

# **Asynchronous data communication mechanisms**

Based on results of our  
Coherent and Comfort  
projects (EPSRC)



# Data communication

- Point to point: connecting two processes (SW or HW)



Data is a stream of items of a set type (e.g. large video frames).

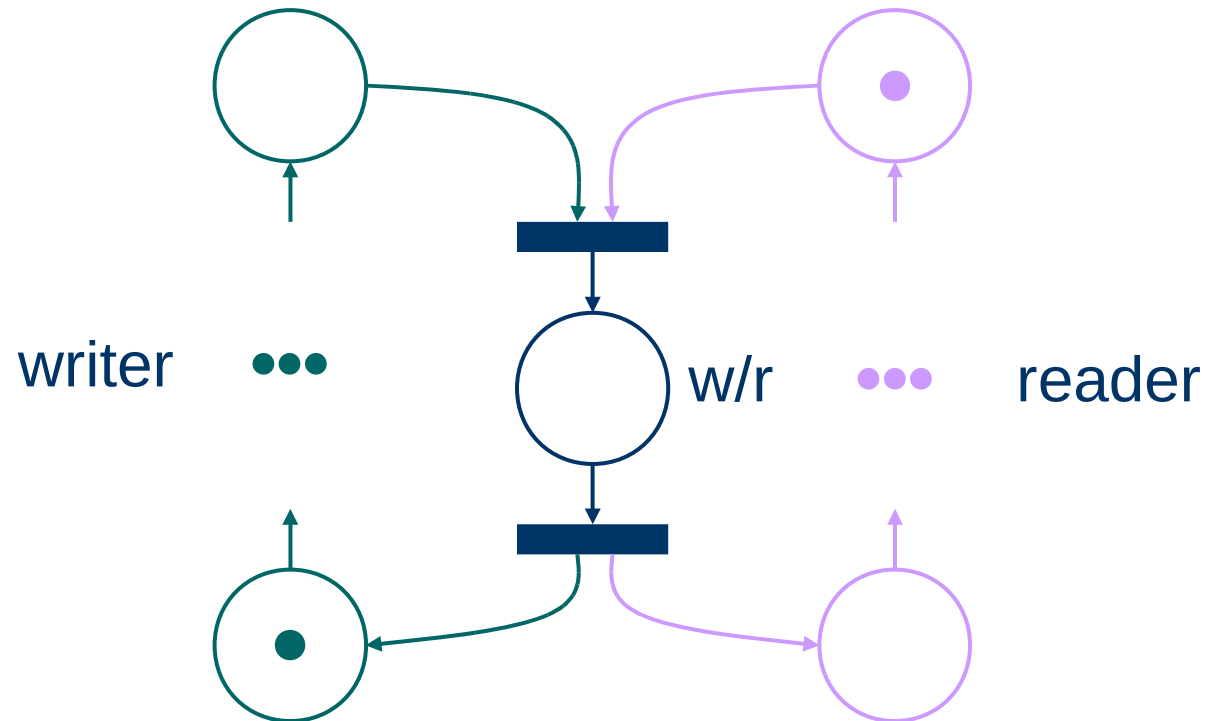
Writer and reader are cyclic processes.

Writer provides one item of data per cycle.

Reader uses one item per cycle.

# Data communication

- Traditional approach



# Data communication

- Traditional approach is synchronized
  - Either reader or writer must wait for the other side during the transfer of **every** data item
  - Not ideal for many concurrent systems (esp. embedded, real-time, and low power systems)

# Asynchrony in concurrent systems

- Inevitable
  - High degrees of integration mean that distribution of global clocks becomes impractical, so even on the same chip there will be asynchrony
  - Distributed systems naturally have local clocks. Synchronizing such clocks can be problematic.
  - Self-timed systems (esp. useful for low power) have no regularly pulsing clocks.

# Asynchrony in concurrent systems

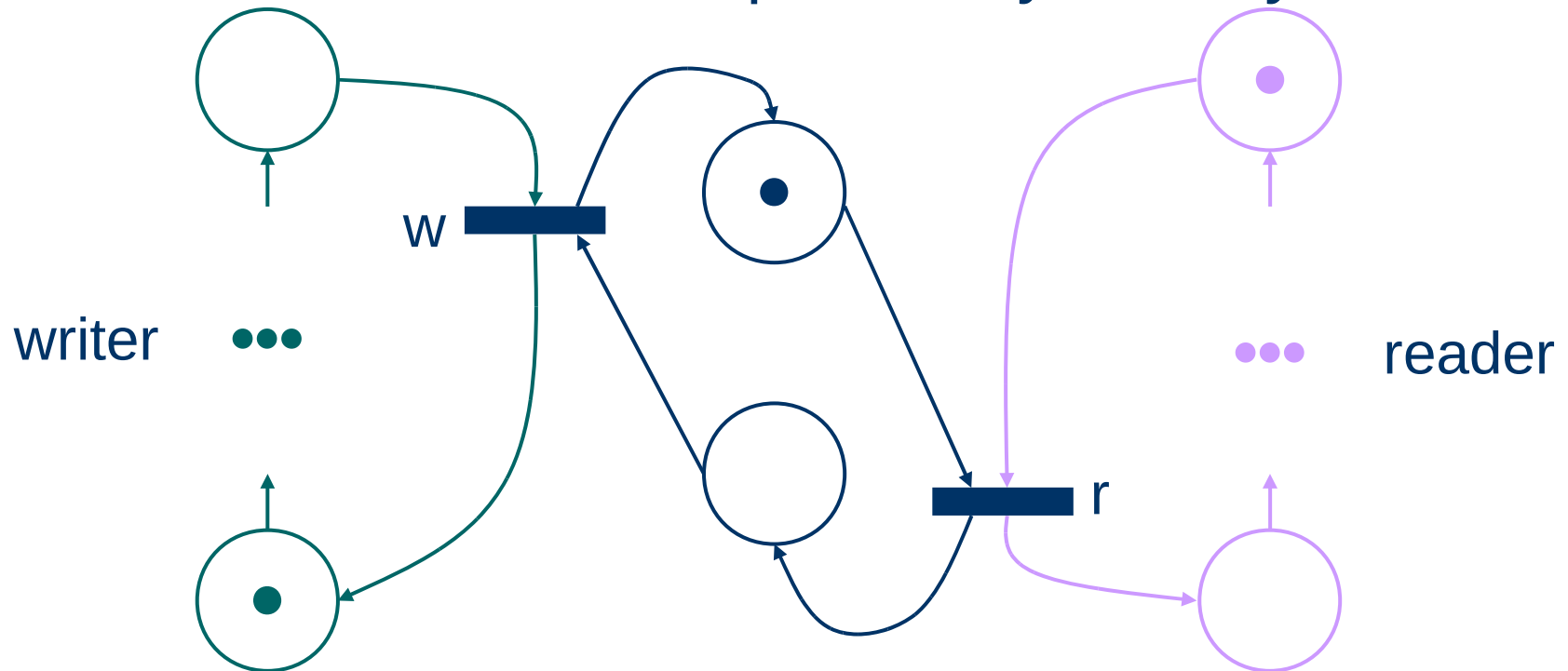
- Real-time elements
  - Most obvious characteristic is timing predictability
  - Such processes thus should not be delayed by outside influences, e.g. data communications with another process
  - Data communications should therefore follow “not obliged to wait for data” protocols

# Asynchrony in concurrent systems

- Low power elements
  - In such things as battery powered remote sensors, timing for data items can depend on two factors
  - Some sensors only produce data when they detect changes in the input signal
  - Other sensors may produce data when their communication partners request data from them
  - Each requires a different approach to data communication synchrony

# Asynchronous data communication

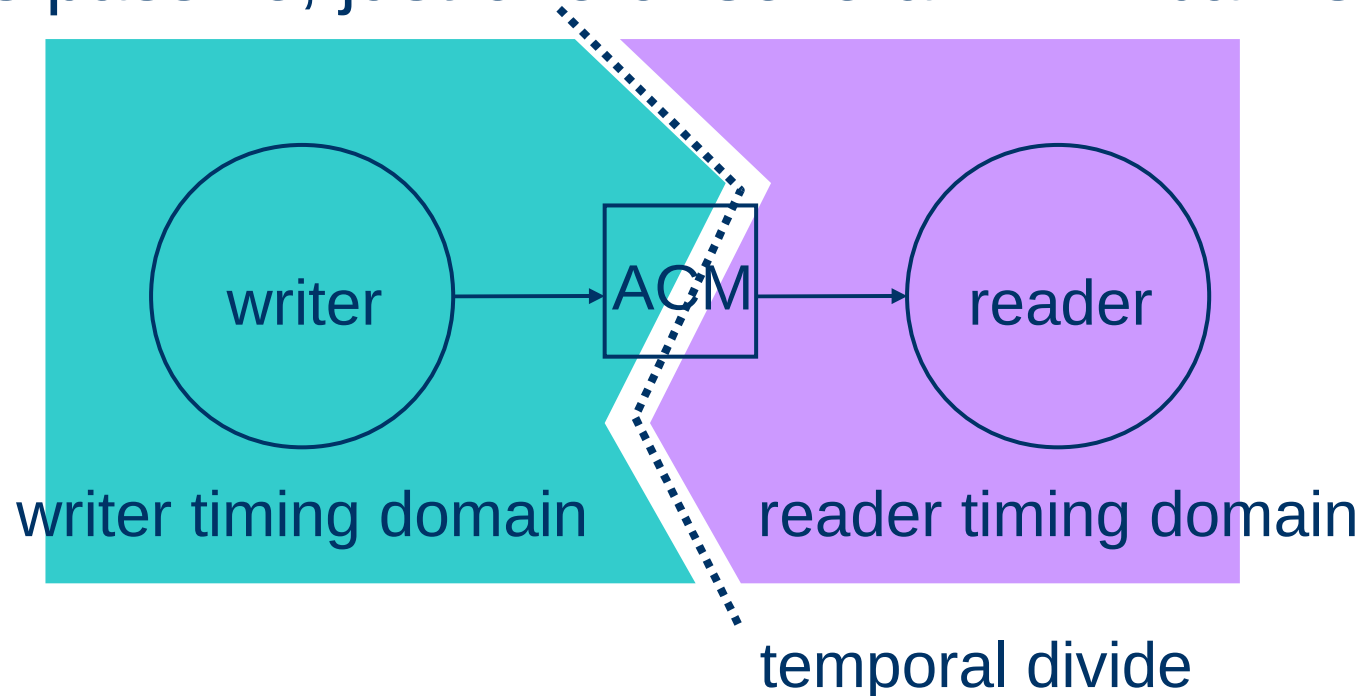
- A buffer increases scope for asynchrony





# ACMs

- An ACM is implemented with shared memory
- It is passive, just one or several RAM banks

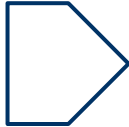





# ACMs

- An ACM can use the following protocols
  - Writer may be required to wait for reader
  - Writer may not be required to wait for reader
  - Reader may be required to wait for writer
  - Reader may not be required to wait for writer
- Qualitative asynchrony specifications naturally divides all ACMs into four types

# ACM Taxonomy

- The four ACM types

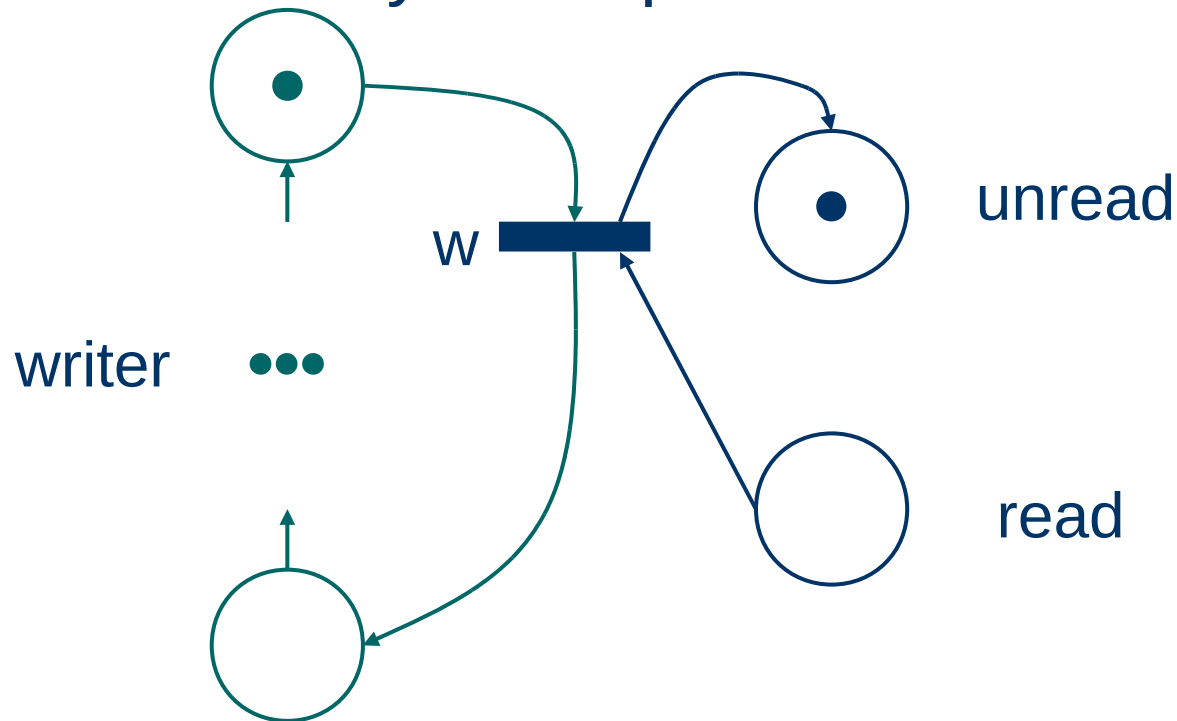
	Reader may wait	Reader may not wait
Writer may not wait	 OW/NRR	 OW/RR
Writer may wait	 NOW/NRR	 NOW/RR

# ACMs

- Consequences of waiting/no waiting
  - Overwriting – new item superseding a previous one when no item in an ACM has been read
  - Rereading – reading previously read item when no newer one is available
- Dangers: what may happen (properties)
  - half-old half-new data – coherence property
  - reading the obsolete data – freshness property
  - violation of order – sequencing property
  - waiting where no waiting is specified

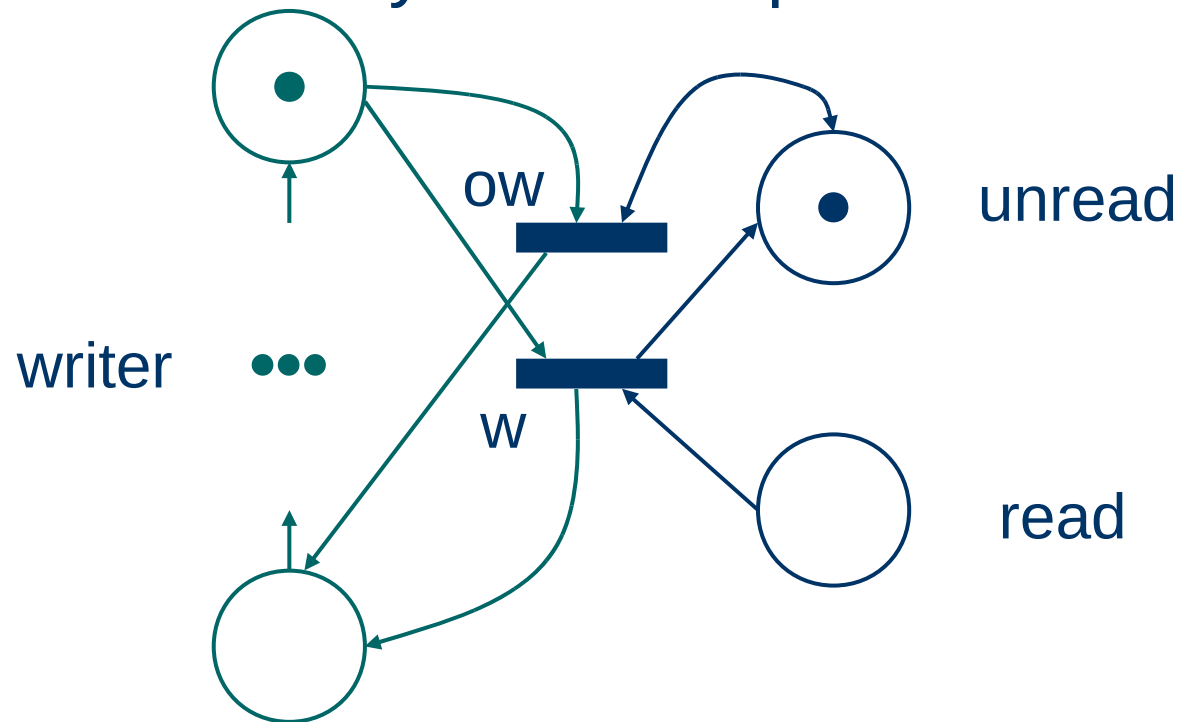
# ACMs: blocking write

- Writer may be required to wait



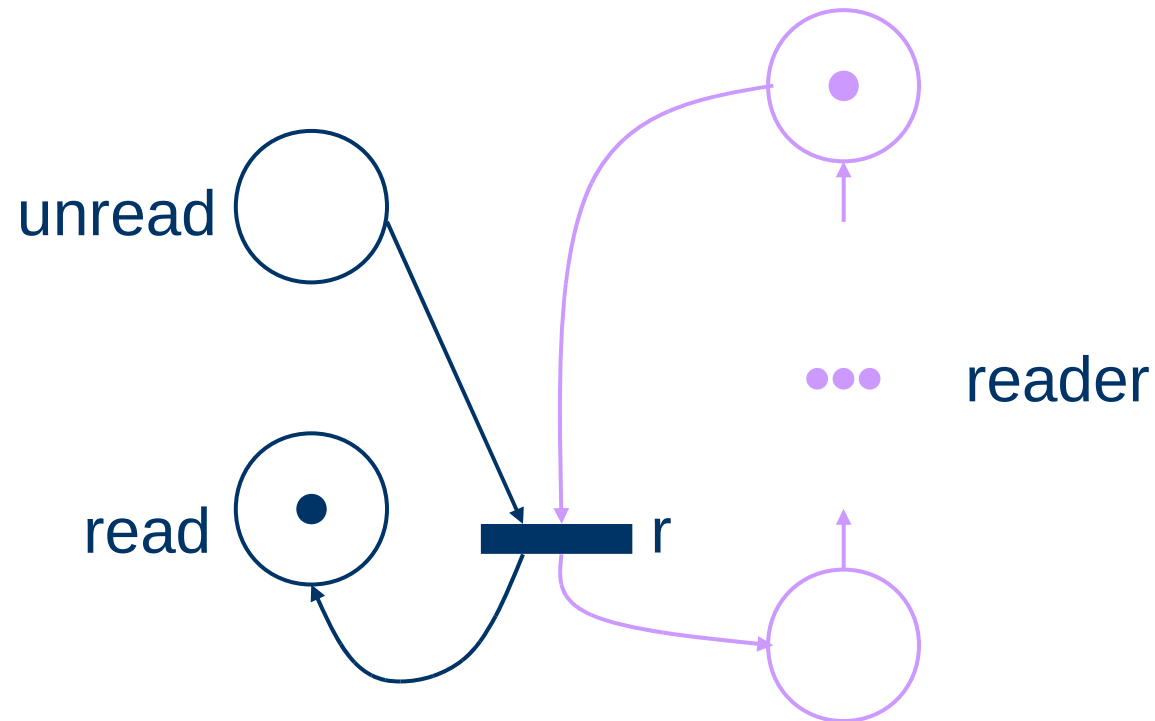
# ACMs: non-blocking write

- Writer may not be required to wait



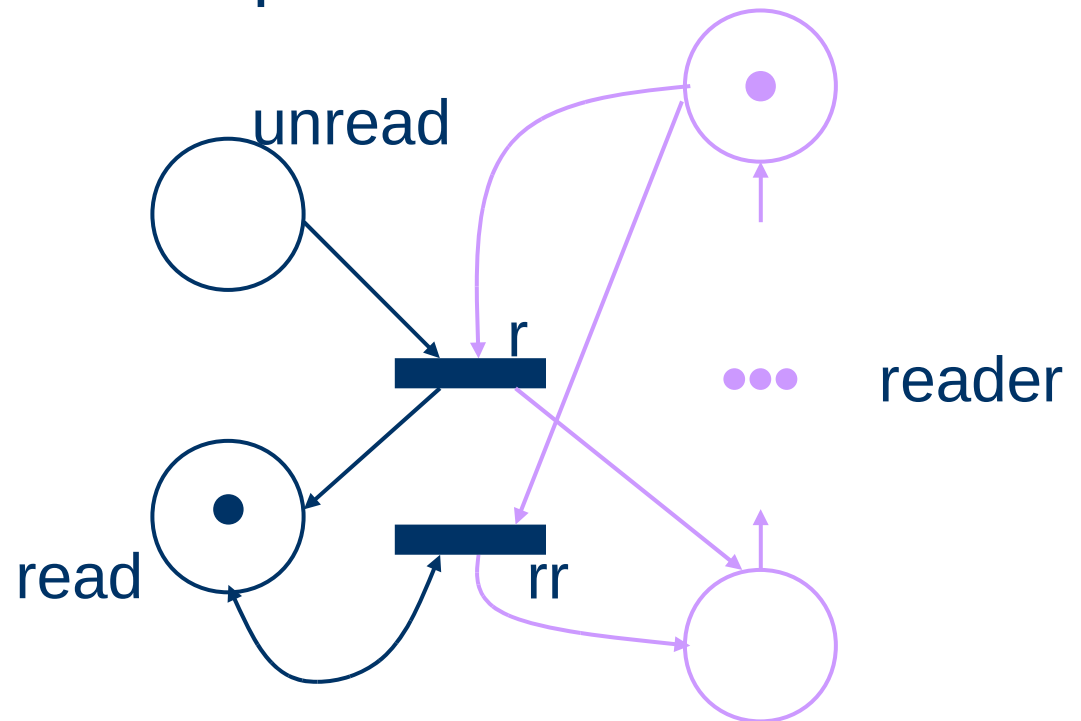
# ACMs: blocking read

- Reader may be required to wait



© 2015 Pearson Education, Inc. or its affiliate(s). All rights reserved.

- Reader may not be required to wait

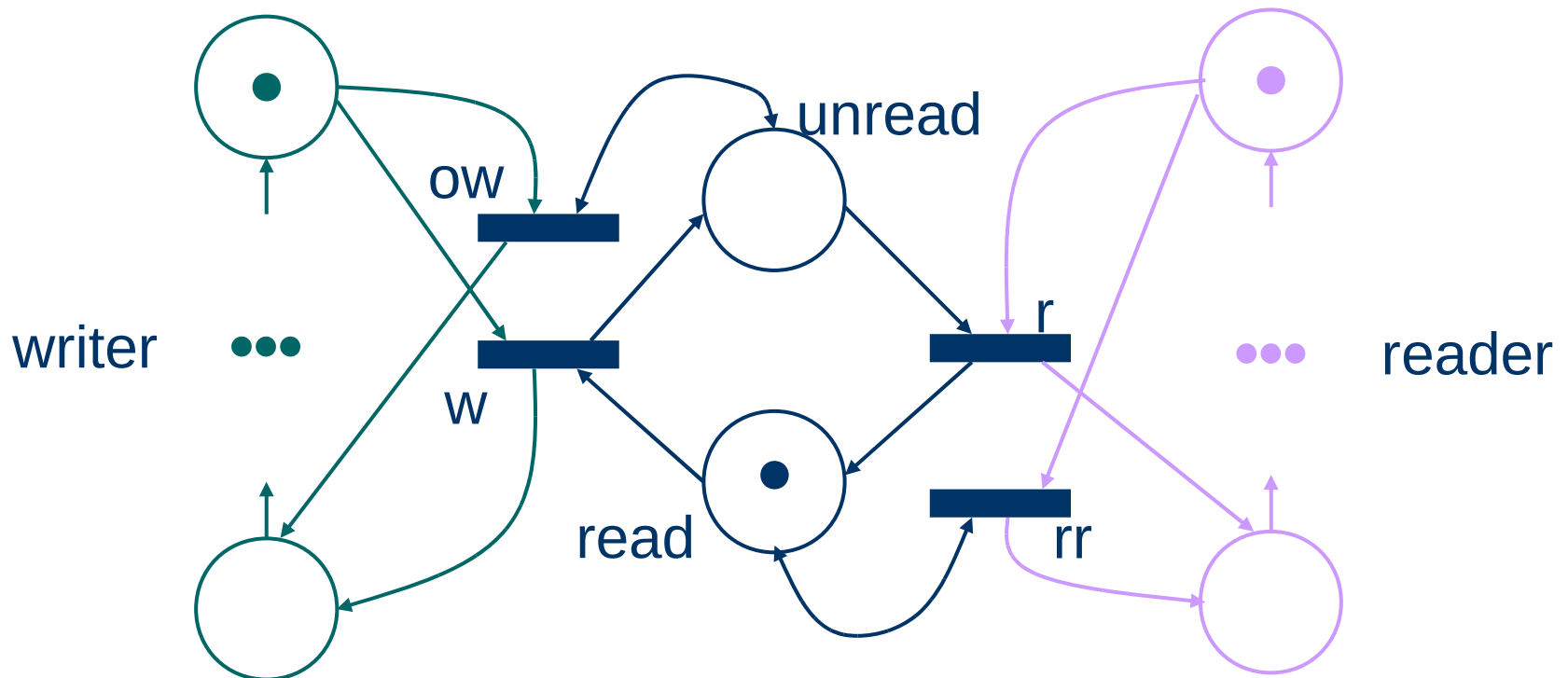




# ACMs: “pool” type



- No wait on either side



# ACMs: “pool” type history



- No wait on either side
  - “Atomic register” – L. Lamport
  - “Pool” – Mascot (e.g. H. Simpson, PhD thesis of Fei Xia and Ian Clark)
  - Connecting together two independently timed processes (e.g. real-time processes)

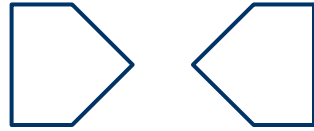
Examples (reference data, tower clock)

# ACMs: FIFO type



- May be required to wait on both sides
  - Traditional buffer in computers
  - “Channel” – Mascot (e.g. H. Simpson)
  - Full data continuity (no OW/RR)
  - Message data
  - Increasing the size of the buffer increases quantitative asynchrony, at the expense of latency (true for all ACMs)
  - Anti-shock CD player example

# ACMs



- May be required to wait on one side
  - Connecting self-motivated timing with reactive timing
  - Signal or command data
  - One side real-time the other low power, etc.
  - Propose examples!

# Virtual experiment

- Real-time game
- Delayed display
- Choosing the ACM type
- ACM implementation
  - Concept
  - Number of slots
  - Control variables
  - Algorithm

# ACMs

- Further reading

- <http://www.async.org.uk/comfort>
- <http://www.async.org.uk/coherent>

Ian G. Clark, Fei Xia, Alex V. Yakovlev, Delong Shang.  
"Data Communication Mechanisms for Systems with  
Heterogeneous Timing". Invited paper at the 2003  
Heterogeneous Computer Systems Workshop,  
University of South Australia, Adelaide. 27th February  
2003

<http://www.staff.ncl.ac.uk/i.g.clark/publications/AUS-workshop-2003.pdf>