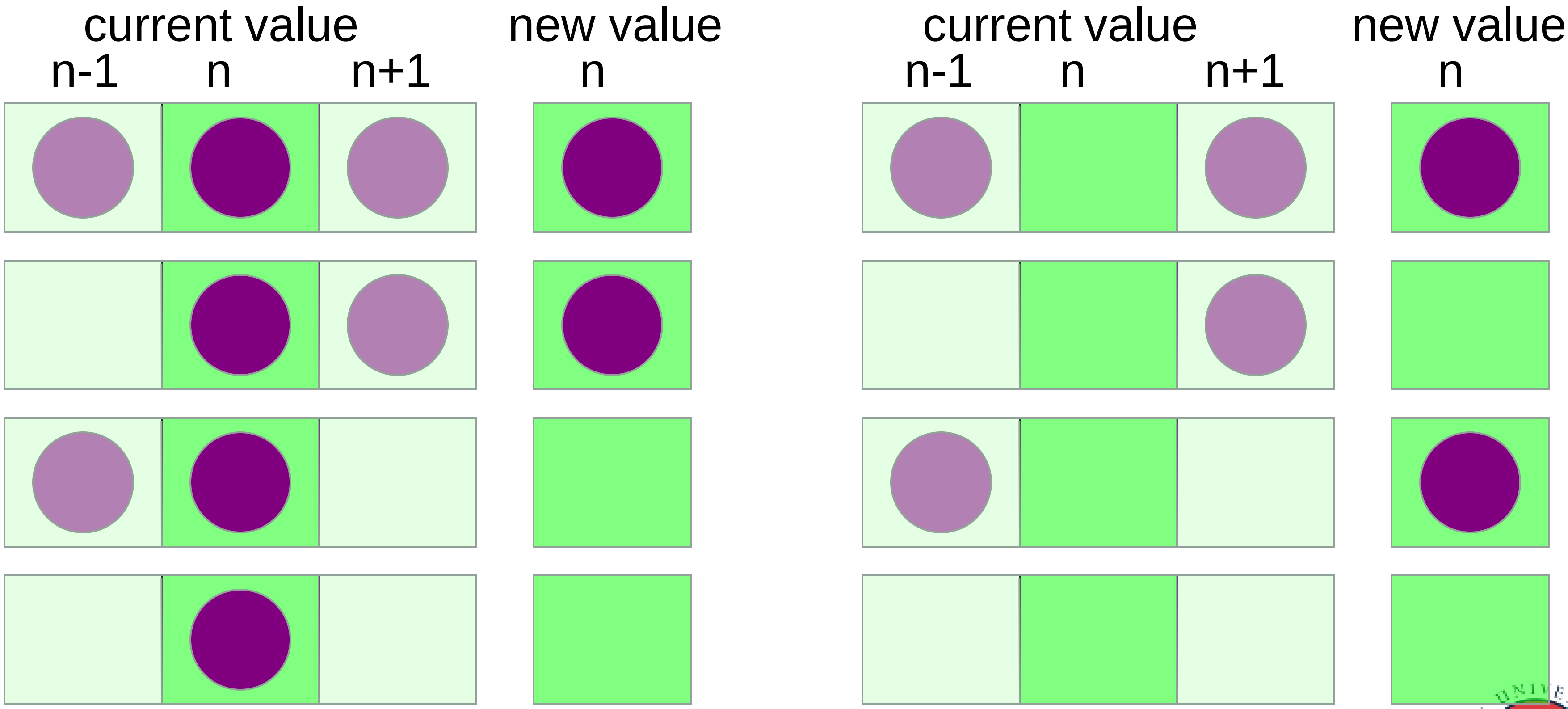
Message-Passing Programming

Cellular Automaton Exercise

Reusing this material

Traffic simulation

- Update rules depend on:
  - state of cell
  - state of nearest neighbours in both directions



State Table

- If $R^t(i) = 0$, then $R^{t+1}(i)$ is given by:

	$R^t(i\text{-}1) = 0$	$R^t(i\ \text{-}1) = 1$
$R^t(i\text{+}1) = 0$	0	1
$R^t(i\text{+}1) = 1$	0	1

- If $R^t(i) = 1$, then $R^{t+1}(i)$ is given by:

	$R^t(i\text{-}1) = 0$	$R^t(i\ \text{-}1) = 1$
$R^t(i\text{+}1) = 0$	0	0
$R^t(i\text{+}1) = 1$	1	1

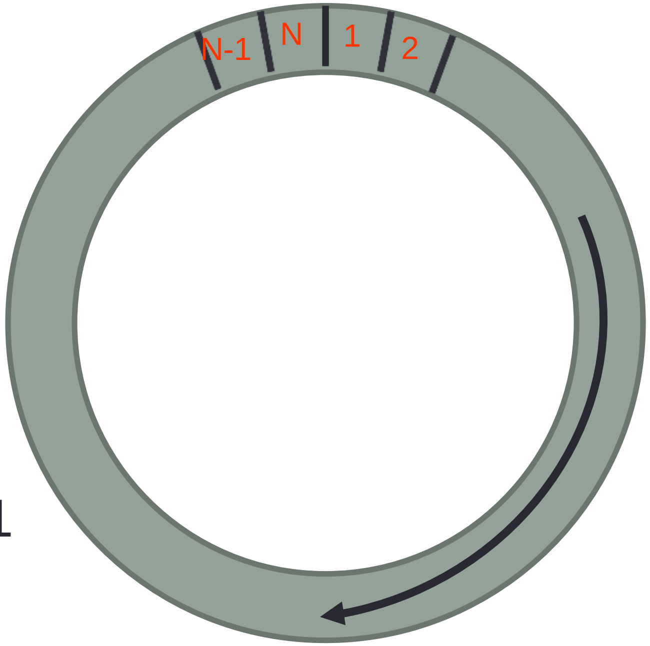
epcc

THE UNIVERSITY OF EDINBURGH

Boundary conditions

- What happens if a car falls off the end of the road?
  - when does a car appear at the start?

- Consider a roundabout
  - periodic boundary conditions:
    - up from last cell *N* is first cell 1
    - down from first cell 1 is last cell *N*

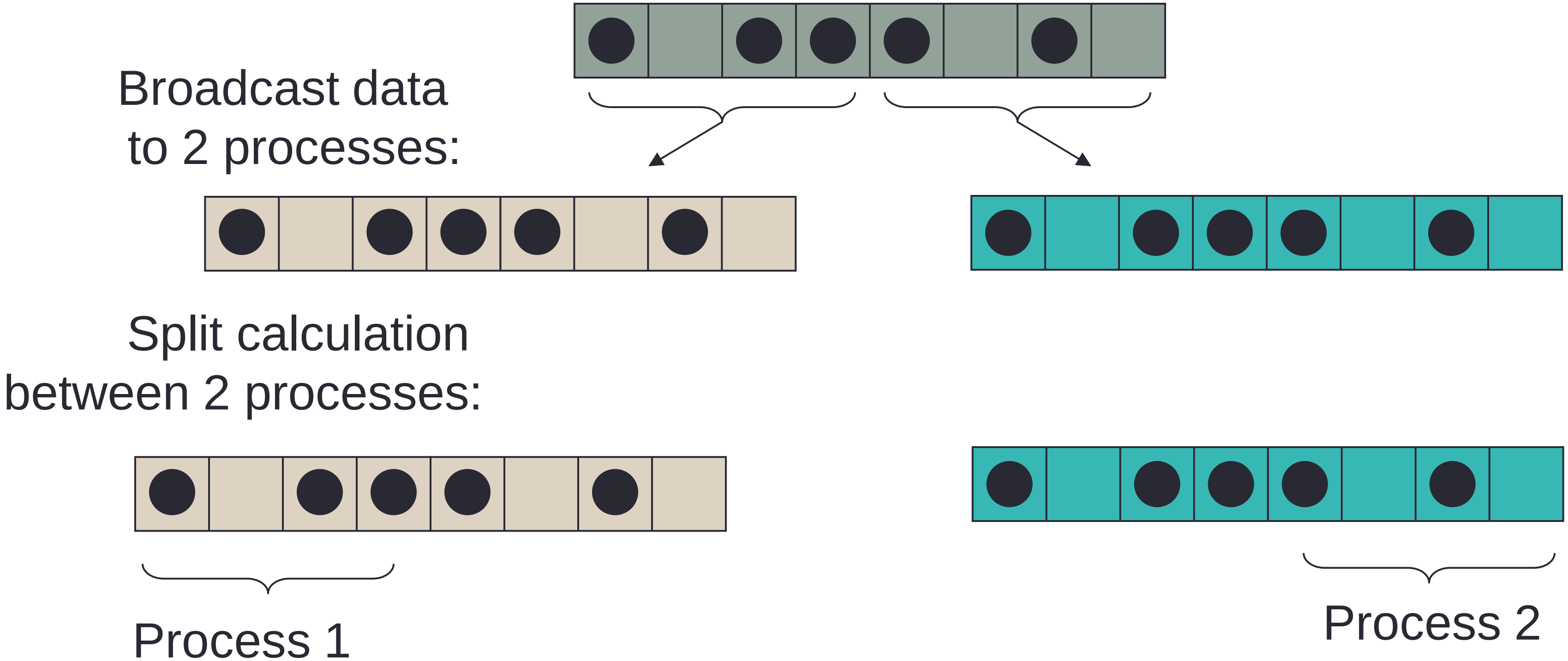


- Implement with *ghost* or *halo* cells
  - road has *N*+2 cells 0, 1, 2, ..., *N*-1, *N*, *N*+1
  - copy cell 1 to cell *N*+1 and cell *N* to cell 0
  - then update cells 1 to *N* as normal

Pseudo Code

```
declare arrays old(i) and new(i), i = 0,1,...,N,N+1
initialise old(i) for i = 1,2,...,N-1,N (eg randomly)
loop over iterations
  set old(0) = old(N) and set old(N+1) = old(1)
  loop over i = 1,...,N
    if old(i) = 1
      if old(i+1) = 1 then new(i) = 1 else new(i) = 0
    if old(i) = 0
      if old(i-1) = 1 then new(i) = 1 else new(i) = 0
  end loop over i
  set old(i) = new(i) for i = 1,2,...,N-1,N
end loop over iterations
```

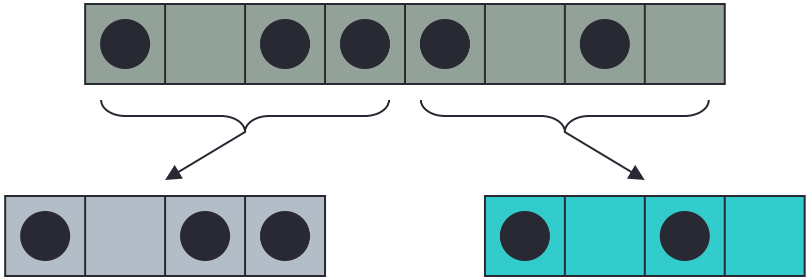
Parallelisation Strategy (1)



- Globally resynchronise all data after each move
  - a **replicated data** strategy
- Every process stores the entire state of the calculation
  - e.g. any process can compute total number of moves

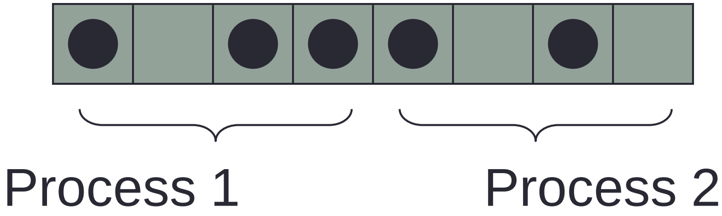
Parallelisation Strategy (2)

Scatter data
between 2 processes:
distributed data strategy



•Internal cells can be updated independently.
•Must communicate with neighbouring processes to update edge cells.
•Sum local number of moves on each process to obtain total number of moves at each iteration.

Split calculation
between 2 processes:



•Each process must know which part of roadway it is updating.
•Synchronise at completion of each iteration and obtain total number of moves.

Parallelisation

- Load balance not an issue
  - updates take equal computation regardless of state of road
  - split the road into equal pieces of size N/P
- For each piece
  - rule for cell i depends on cells i-1 and i+1
  - the N/P - 2 interior cells can be updated independently in parallel
  - however, the edge cells are updated by other processors
    - similar to having separate rules for boundary conditions
- Communications required
  - to get value of edge cells from other processors
  - to produce a global sum of the number of cars that move

Message Passing Parallelisation