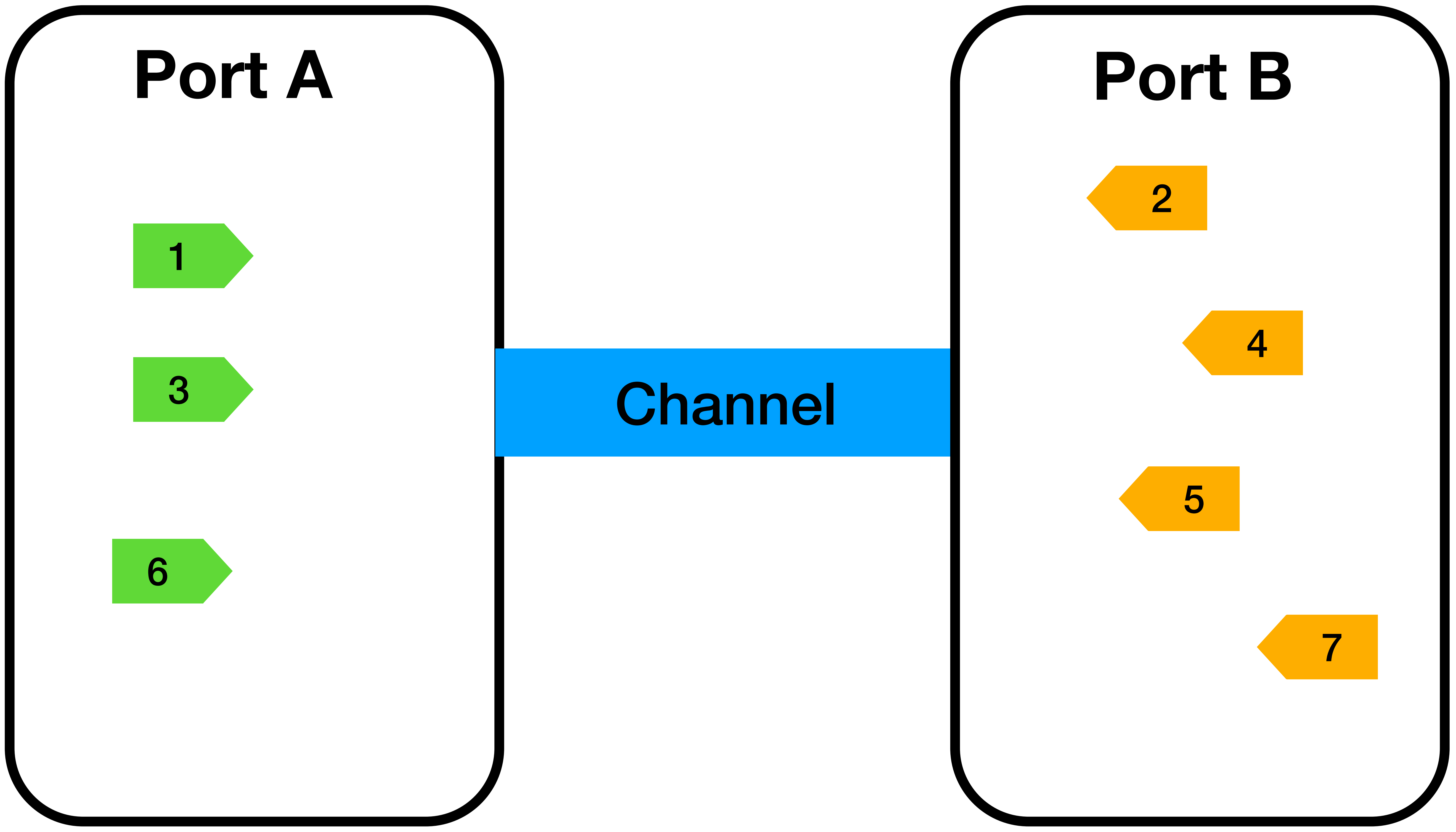
Sequence Dependent Scheduling 1

Peter Stuckey & Guido Tack

MONASH University

Single Channel Problem



#A-to-B = **East**

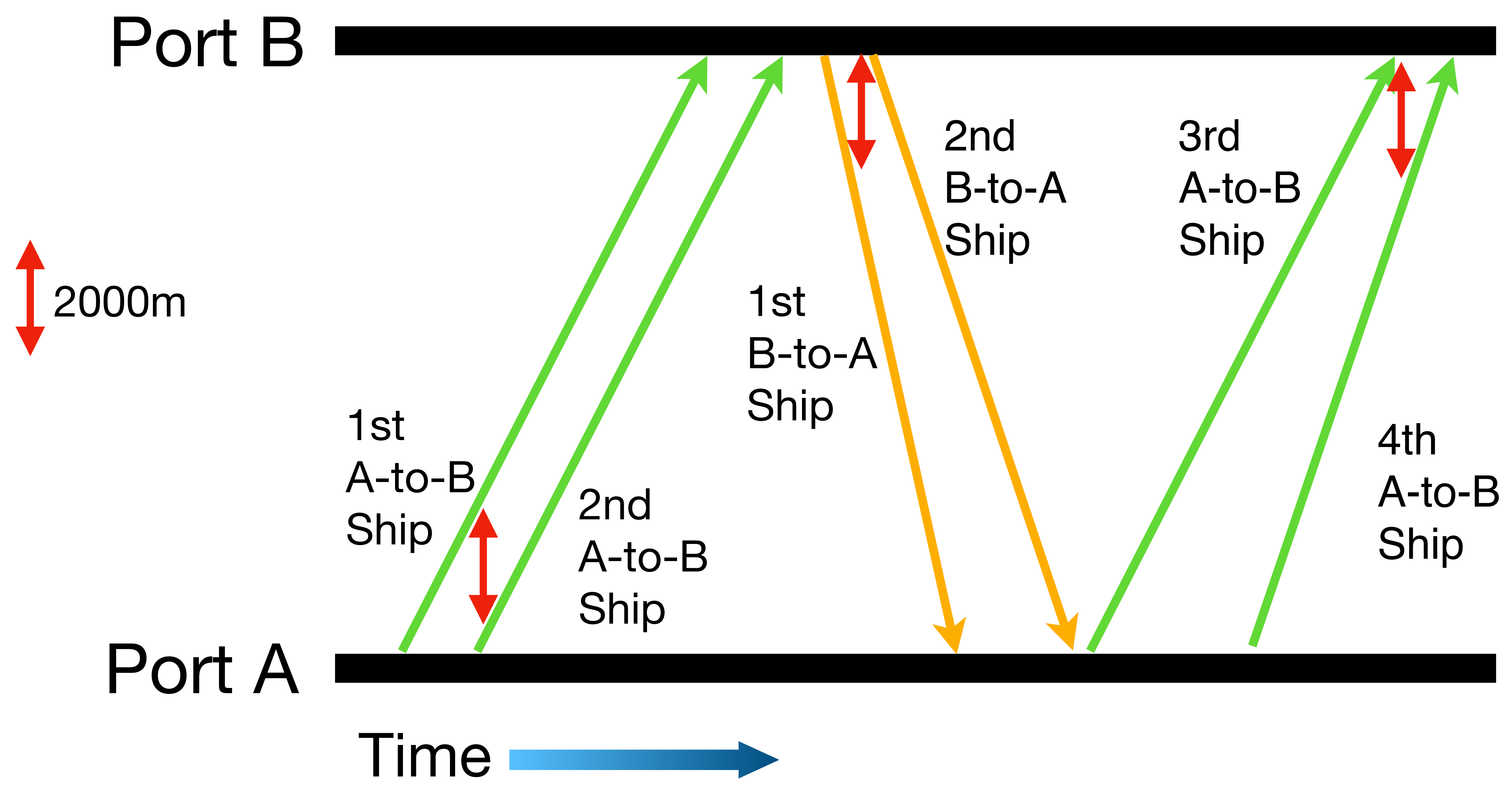
#B-to-A = **West**

Single Channel Problem

- Given ports `A` and `B` connected by a channel. Consider a set of ships going `East` from port `A` to `B`, or going `West` from port `B` to `A`. We need to choose when the ships should enter the channel
  - Each ship has a specific speed and can leave **no earlier than** a desired time for that ship
  - The channel is 32000m long
  - A ship can enter only if the channel is **clear**, i.e. no ships sail in opposite directions simultaneously
  - Two ships cannot be closer than 2000m
  - **Minimize** the time to move all the ships

Leeway Constraint

- A channel is a complex unary resource
- The time distance between ships is dependent on the relative directions and relative speeds



Sequence Dependence

- Particularly for unary resources
- The schedule may depend on which task precedes another on that resource
- Examples
  - smelting: machinery must cool to perform “cold” task after “hot task”
  - embroidery: colors of thread may need changing between tasks
  - single channel: ships traveling in different directions need to wait until channel is clear
- Effectively the start time of the next task is delayed depending on the previous task

Single Channel Data

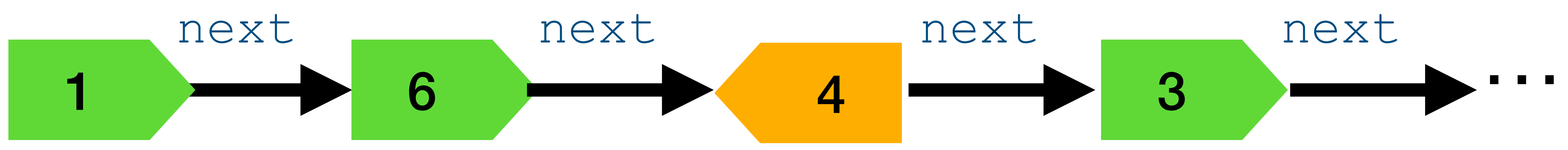
```
int: len;
int: leeway;  % leeway between 2 ships
int: max_time;
set of int: Time = 0..max_time;

enum Ship;
array[Ship] of int: speed;   % 1000m time
array[Ship] of int: desired; % desired time
enum Direction = { West, East };
array[Ship] of Direction: dirn;
```

Single Channel Data

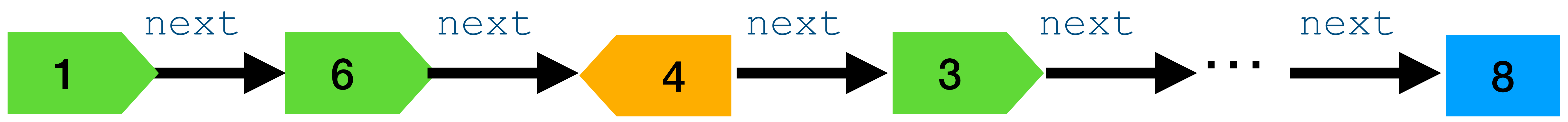
```
len = 32;
Ship = { Tanacious, Seaworthy, Majestic,
         Dauntless, Joyful, Steadfast,
         Worthy };
speed = [5,6,4,5,4,7,3];
desired = [0,0,47,33,100,125,175];
dirn = [West,East,West,East,East,West,East];
max_time = 600;
leeway = 2;
```

Modeling Activity Sequences

- In order to make decision about the next activity (ship), we need to know what the current activity (ship) is



- But the last ship does not have a next ship!
- Introduce a dummy ship at the end



Single Channel Decisions

```
enum Ship0 = S(Ship) ++ {Dummy}; % add dummy
enum Direction0 = D(Direction) ++ {Nowhere};
array[Ship0] of Direction0: dirn0  = array1d(Ship0,
   [D(dirn[s]) | s in Ship] ++ [Nowhere]);
array[Ship0] of int: speed0 = array1d(Ship0,
    speed ++ [0]);

array[Ship0] of var Time: start;
array[Ship0] of var Time: end =
    [start[s] + len*speed0[s] | s in Ship0];

array[Ship] of var Ship0: next; % next ship
```

Single Channel Constraints

- Dummy ship is last

```
constraint start[Dummy] = max_time;
```

- Cannot leave before desired time

```
constraint forall(s in Ship)(
  start[S(s)] >= desired[s]
);
```

- The next ships are all different

```
constraint all_different(next);
```

Single Channel

- Reasoning about the channel
    - once we know the next ship, it’s reasonably simple
- If the next ship is in the **opposite direction**, then it can only start **once the current ship ends**
- If the next ship is in the **same direction**, then it must start **after the current ship travels 2000m**
- Is that enough?
    - NO, a ship is **not** allowed to “catch up”
    - the next ship must still be at least 2000m apart from the current ship **when** the current ship reaches the destination

Single Channel

- Relationship between a ship and its next ship

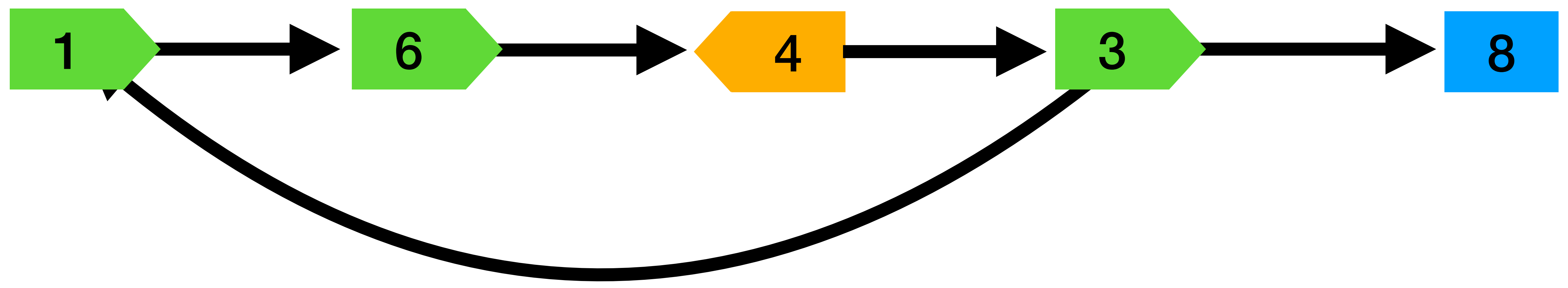
```
constraint forall(s in Ship) (
  if dirn0[S(s)] != dirn0[next[s]] then
    end[S(s)] <= start[next[s]]
  else
    start[S(s)]+speed[s]*leeway <= start[next[s]]
    /\
    end[S(s)] <= end[next[s]]-speed0[next[s]]*leeway
  endif
);
```

- Objective

```
solve minimize max(s in Ship)(end[S(s)]);
```

Subtleties of the Model

- Is the `all_different` constraint enough
  - for example this satisfies the constraint



- **BUT** start times of the ships increase
- For other similar problems we will need the `circuit` global constraint

Single Channel Data

```
start = [271, 43, 385, 33, 115, 281, 175, 600];
end   = [431, 235, 513, 193, 243, 505, 271, 600];
next  = [S(Steadfast), S(Joyful), Dummy,
         S(Seaworthy), S(Worthy), S(Majestic),
         S(Tanacious)]);
----------
==========
```

EOF