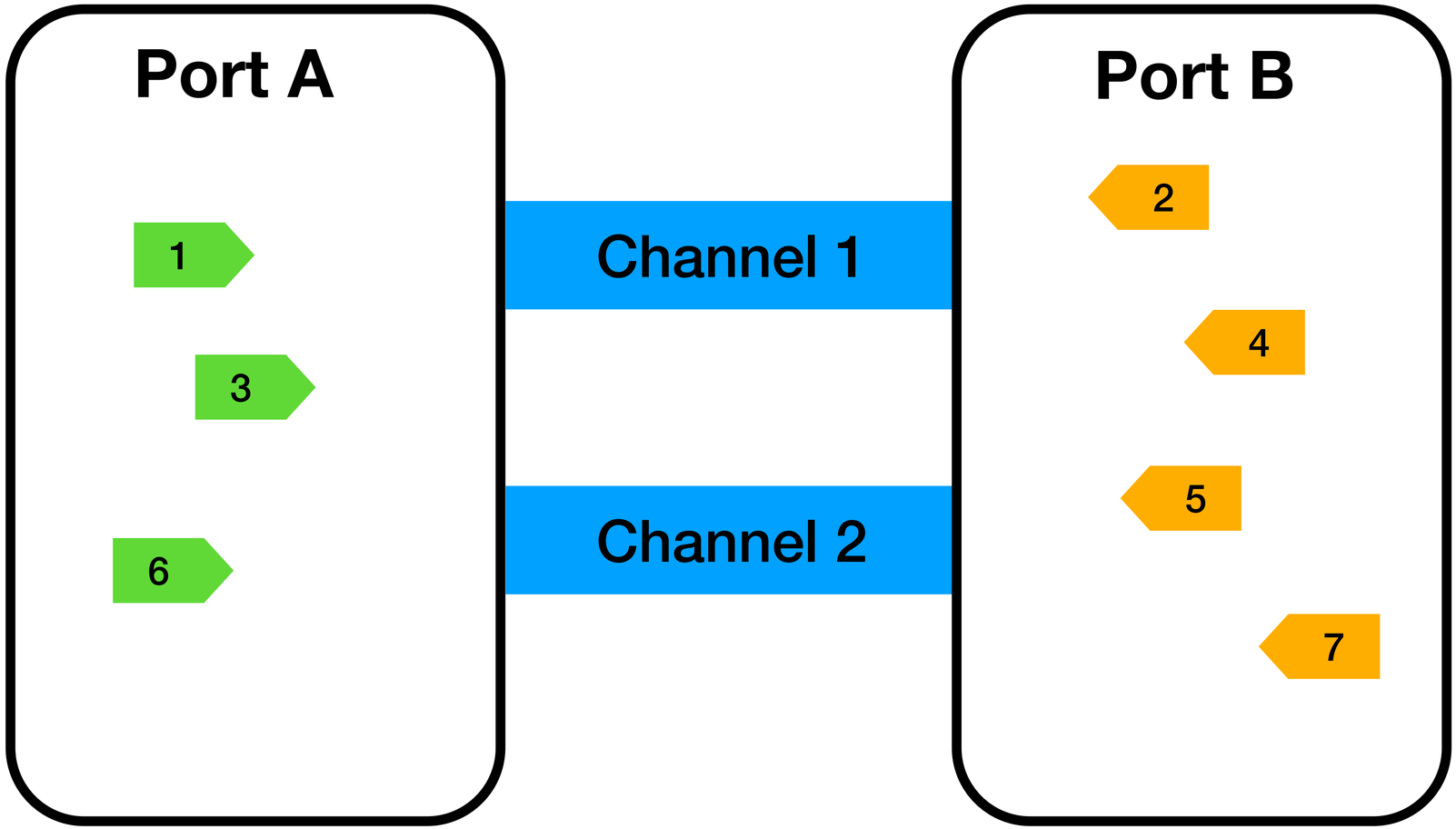
Sequence Dependent Scheduling 2

Peter Stuckey & Guido Tack

MONASH
University

Double Channel Problem



#A-to-B = East

#B-to-A = West

Double Channel Problem

- Given ports `A` and `B` connected by **channels 1 and 2**. Consider a set of ships going `East` from port `A` to `B`, or going `West` from `B` to `A`. We need to choose (a) **which channel to use for each ship** and (b) when the ships should enter the selected channels
  - Each ship has a specific speed and can leave **no earlier than** a desired time for that ship
  - Channels 1 and 2 are 24000m and 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

Double Channel Data

```
enum Channel;
array[Channel] of 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;
```

Double Channel Data

```
Channel = {C1, C2};
len = [24,32];
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 = array1d(Ship,
[West,East,West,East,East,West,East]);
max_time = 400;
leeway = 2;
```

Double Channel Decisions

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

array[Ship0] of var Channel: channel;

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

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

Double Channel Constraints

- **Changed:** Dummy ships are last and in a fixed channel

```
constraint forall(c in Channel)(
  start[Dummy(c)] = max_time
);
constraint forall(c in Channel)(
  channel[Dummy(c)] = c
);
```

- **New:** `next` of ship is in the same channel

```
constraint forall(s in Ship)(
  channel[next[s]] = channel[S(s)]
);
```

- Other constraints **remain the same**

Solving the Model

```
channel =[C1, C2, C1, C2, C2, C1, C2, C1, C2];
start   =[0, 43, 47, 33, 115, 125, 175, 400,
          400];
end     =[120, 235, 143, 193, 243, 293, 271, 400,
          400];
next    =[S(Majestic), S(Joyful), S(Steadfast),
          S(Seaworthy), S(Worthy), Dummy(C1),
          Dummy(C2)]);
----------
==========
```

Order Dependent Setup Times

- If there are order dependent setup times or costs
  - model the **next task** explicitly
  - add constraints to ensure the setup time or cost is paid
- Examples are:
  - direction change in channel
  - mode change in machine shop scheduling
  - cool down time for smelting jobs at different temperatures

Summary

- Complex scheduling applications
  - have **interdependencies** between a task and the task that follows it

- We need to model for each task
  - which task is next, and
  - how that constrains the schedule

EOF