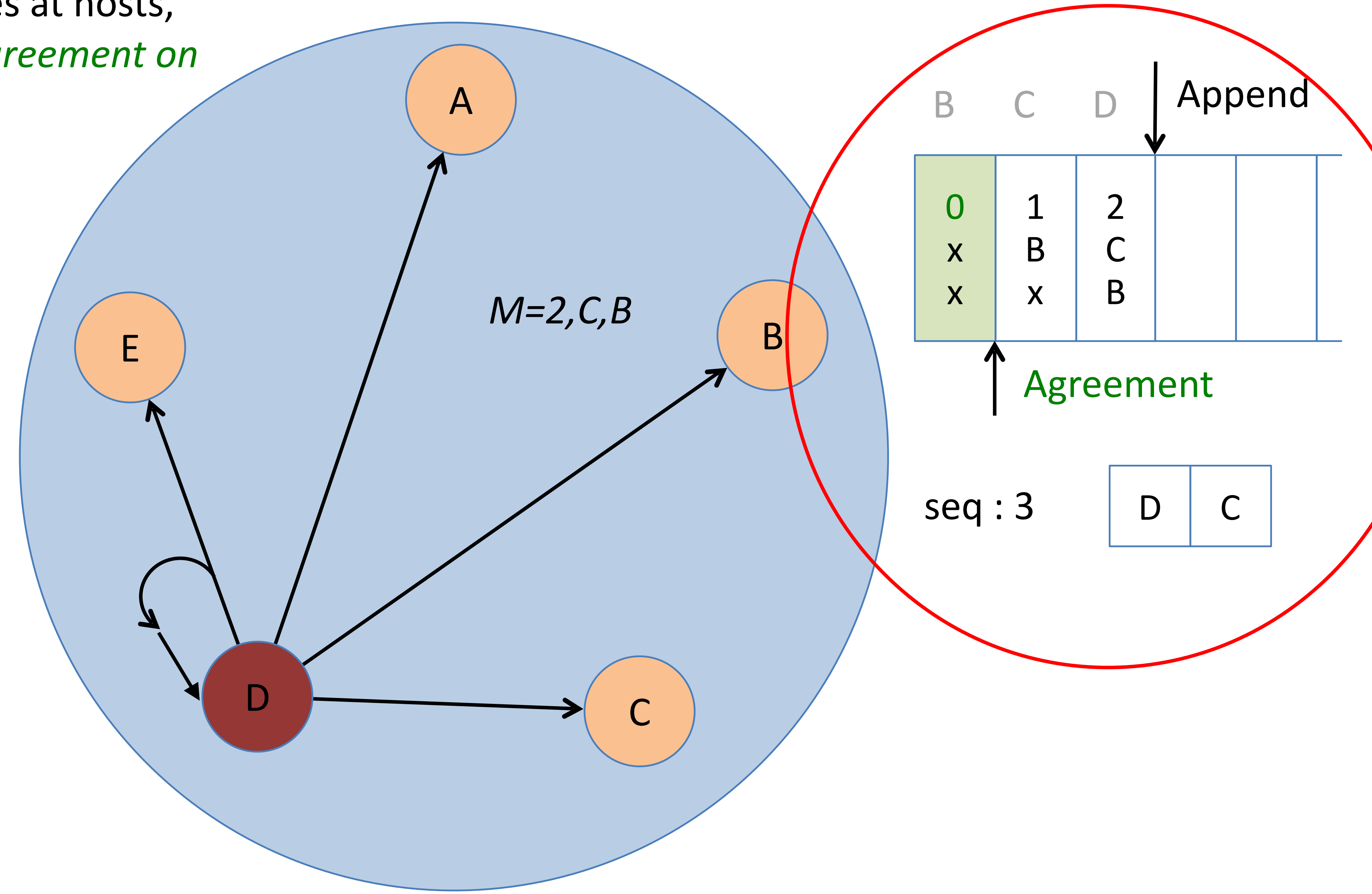
Anatomy of the Exo Protocol

Message from D arrives at hosts, causing *update and agreement on sequence number 0*

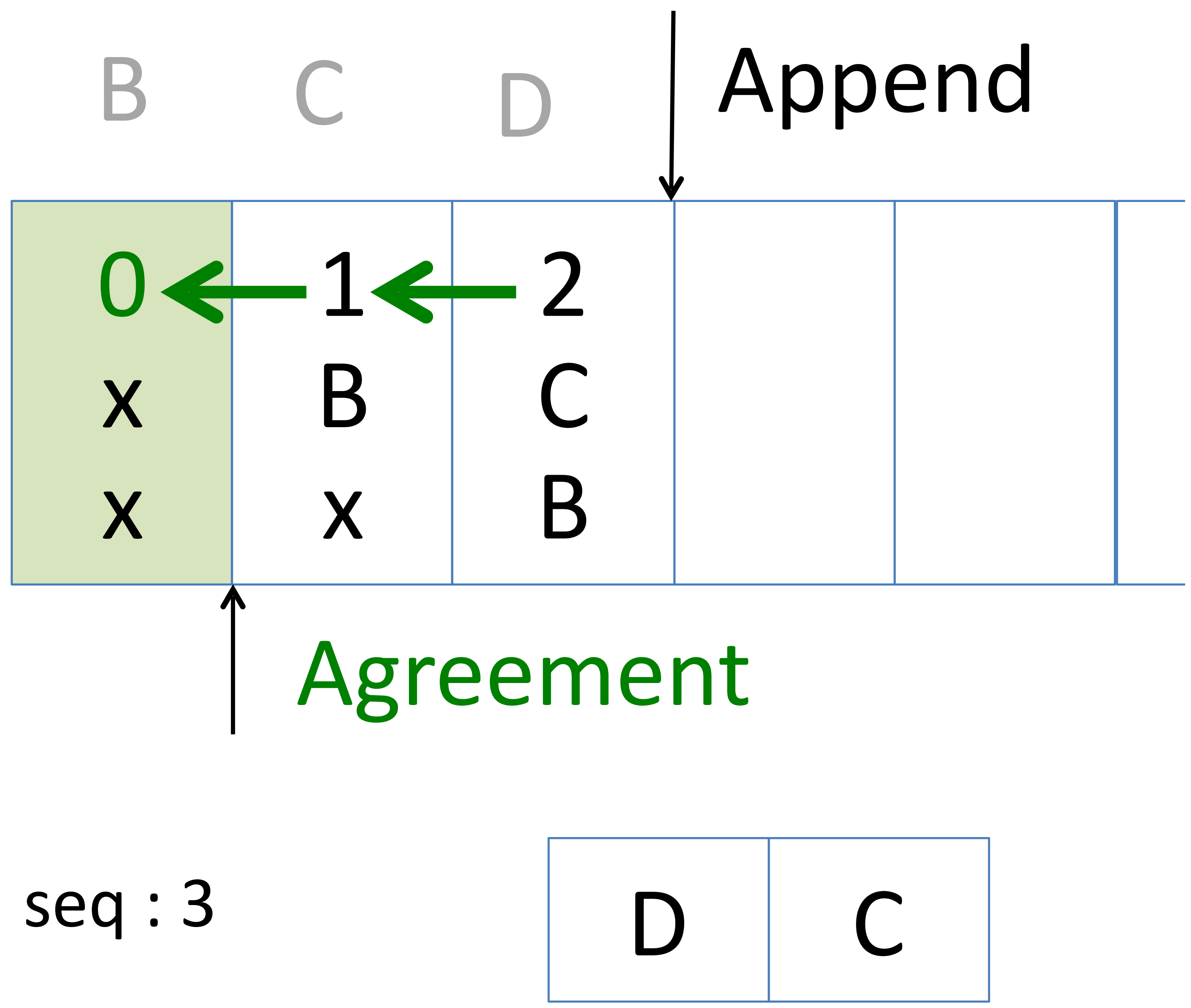


Zooming in…

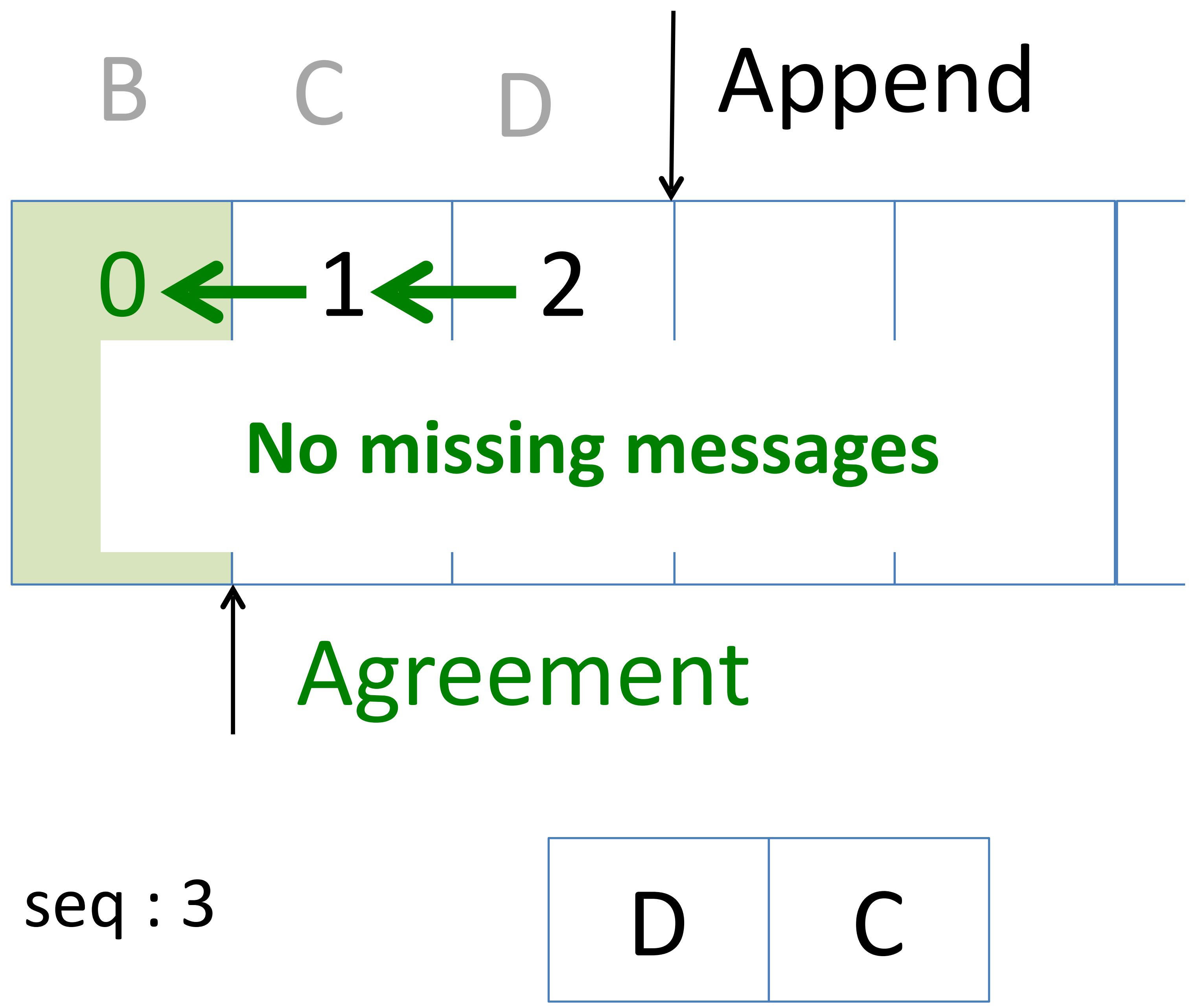
Message from D arrives at hosts, causing *update and* *agreement on sequence number 0*



Zooming in…

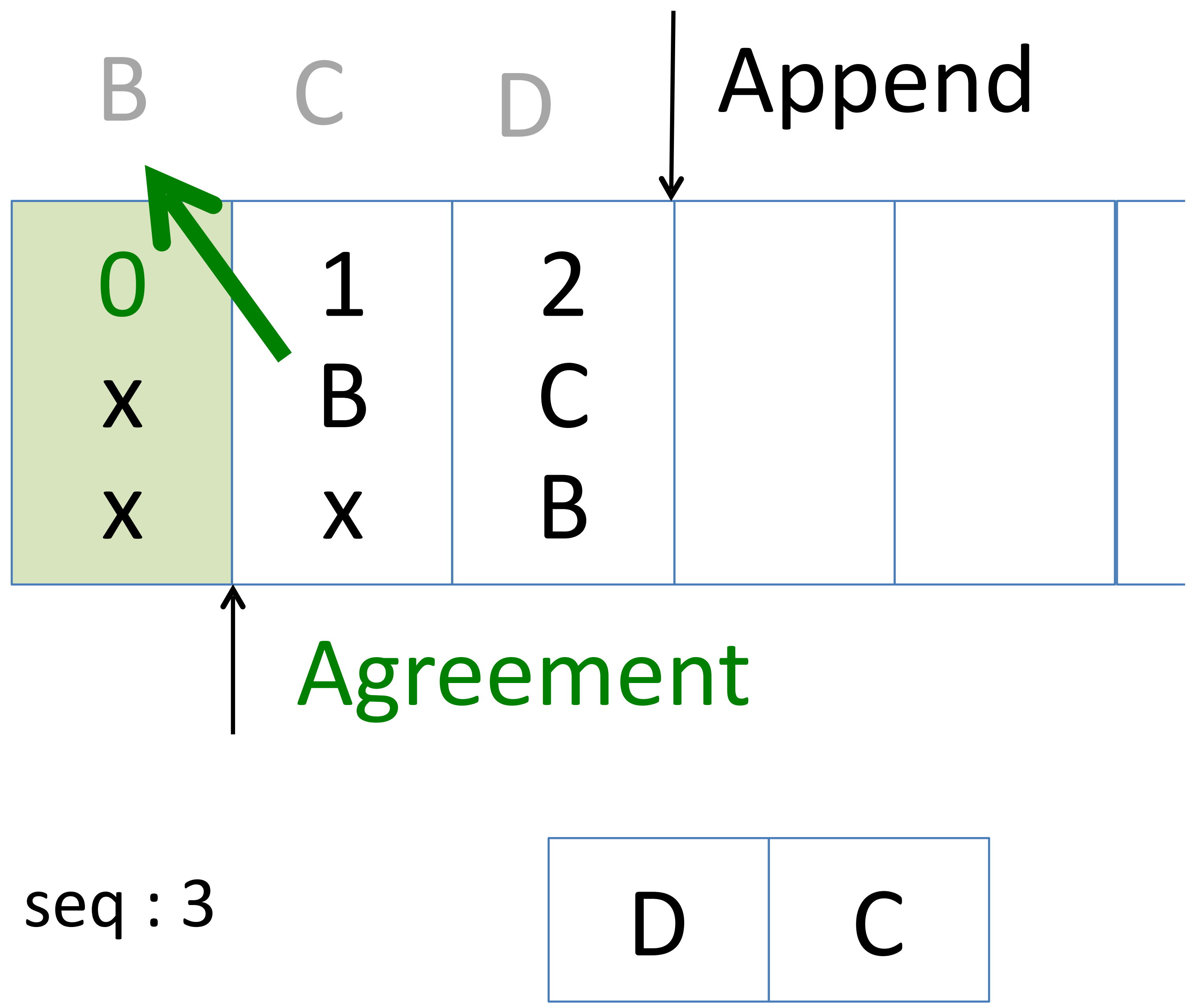


Zooming in…

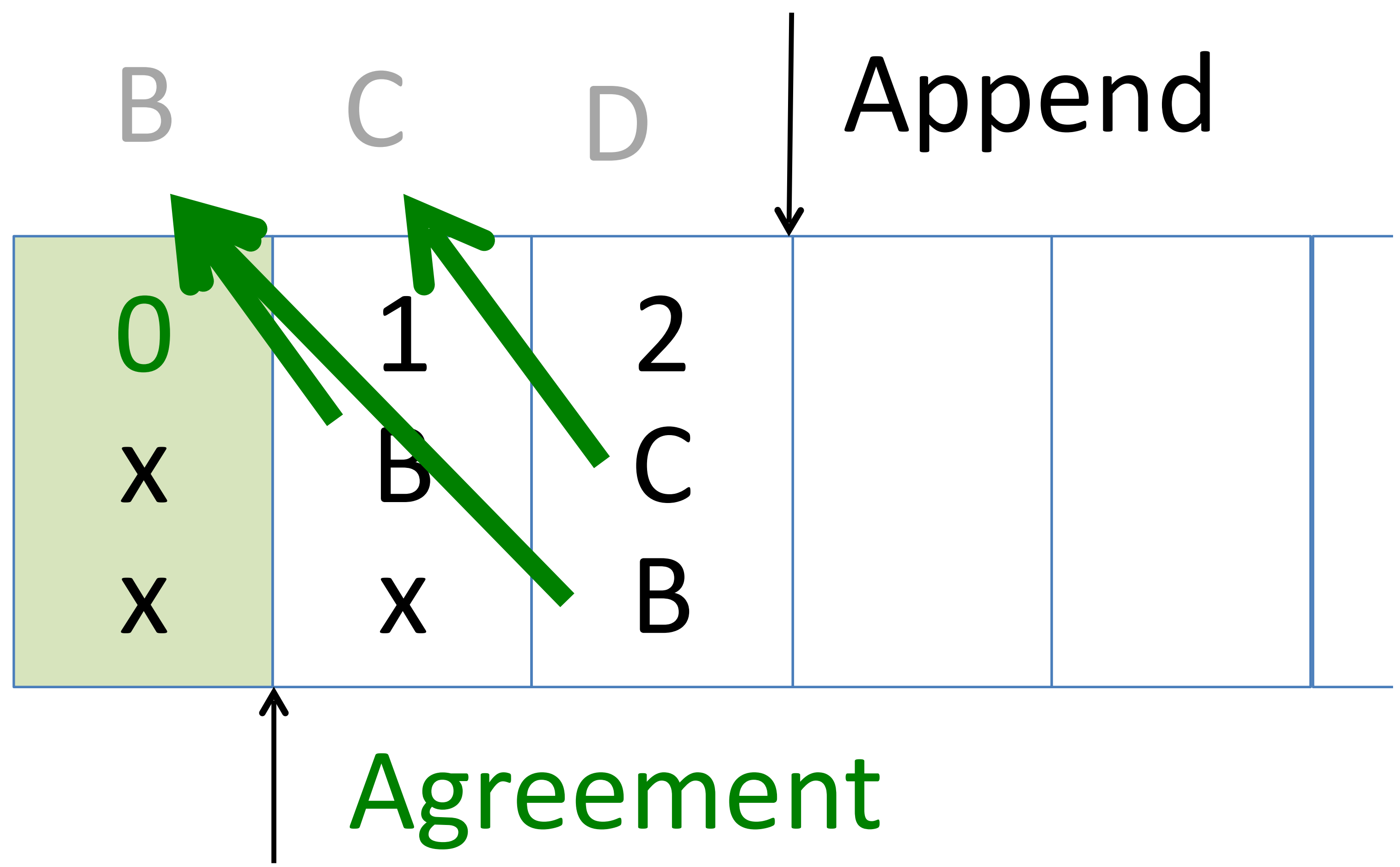


NB: Using a 64bit sequence number gives >400 years operation at 1 message per nano-second.

Zooming in…



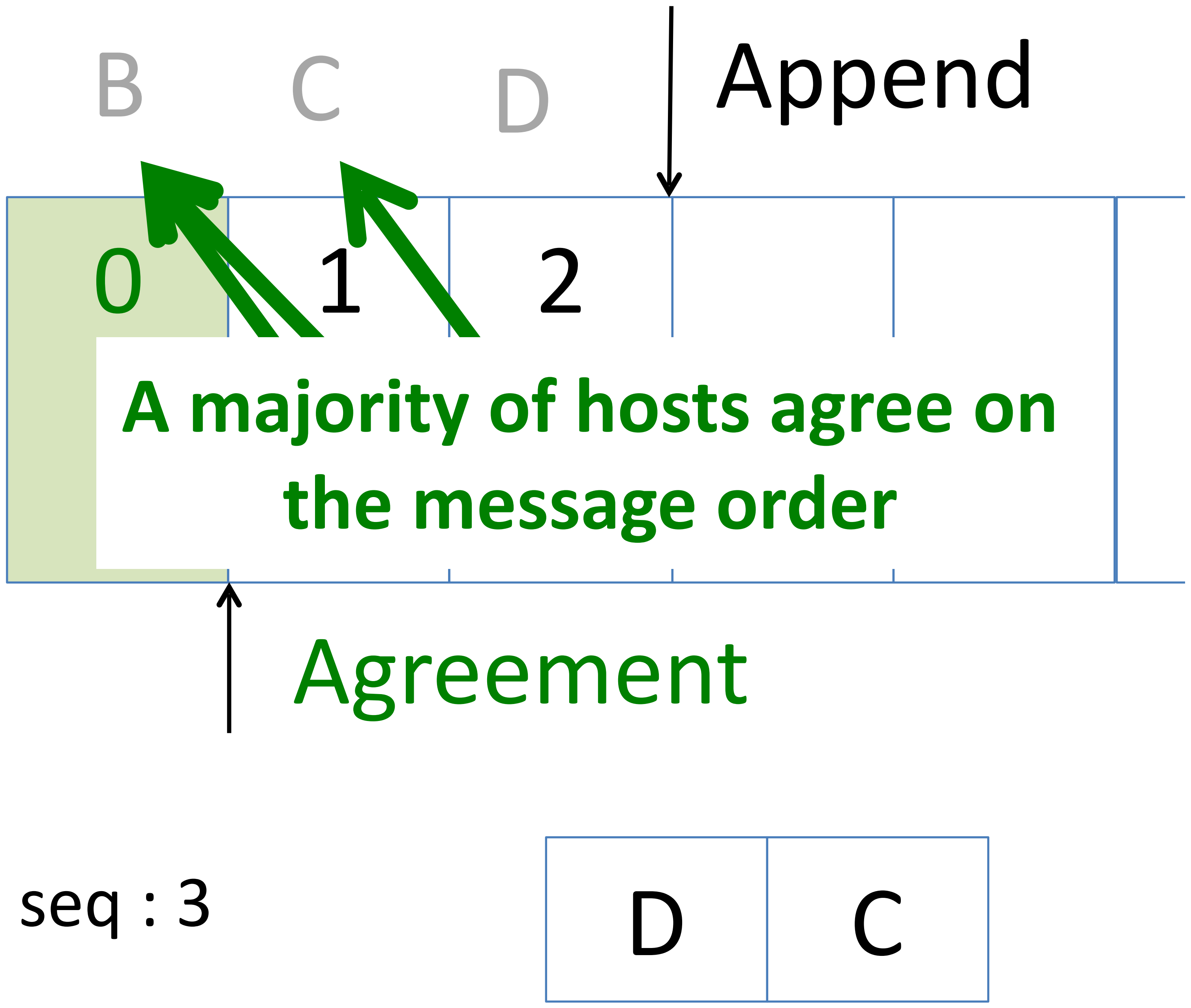
Zooming in…



seq : 3

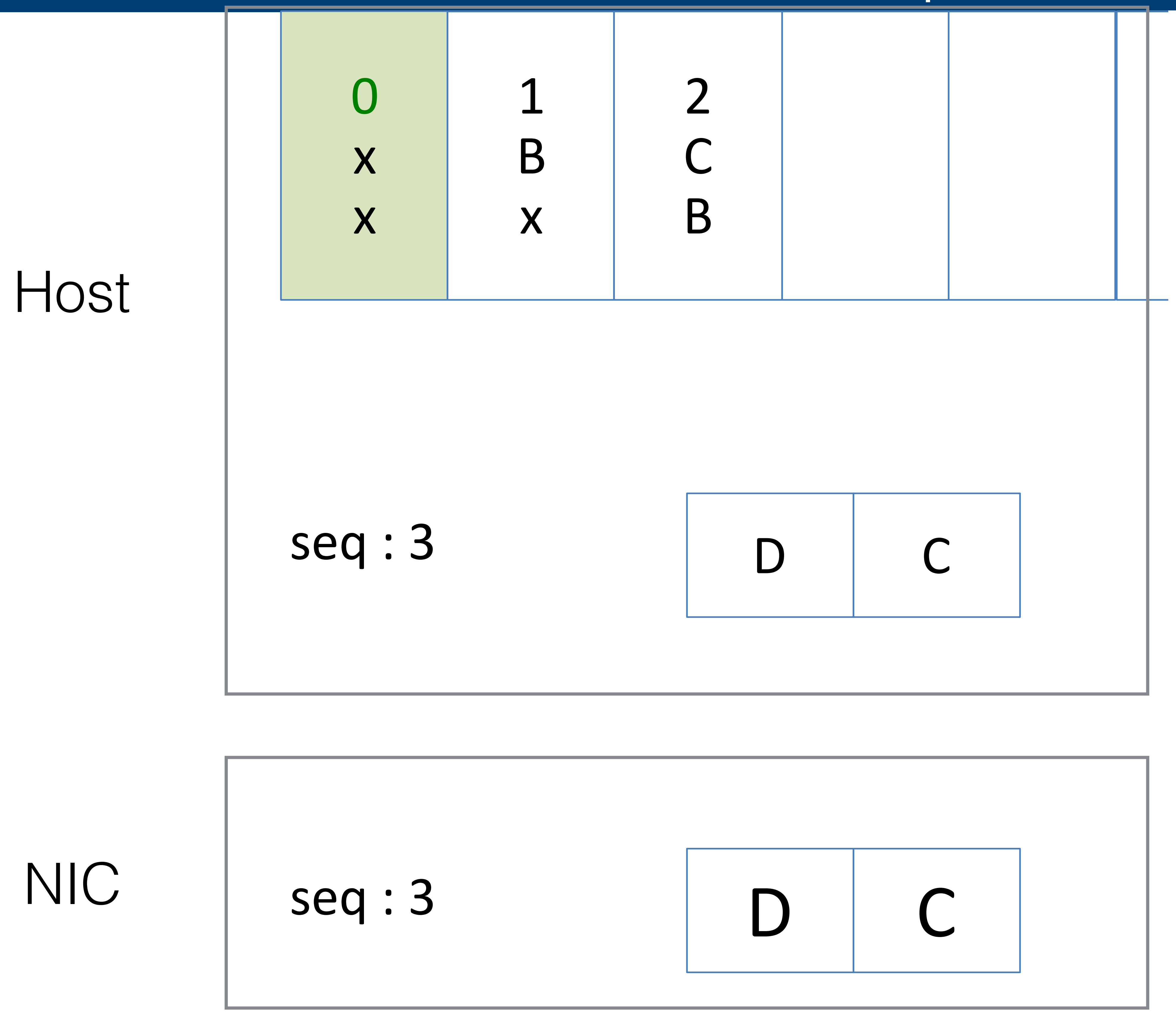
D	C

Zooming in…



NB: Assuming at most 256 nodes, message history is 128B and can

How to accelerate the protocol?



Implementation

- Protocol implemented over CamIO
  - Runs on TCP, Broadcast UDP, Raw Frames and OpenOnload (ExaSock)
- Offload engine (mostly) implemented in FPGA
  - Currently working for empty messages.

Results

Linux TCP	36µs/msg	28K msg/sec)
Linux UDP	29µs/msg	(34K msg/sec)
Linux UDP (broadcast)	16µs/msg	(62K msg/sec)
Linux TCP (loopback)	9µs/msg	(110K msg/sec)
Linux UDP (loopback)	4us/msg	(231K msg/sec)
Libexanic (raw frames)	3µs/msg	(359K msg/sec)
Exo HW offload	0.4µs/msg	(2500K msg/sec)

Results

Linux TCP	**36μs/msg**	**28K msg/sec)**
Linux UDP	29µs/msg	(34K msg/sec)
Linux UDP (broadcast)	16µs/msg	(62K msg/sec)
Linux TCP (loopback)	9µs/msg	(110K msg/sec)
Linux UDP (loopback)	4us/msg	(231K msg/sec)
Libexanic (raw frames)	3µs/msg	(359K msg/sec)
Exo HW offload	**0.4μs/msg**	**(2500K msg/sec)**

≈ 100x Faster

Concuslisions

- Exo is a work in progress
- We use a specialised network and hardware offload to build fast scalable coordination for rack-scale architectures
- Initial results suggest at least a 100x speed improvement over existing protocols/systems

Questions?

Thanks to our sponsors

UNIVERSITY OF CAMBRIDGE

EXABLAZE

EPSRC

Engineering and Physical Sciences Research Council

Research

Systems

Group

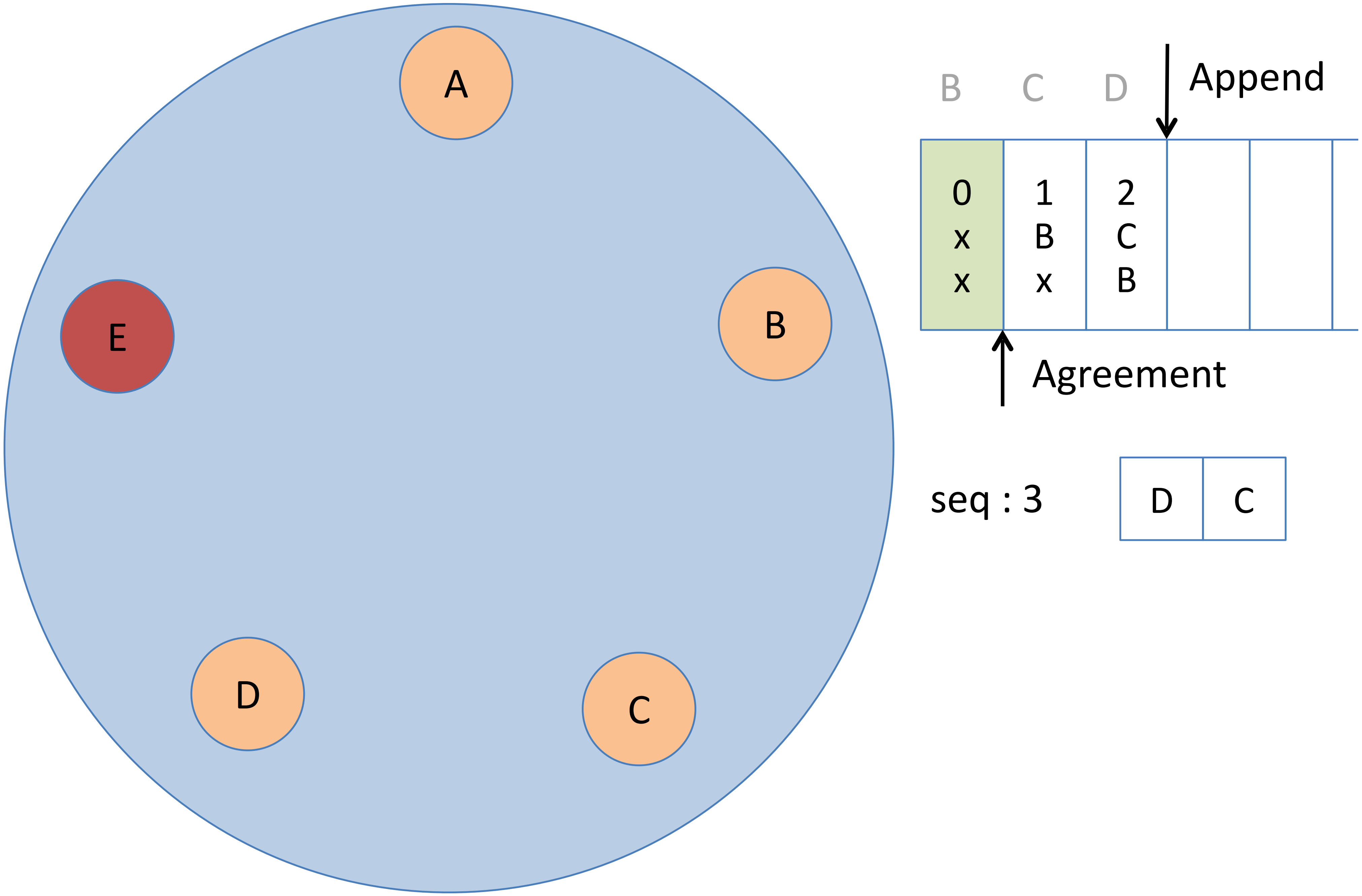
srg@cambridge:~$

This work was jointly supported by the EPSRC INTERNET Project EP/H040536/1 and the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-11-C-0249. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Backups

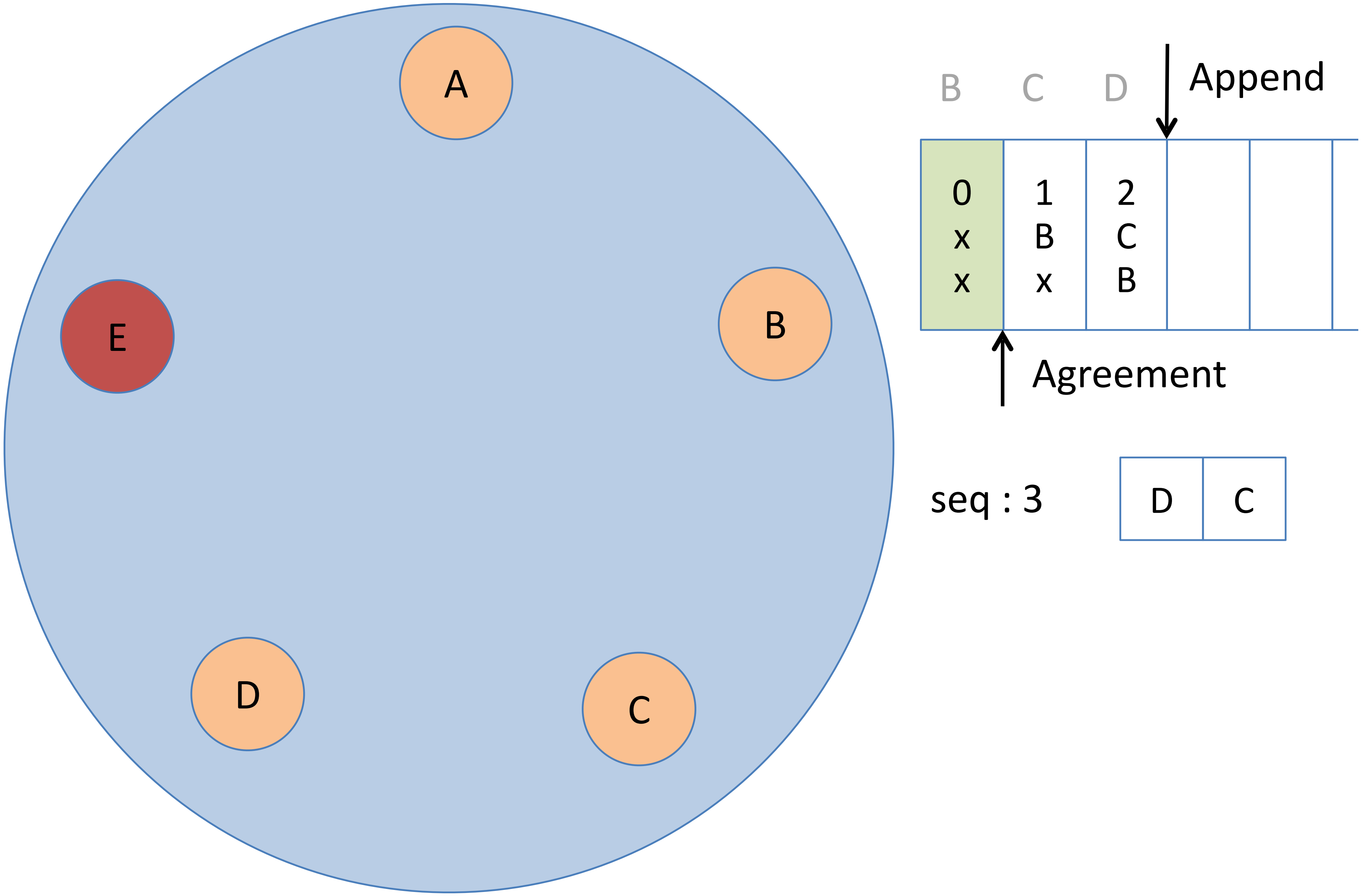
Anatomy of the Exo Protocol

E now has the token



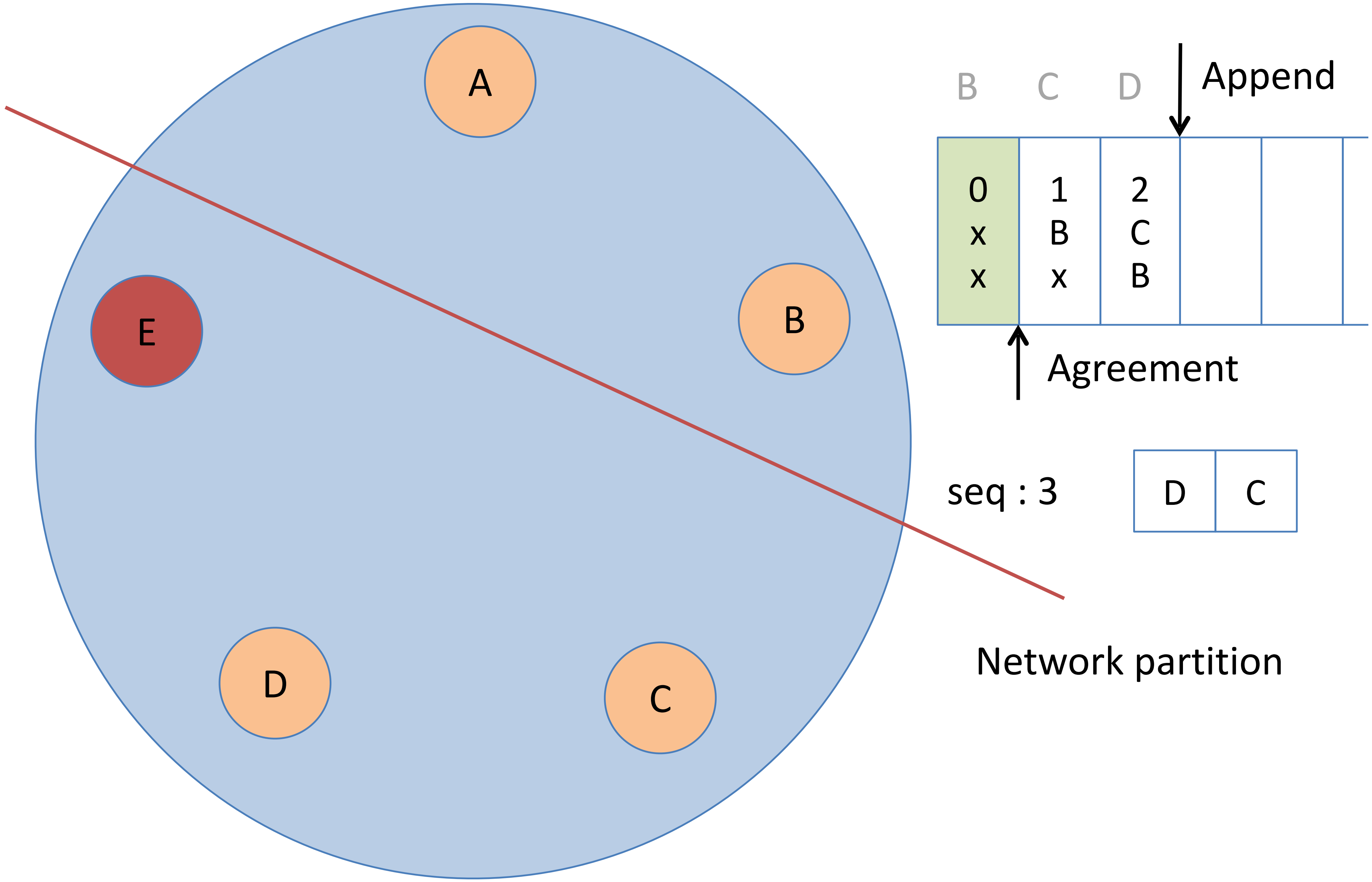
Handling Errors?

E now has the token



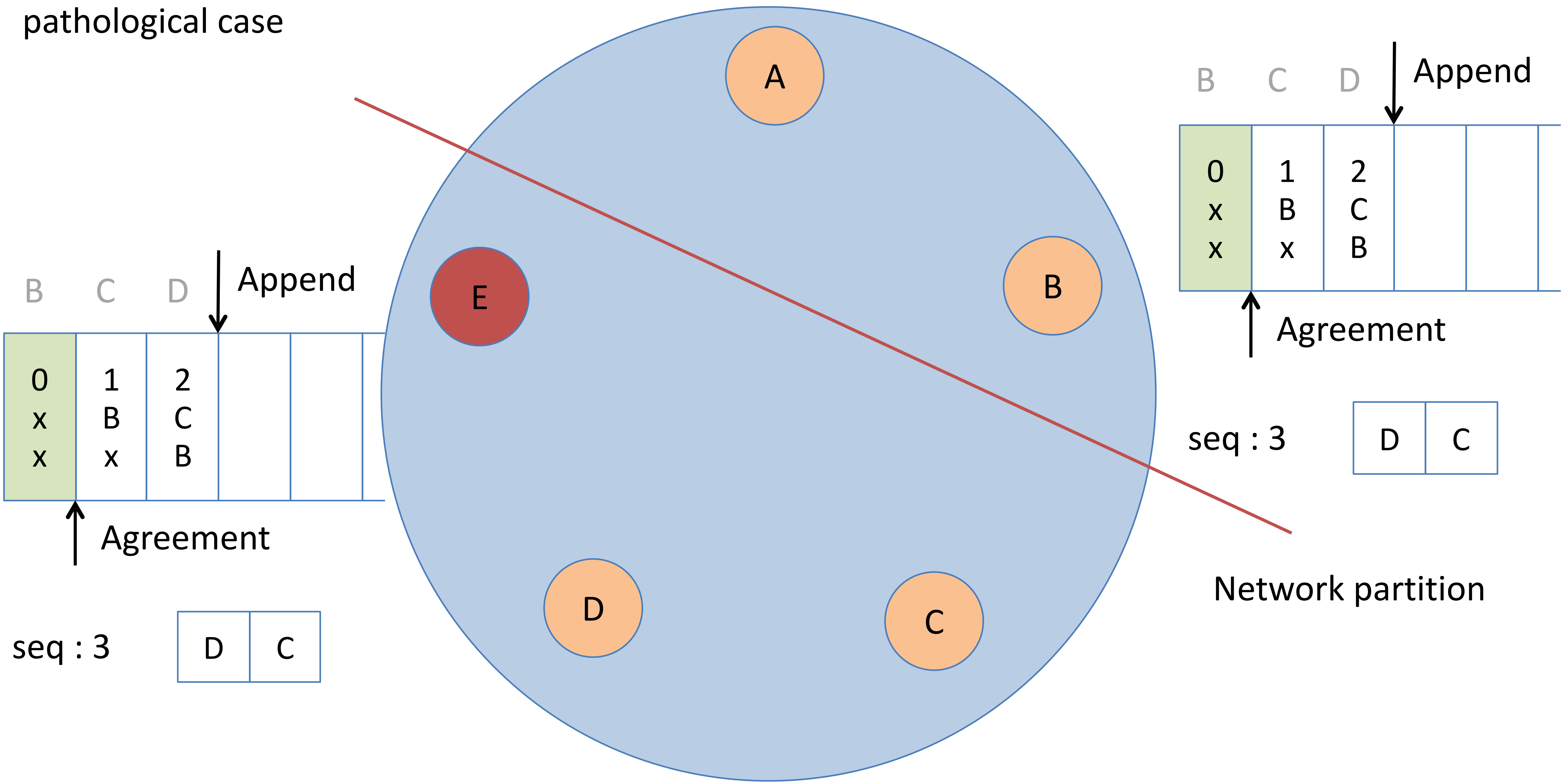
Handling Errors?

Consider the following pathological case



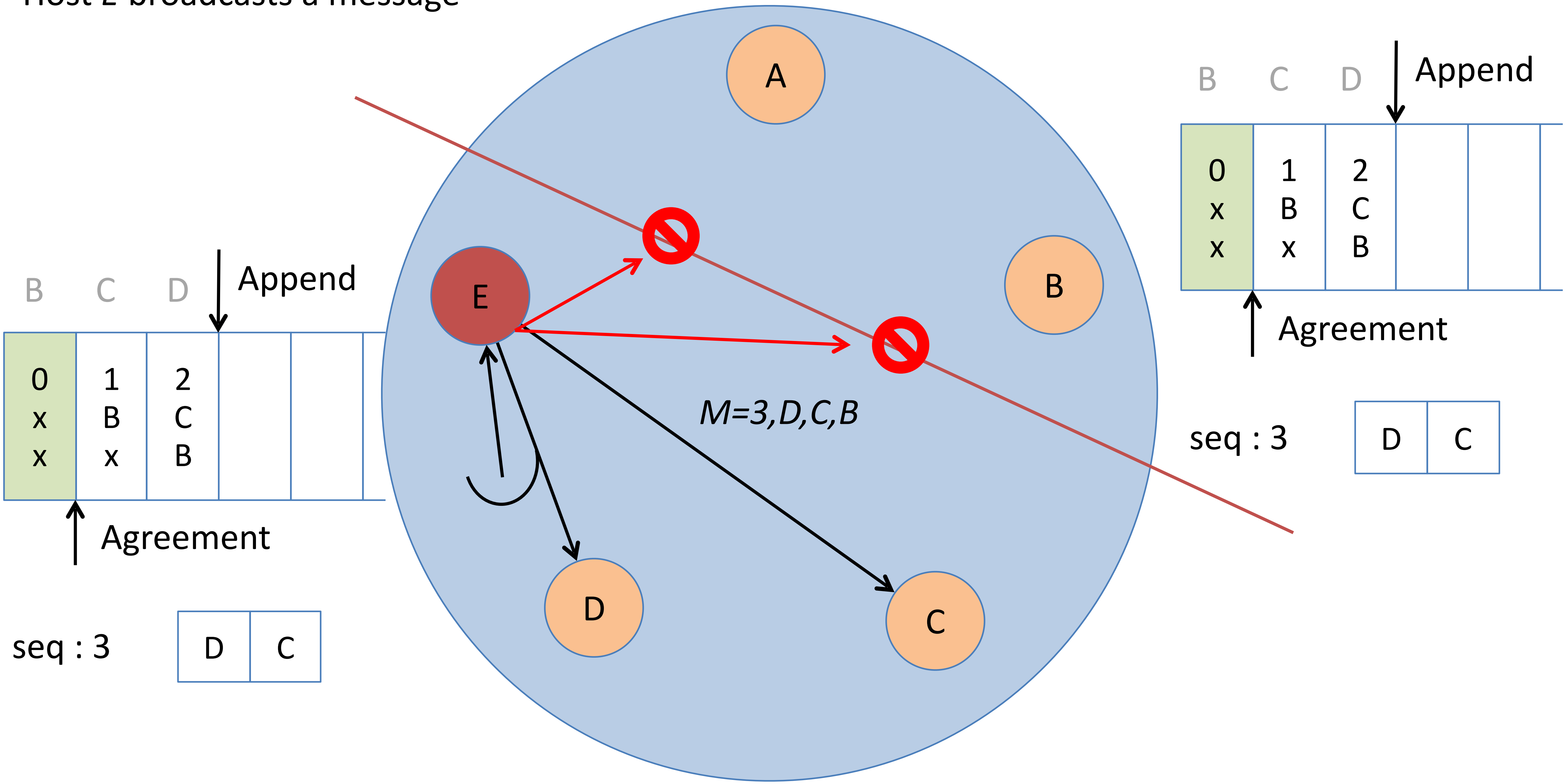
Handling Errors?

Consider the following pathological case



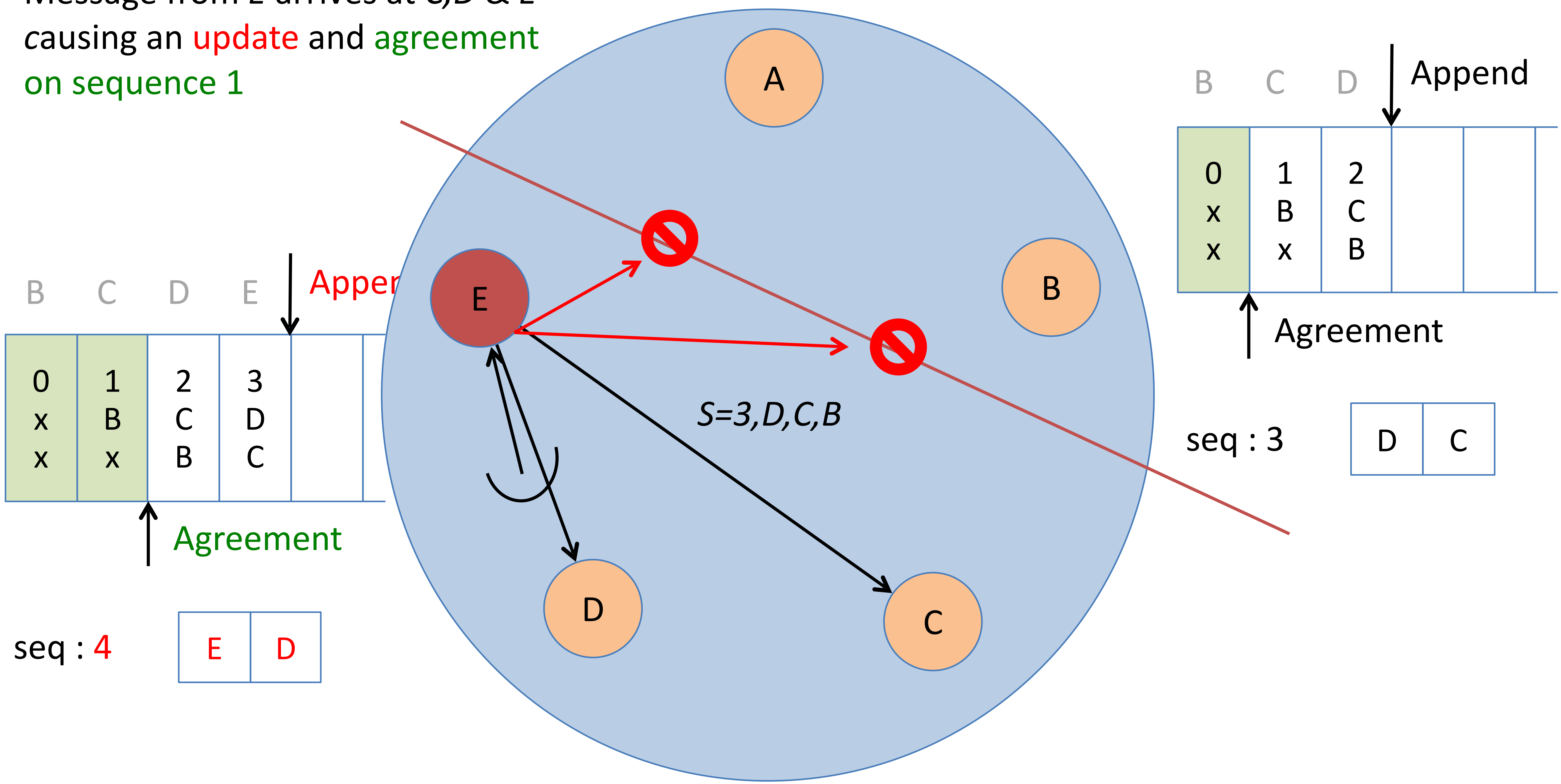
Handling Errors?

Host *E* broadcasts a message



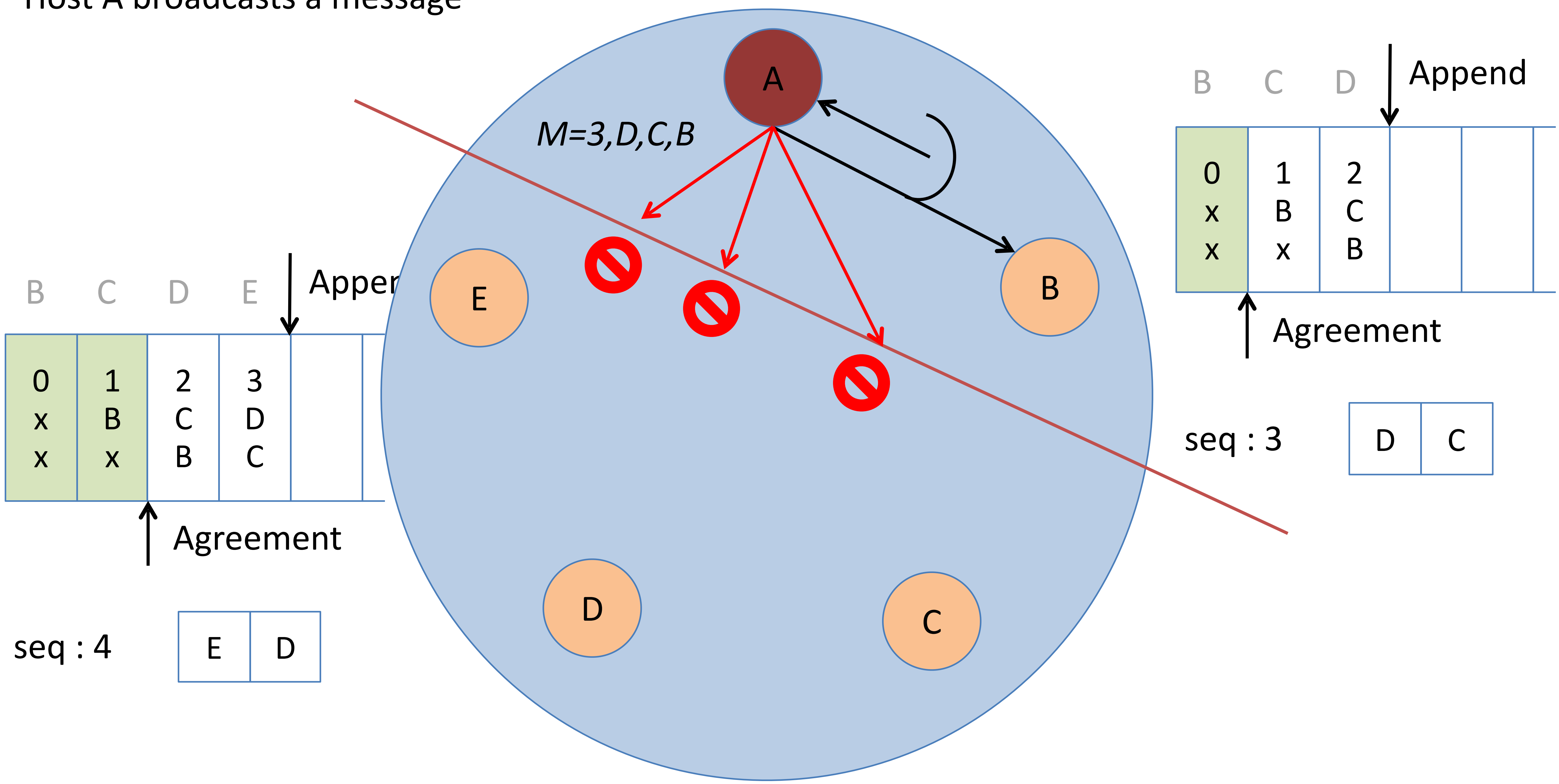
Handling Errors?

Message from *E* arrives at *C,D* & *E* causing an update and agreement on sequence 1



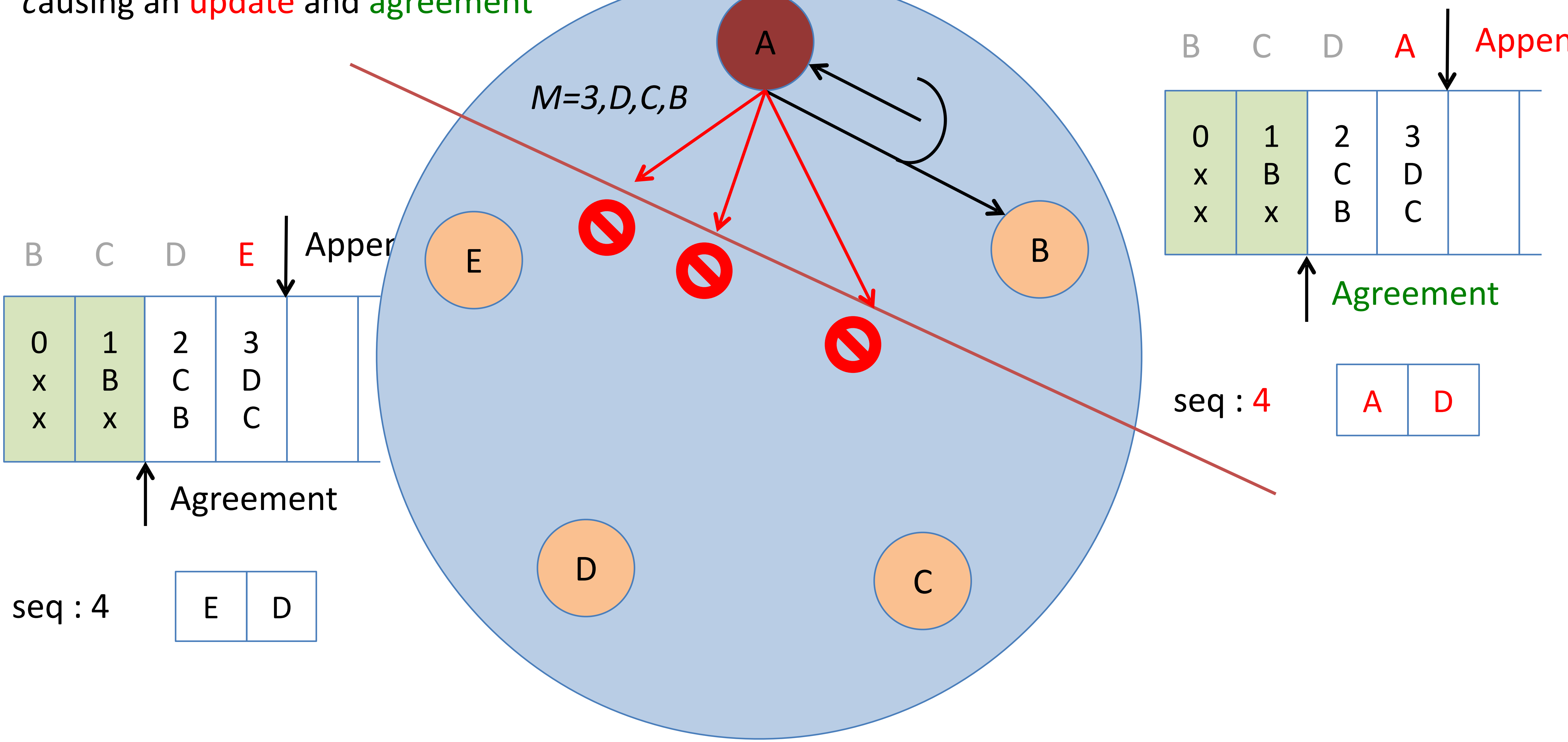
Handling Errors?

Host A broadcasts a message



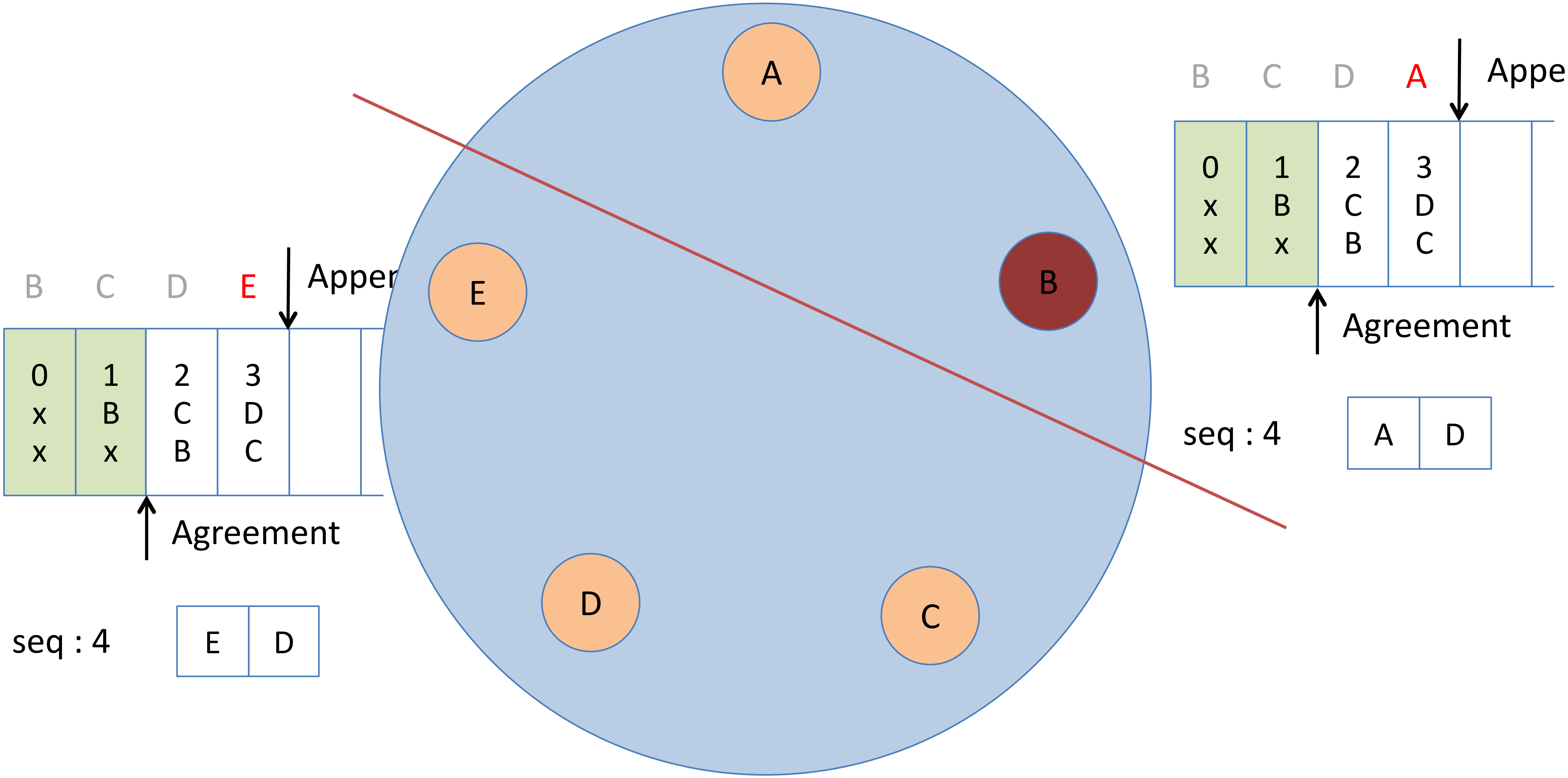
Handling Errors?

Message from A arrives at A & B, causing an update and agreement



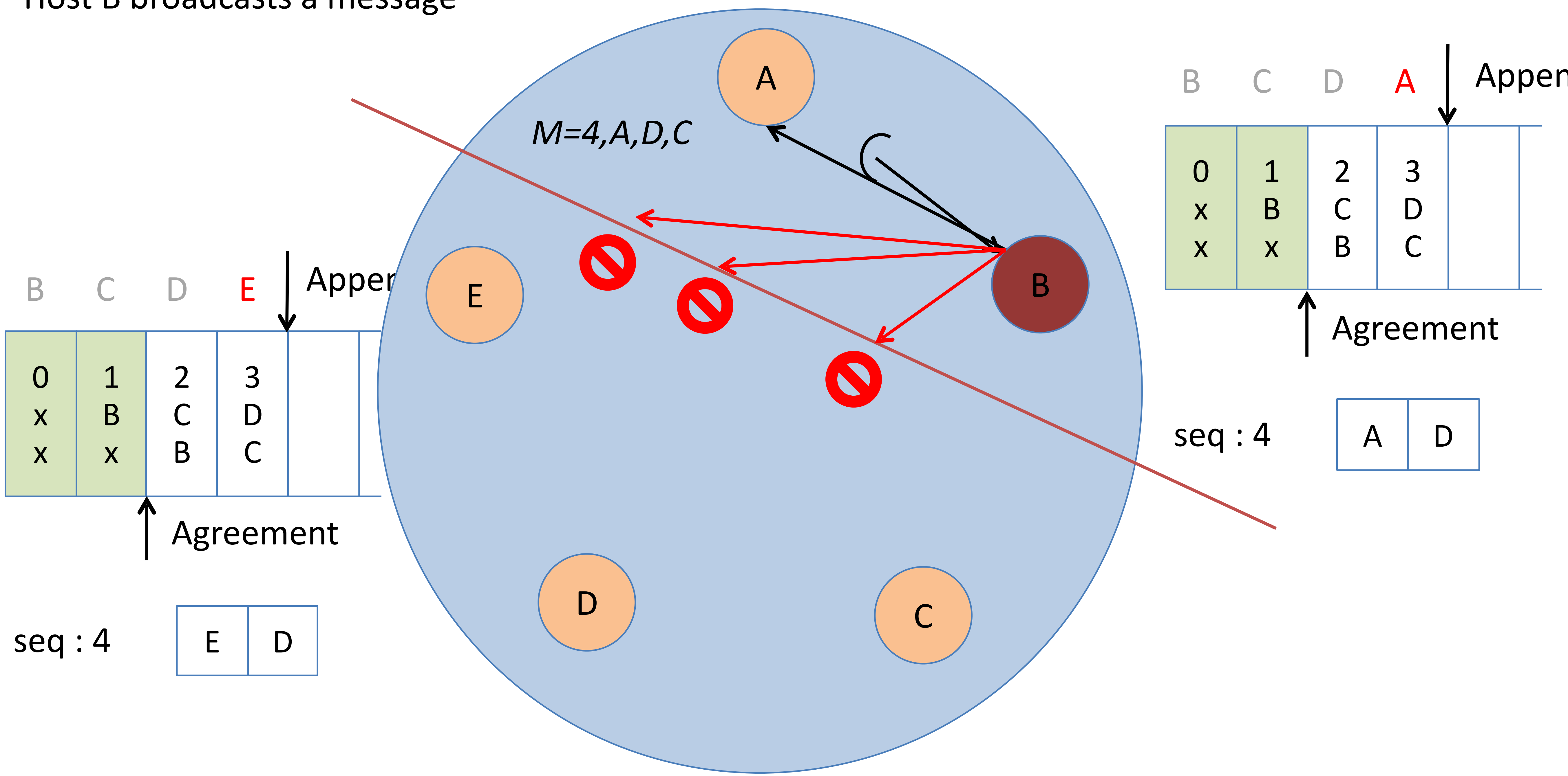
Handling Errors?

B now has the token



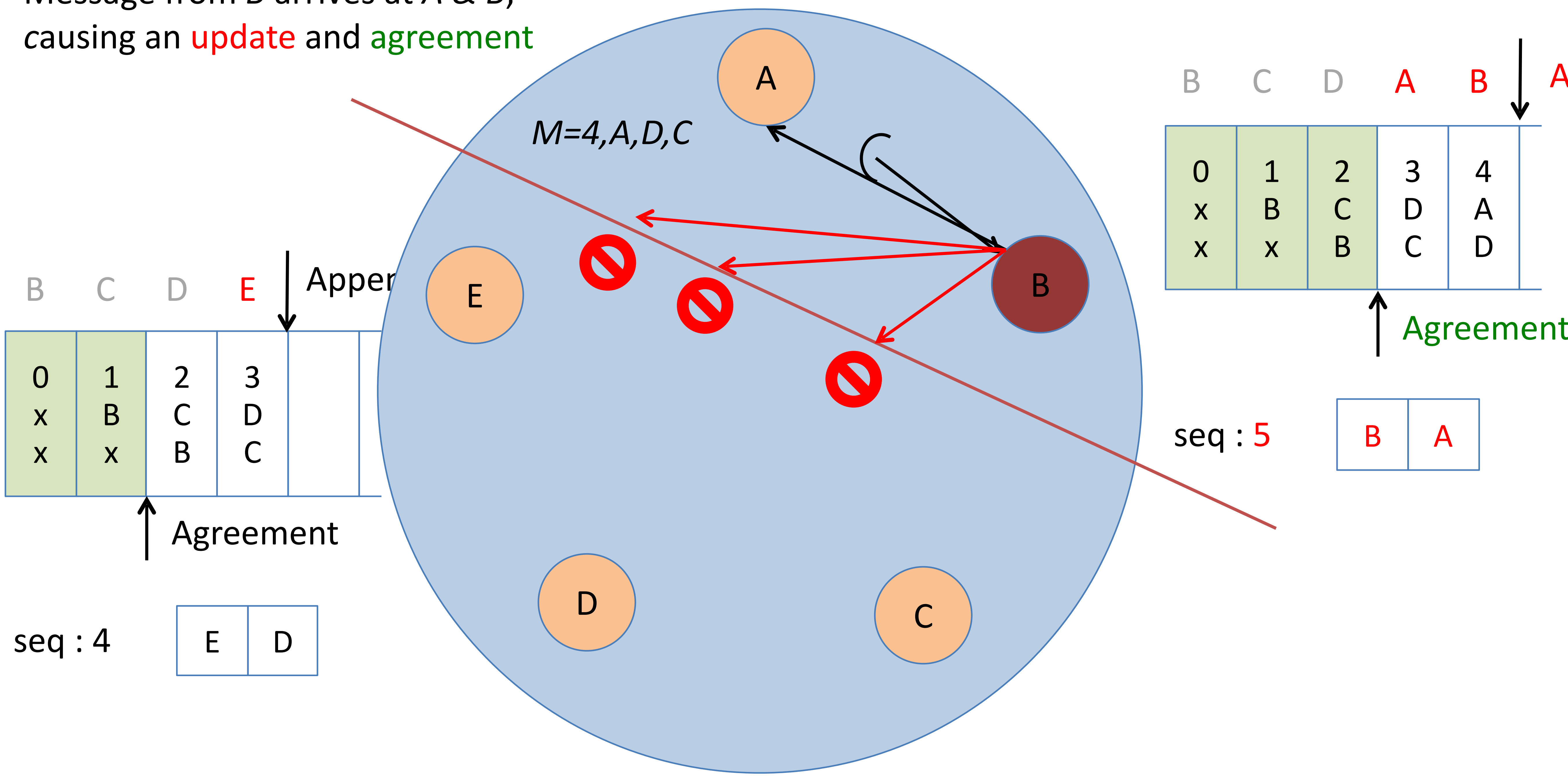
Handling Errors?

Host B broadcasts a message



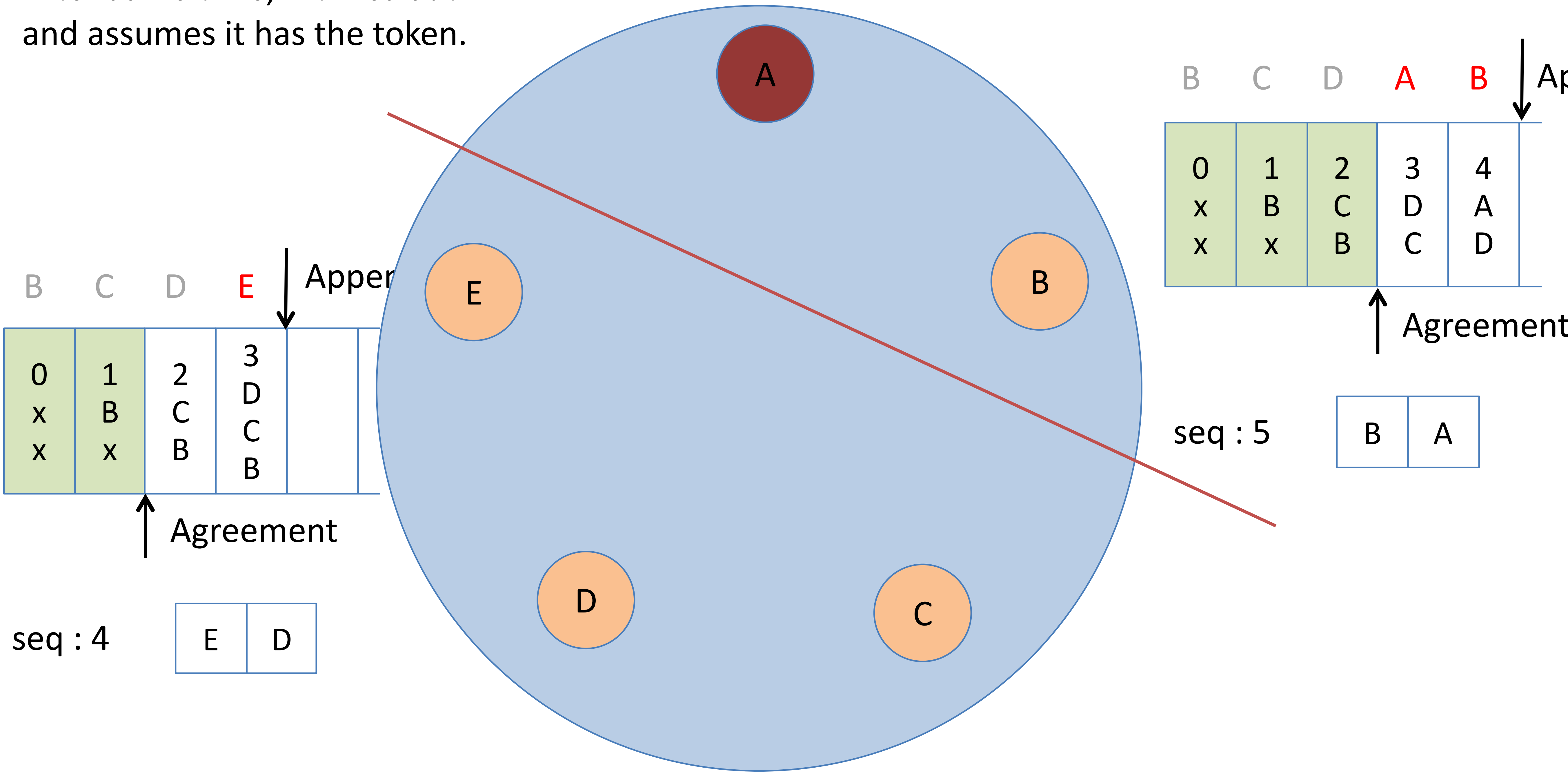
Handling Errors?

Message from *B* arrives at *A & B*, causing an update and agreement



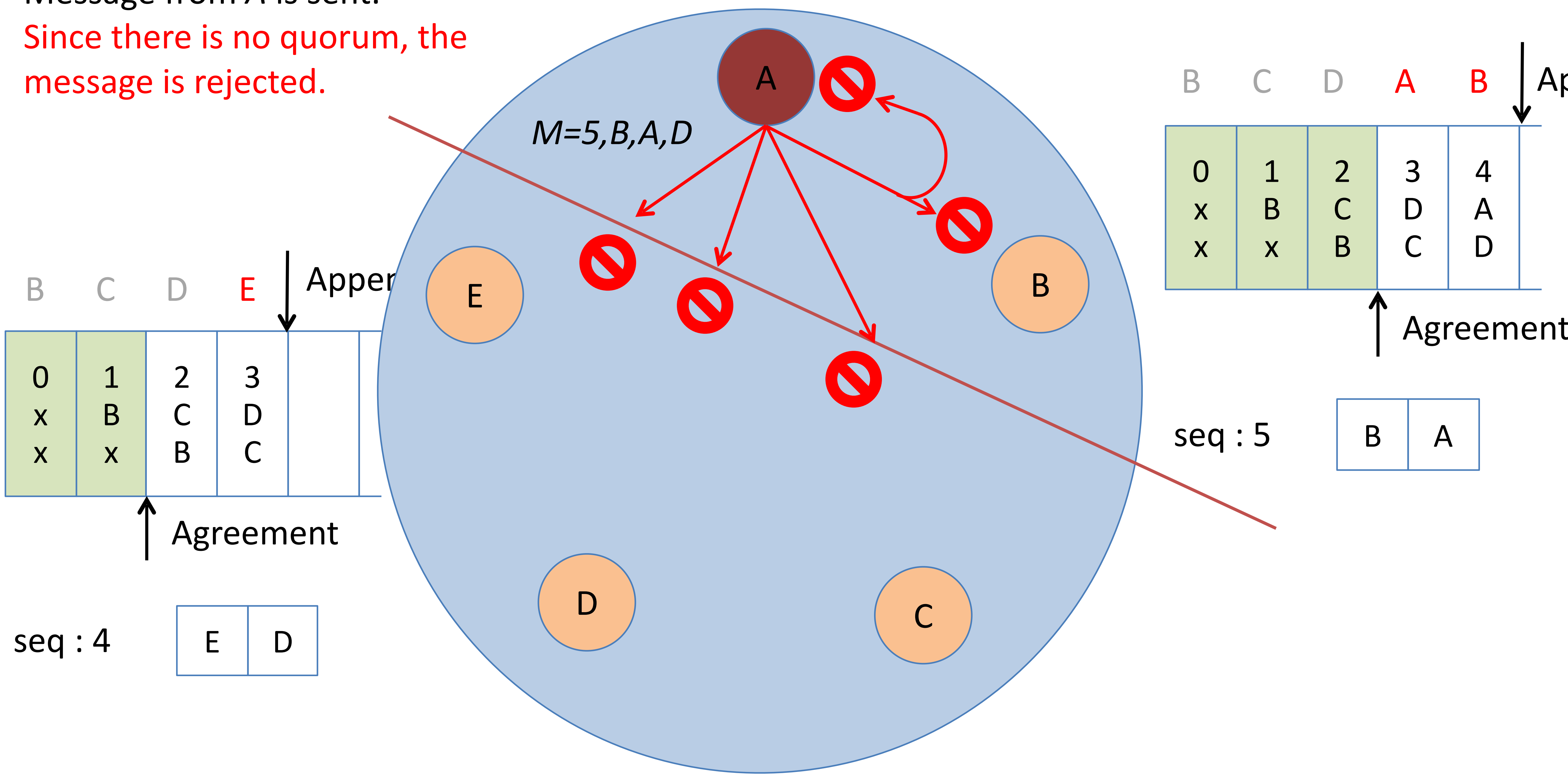
Handling Errors?

After some time, *A* times out and assumes it has the token.



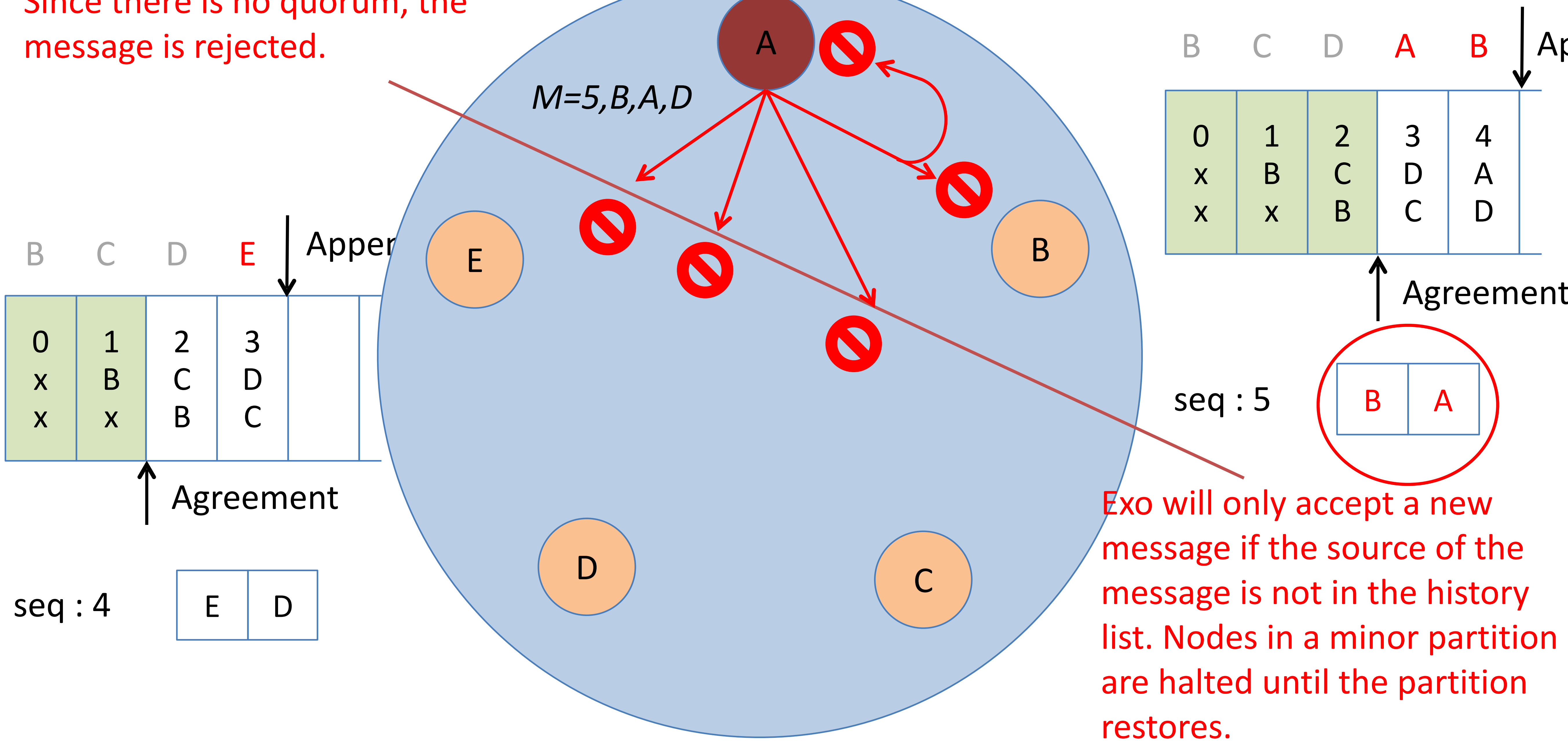
Handling Errors?

Message from *A* is sent.
Since there is no quorum, the message is rejected.



Handling Errors?

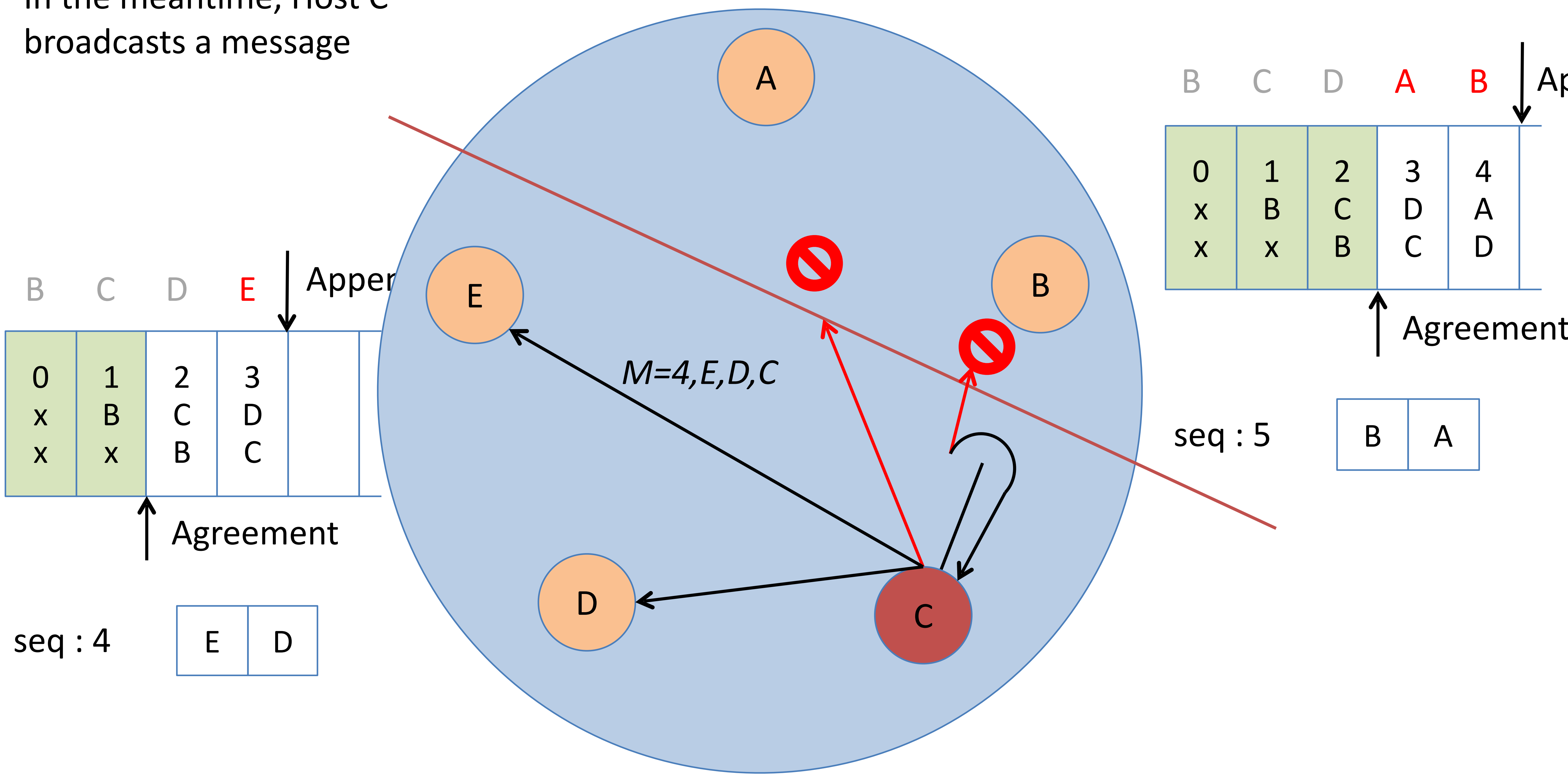
Message from *A* is sent.
Since there is no quorum, the message is rejected.



Exo will only accept a new message if the source of the message is not in the history list. Nodes in a minor partition are halted until the partition restores.

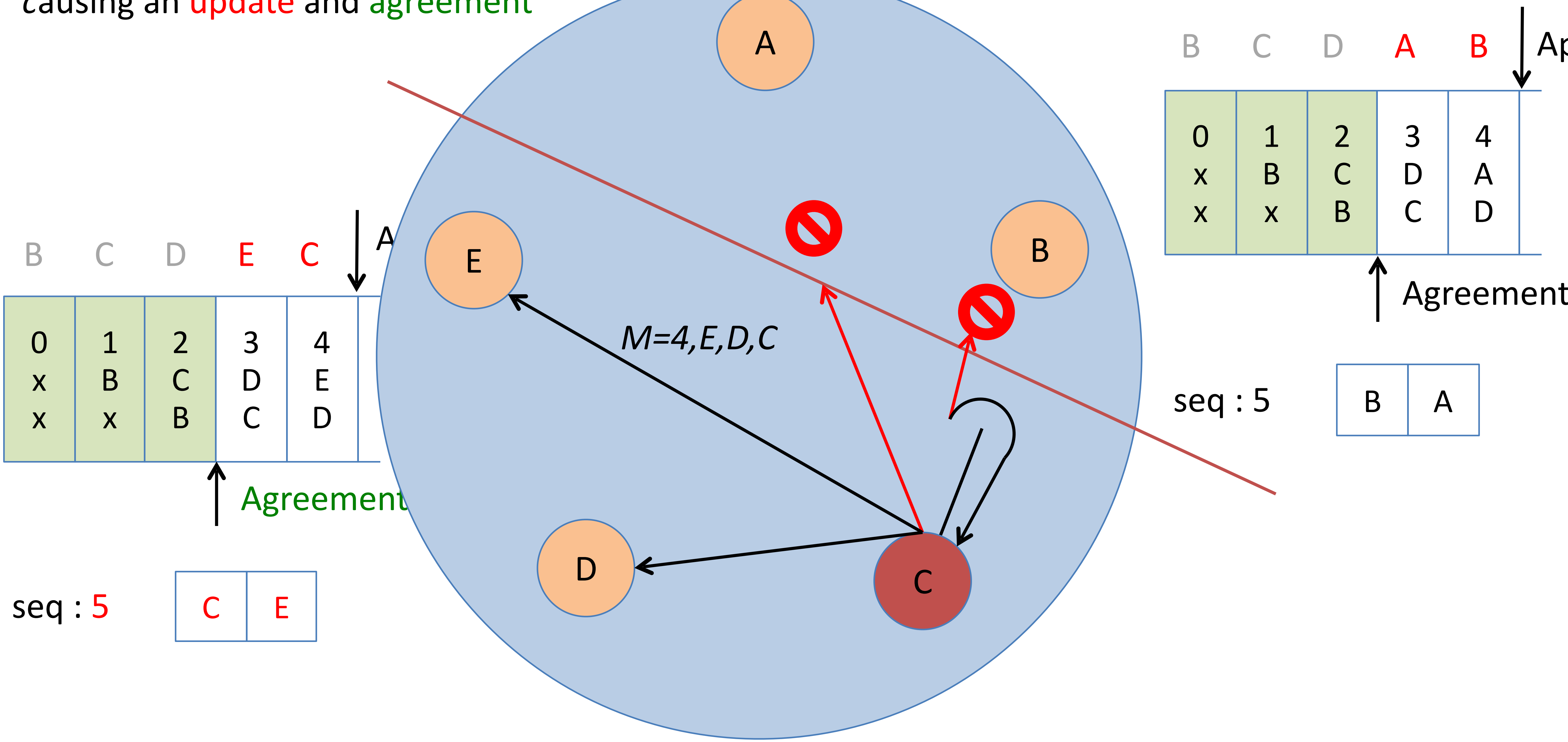
Handling Errors?

In the meantime, Host C broadcasts a message



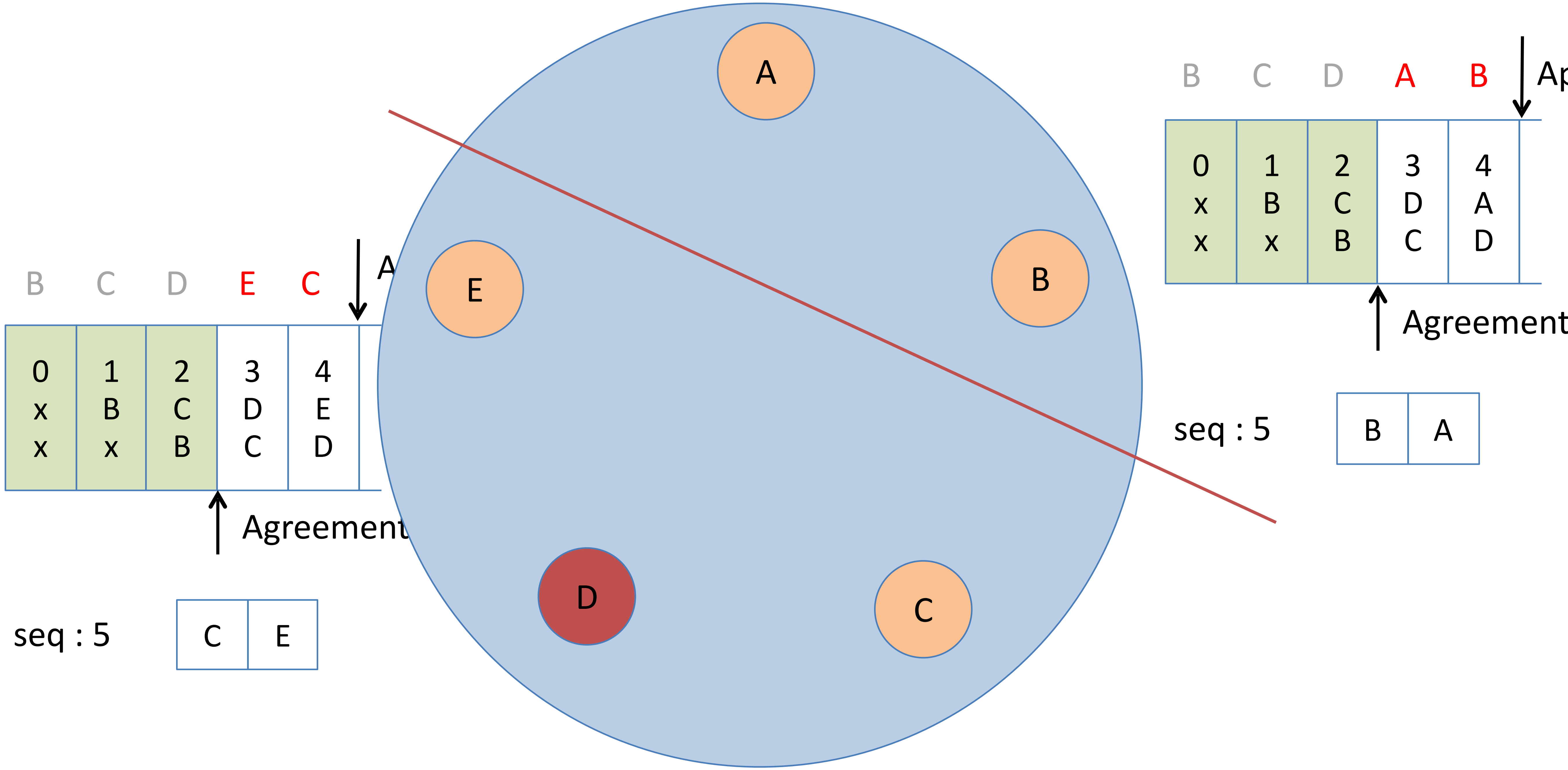
Handling Errors?

Message from *B* arrives at *A & B*, causing an update and agreement



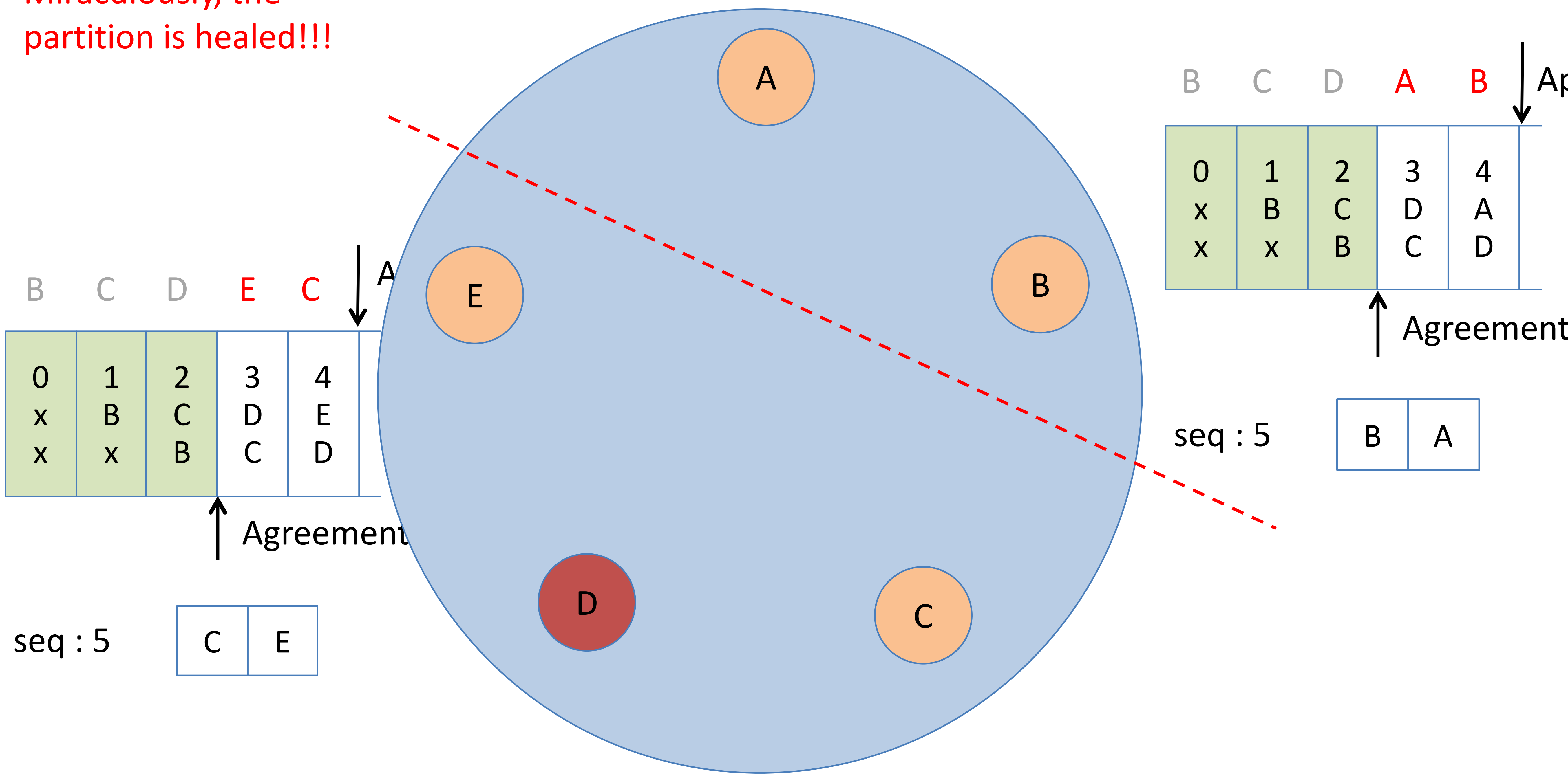
Handling Errors?

D now has the token



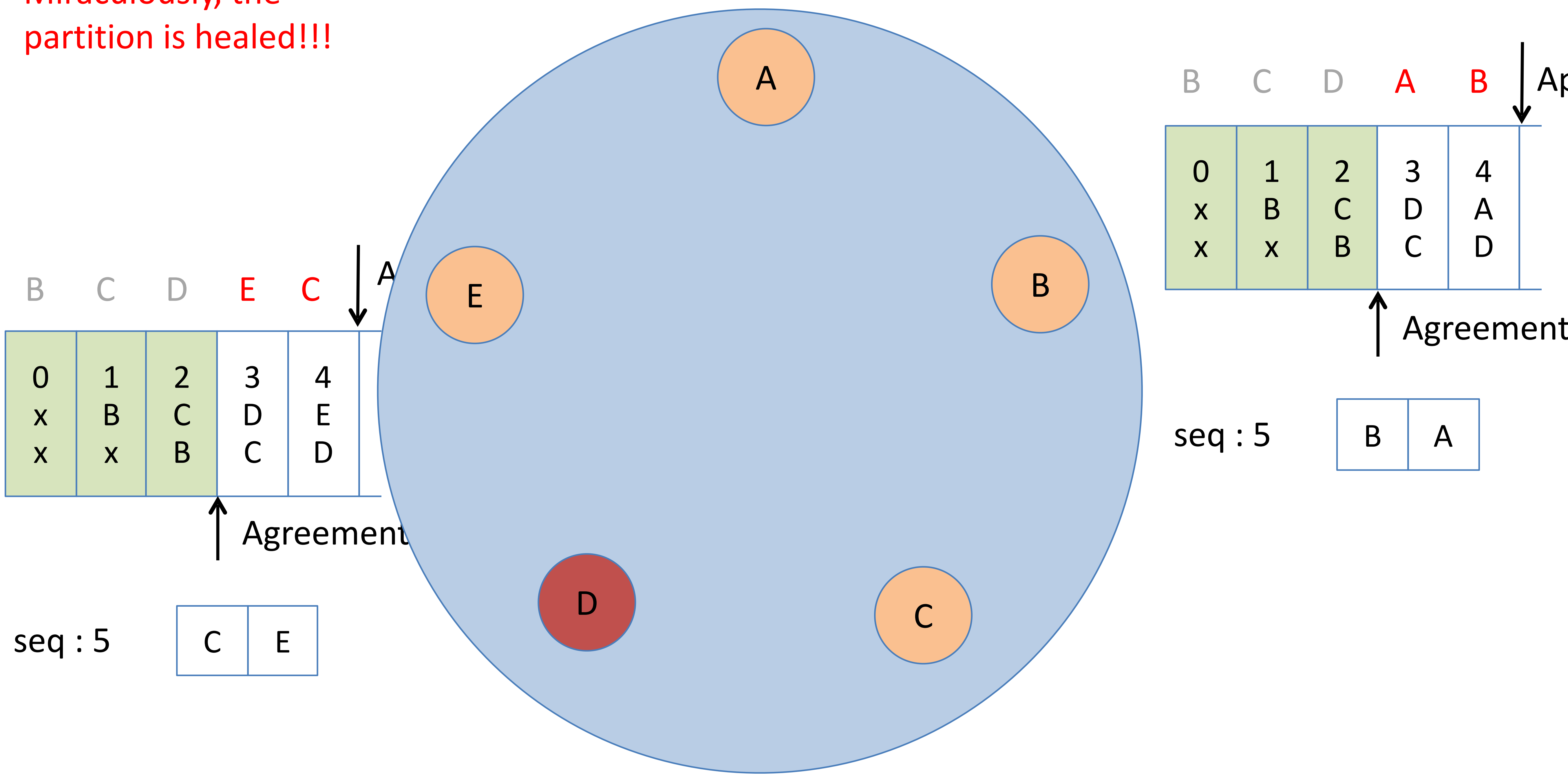
Handling Errors?

Miraculously, the partition is healed!!!



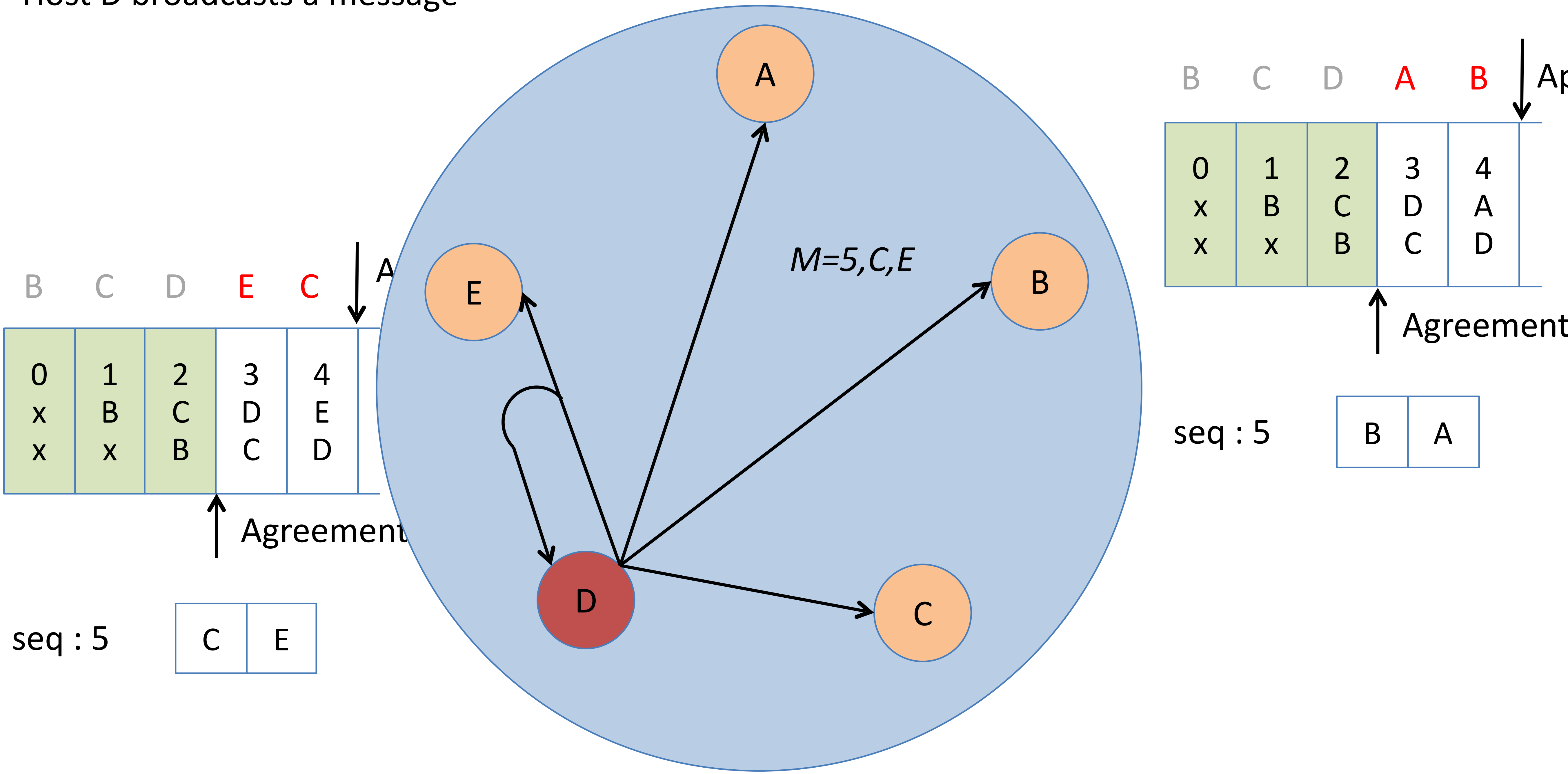
Handling Errors?

Miraculously, the partition is healed!!!



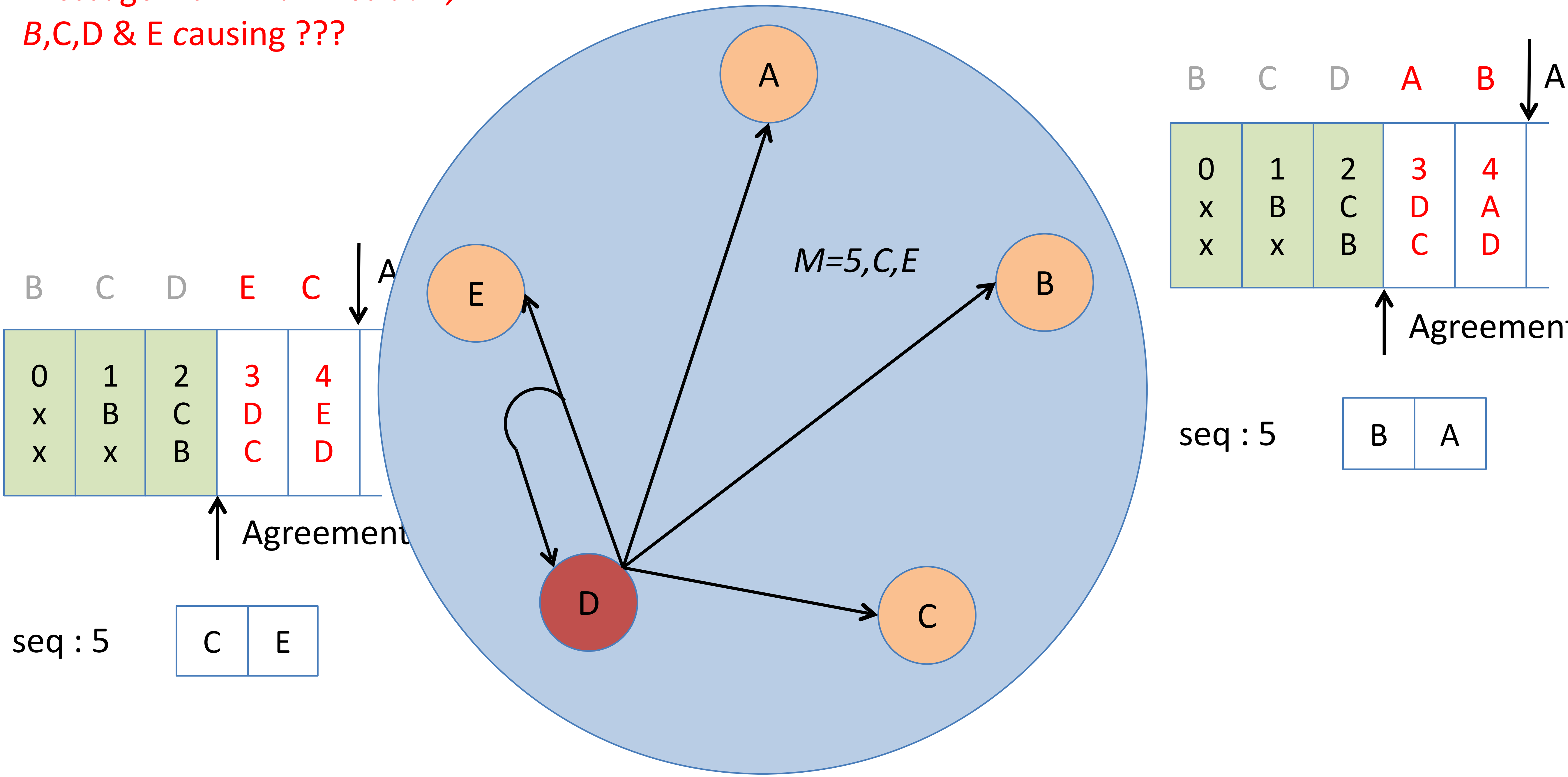
Handling Errors?

Host D broadcasts a message



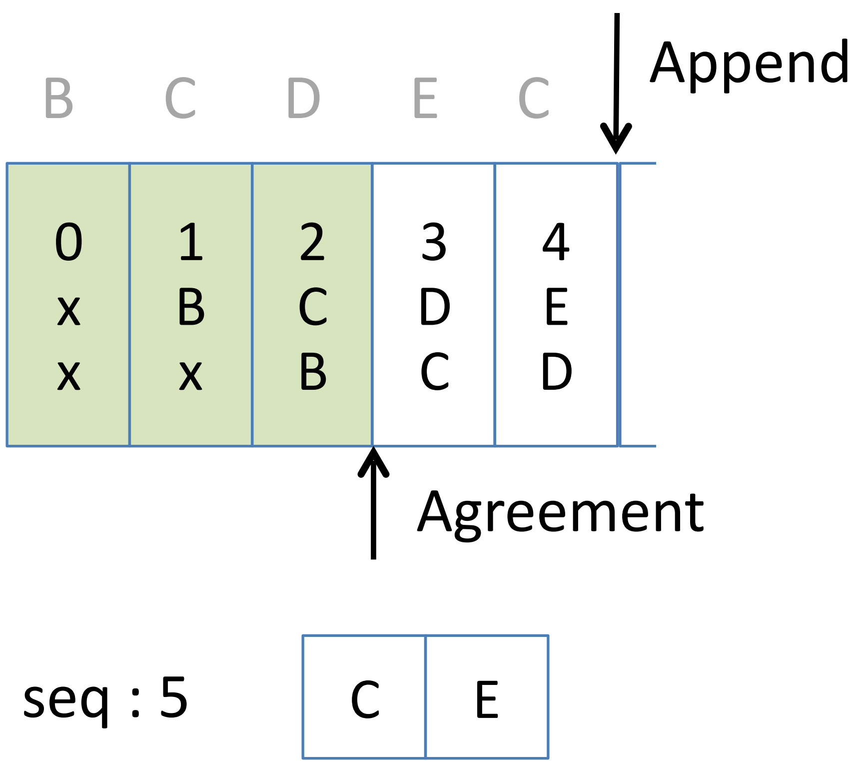
Handling Errors?

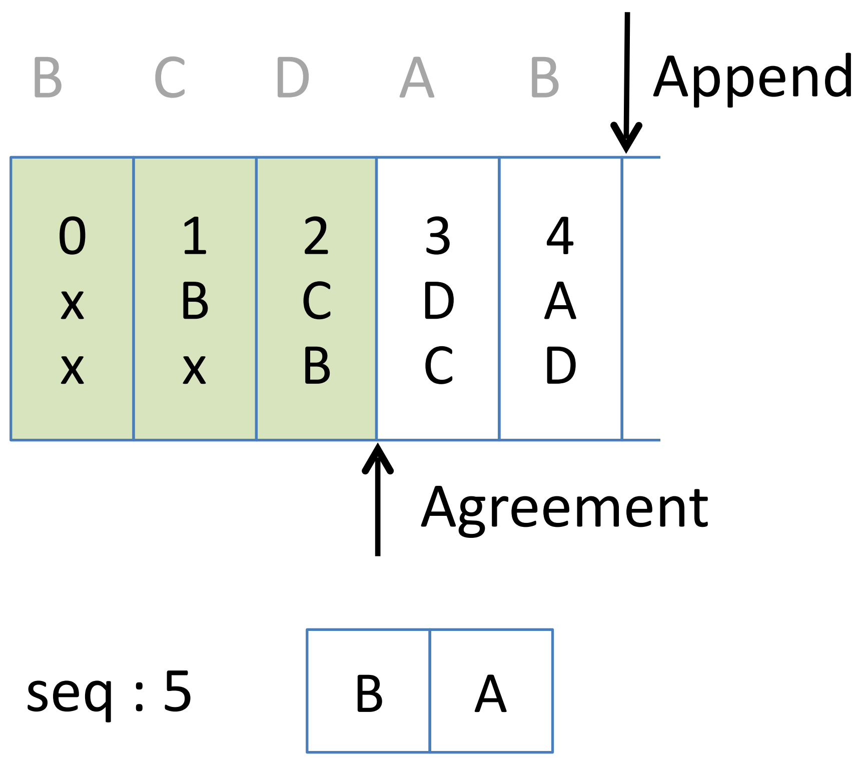
Message from *D* arrives at *A*, *B*,C,D & E *causing* ???



Handling Errors?

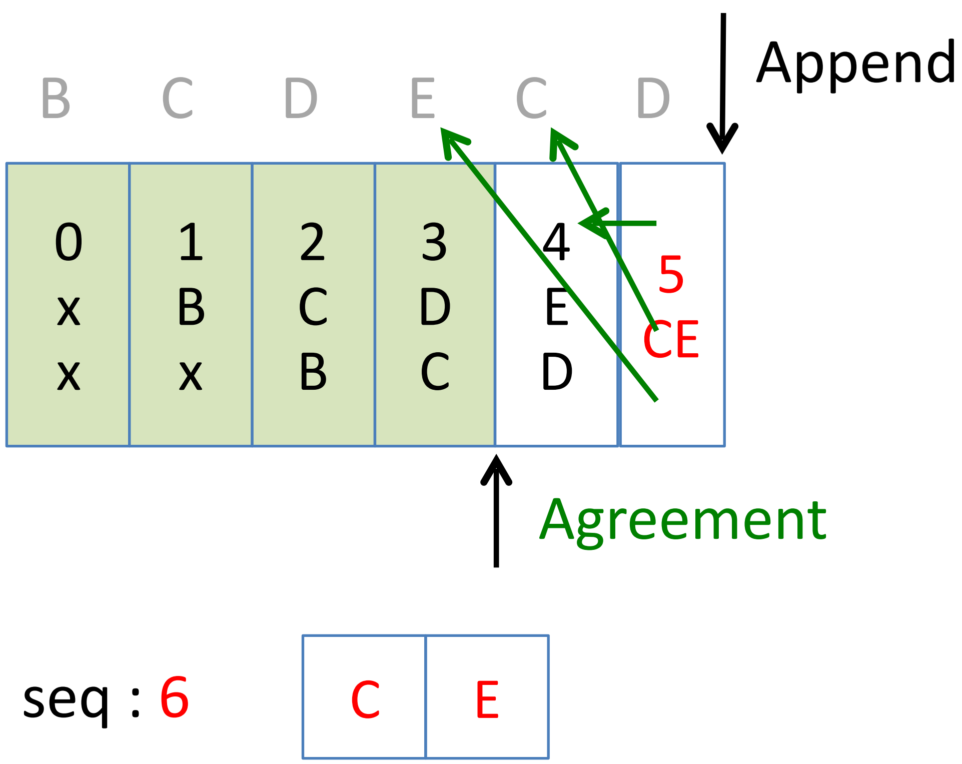
Message from *D* arrives at *A*,
B,C,D & E *causing* ???

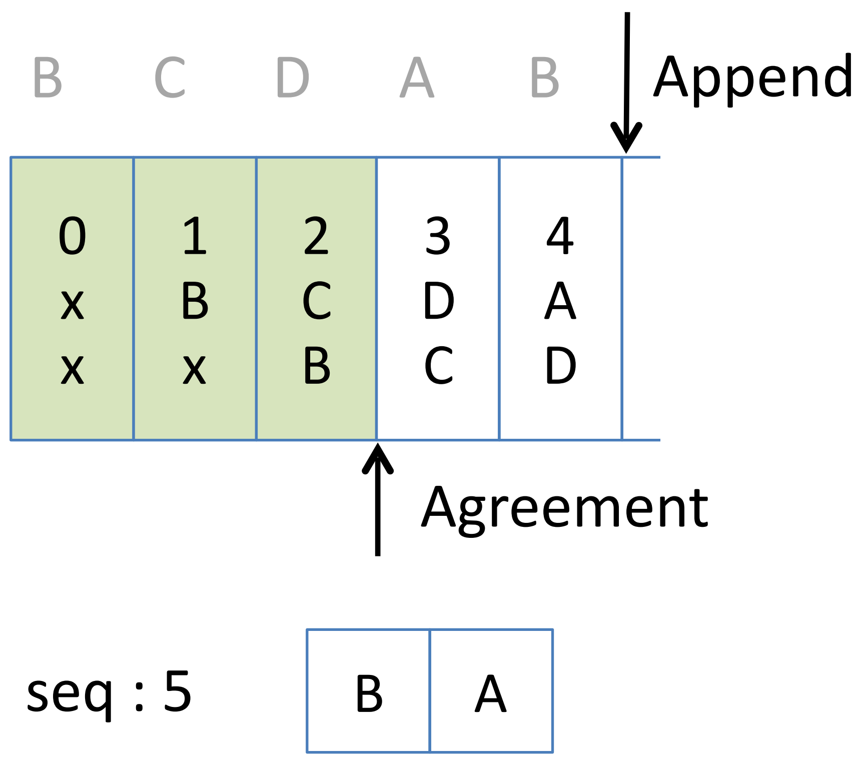




Handling Errors?

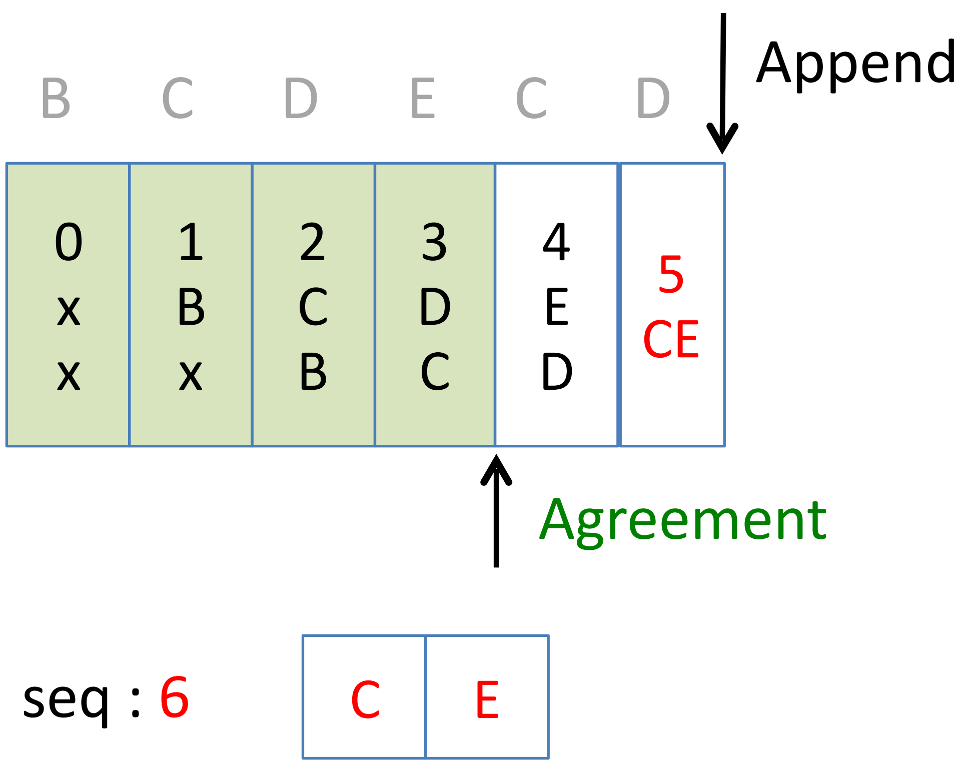
Message from *D* arrives at *A*, *B*,C,D & E *causing* ???

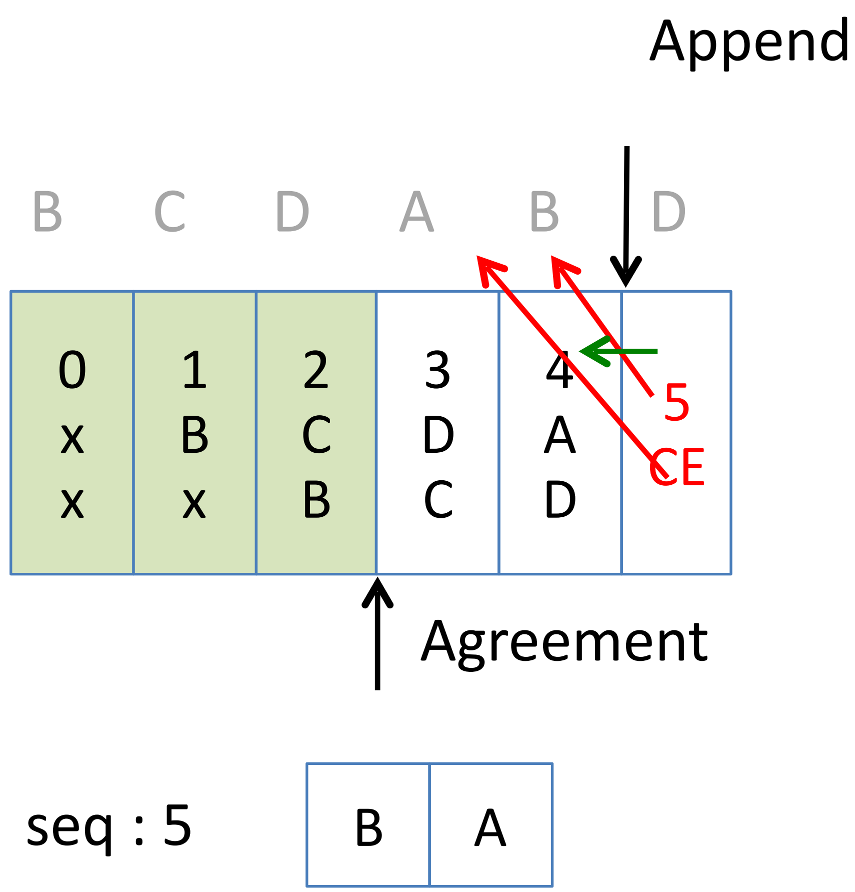




Handling Errors?

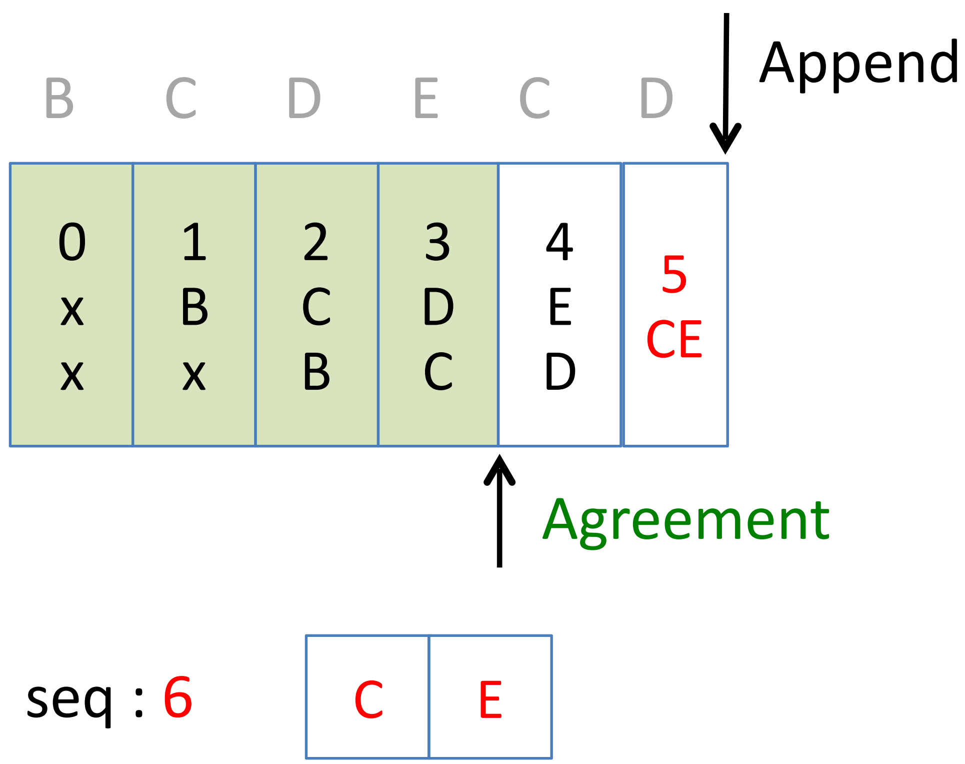
Message from *D* arrives at *A, B*,C,D & E *causing* ???

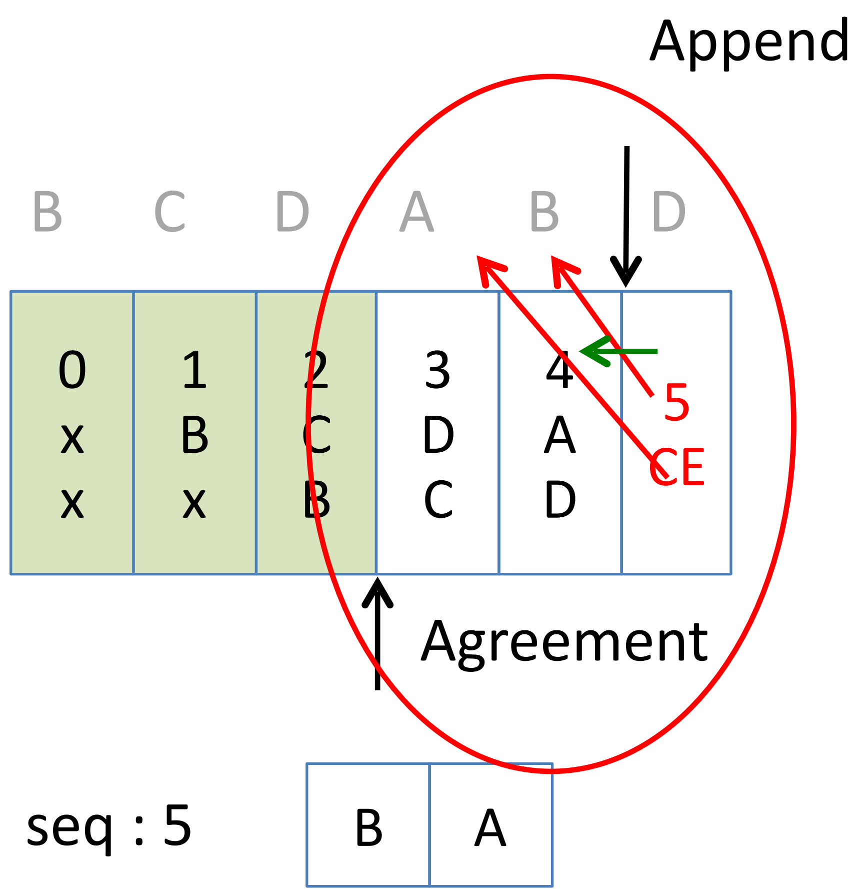




Handling Errors?

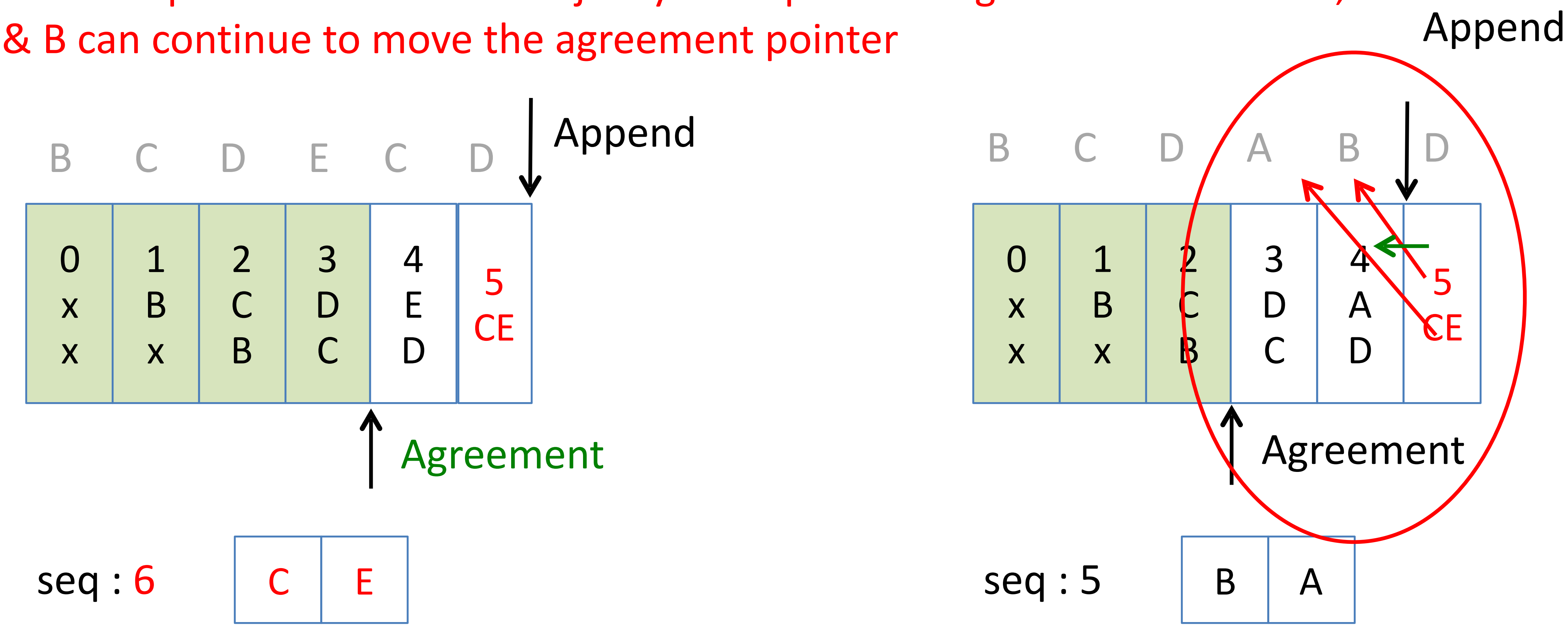
Message from *D* arrives at *A, B*,C,D & E *causing*
-update and agreement to C, D & E
-error detection for A & B





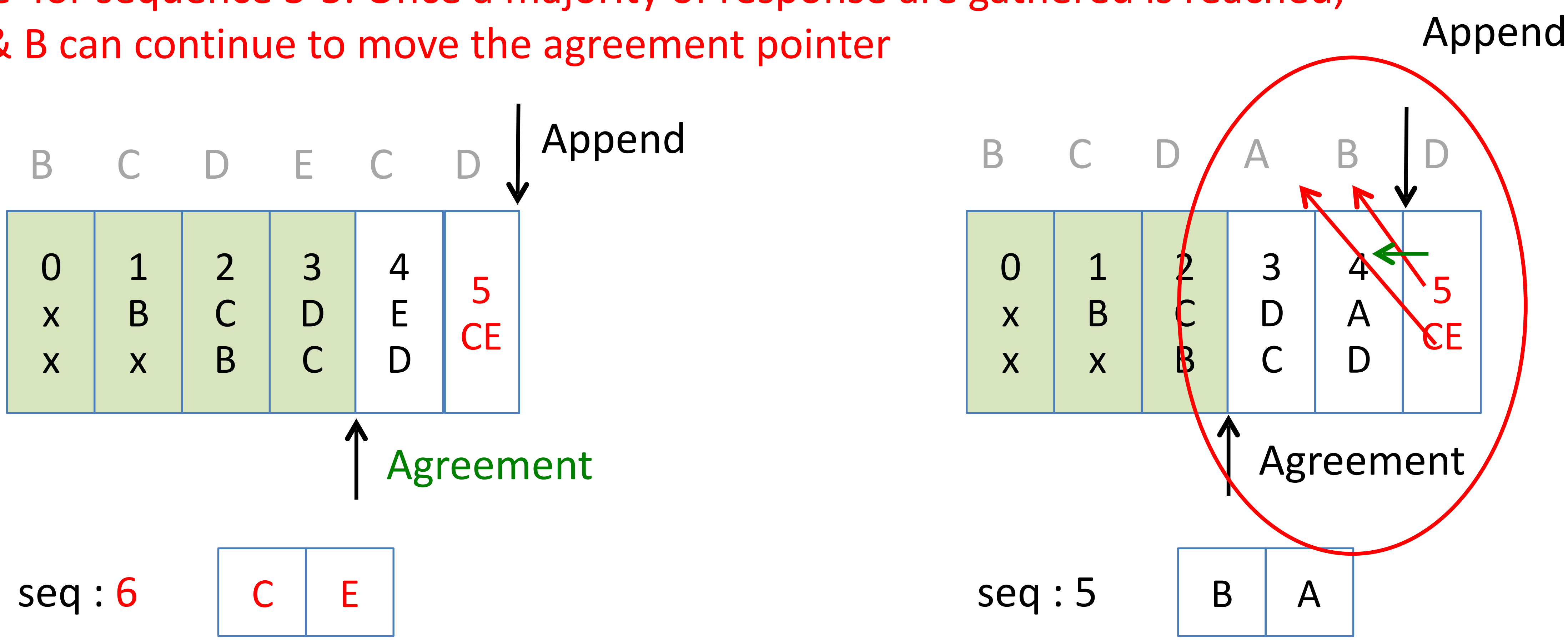
Handling Errors?

Hosts A &B must now use the out of band network to query C,D & E about their log state for sequence 3-5. Once a majority of response are gathered is reached, host A & B can continue to move the agreement pointer



Handling Errors?

Hosts A &B must now use the out of band network to query C,D & E about their log state for sequence 3-5. Once a majority of response are gathered is reached, host A & B can continue to move the agreement pointer



NB: New messages can be added to the log at the append pointer, but the agreement pointer cannot move until consensus has been reached