Considering the Real Cost of Communication in Task Scheduling

Oliver Sinnen

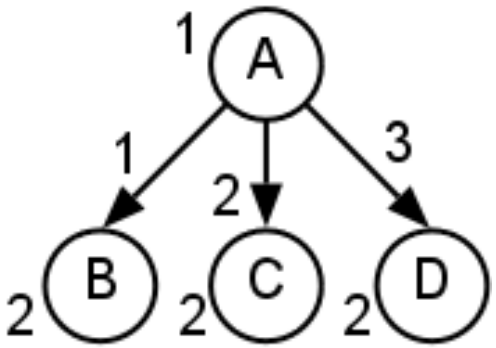
Department of Electrical and Computer Engineering
University of Auckland
New Zealand

www.ece.auckland.ac.nz/~sinnen/

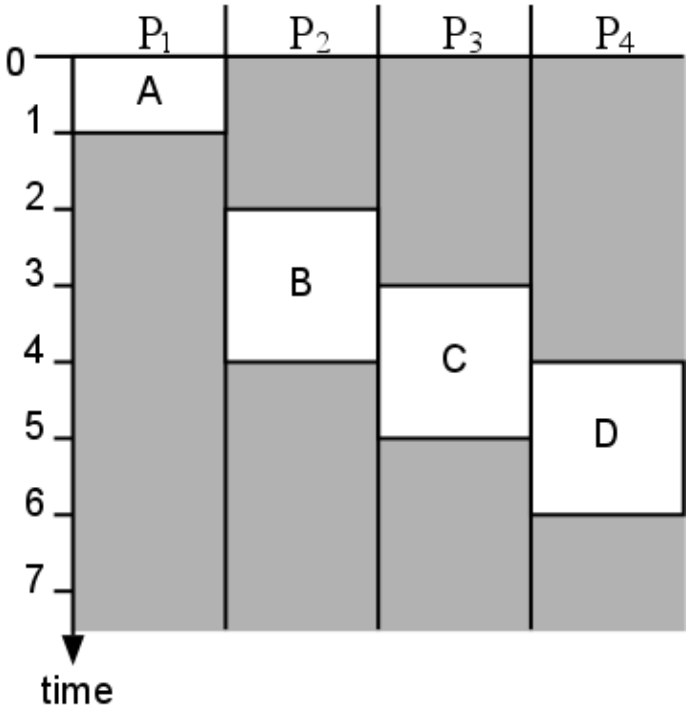
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

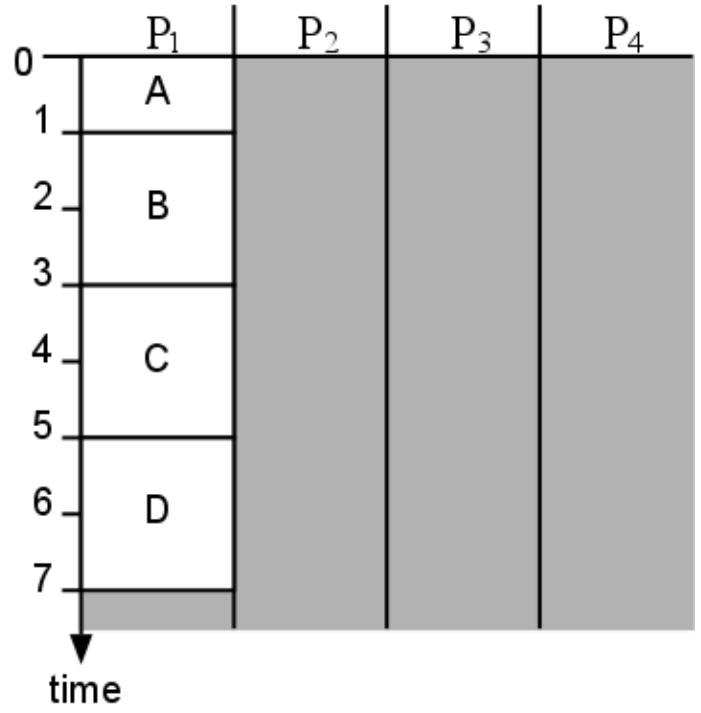
Scheduling trade-off



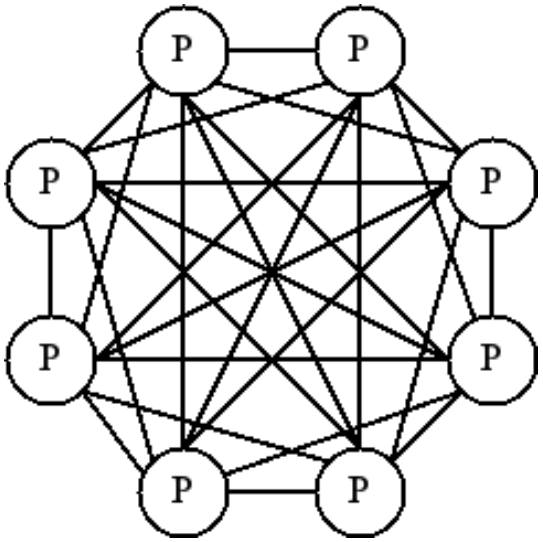
high concurrency



low communication



Classic system model



e.g. 8 processors

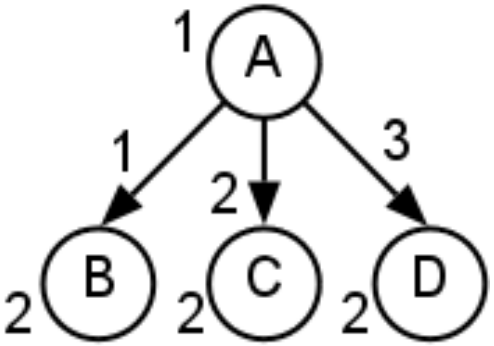
- Dedicated system
- Dedicated processors
- Zero-cost local communication
- Communication subsystem
- Concurrent communication
- Fully connected

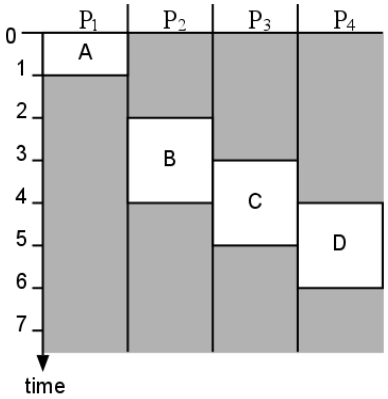
=> Part I: Scheduling models to consider contention and processor involvement

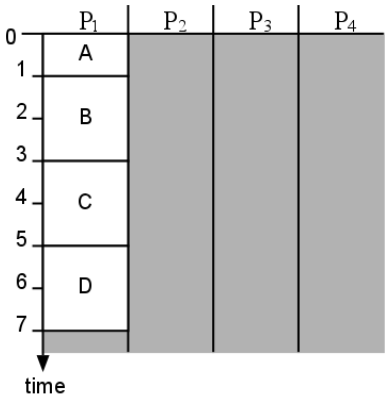
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

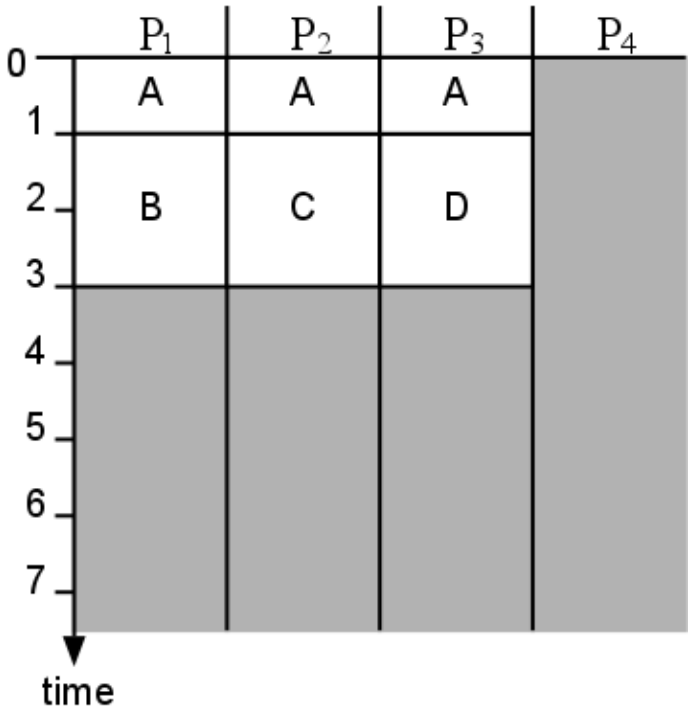
Task duplication can help







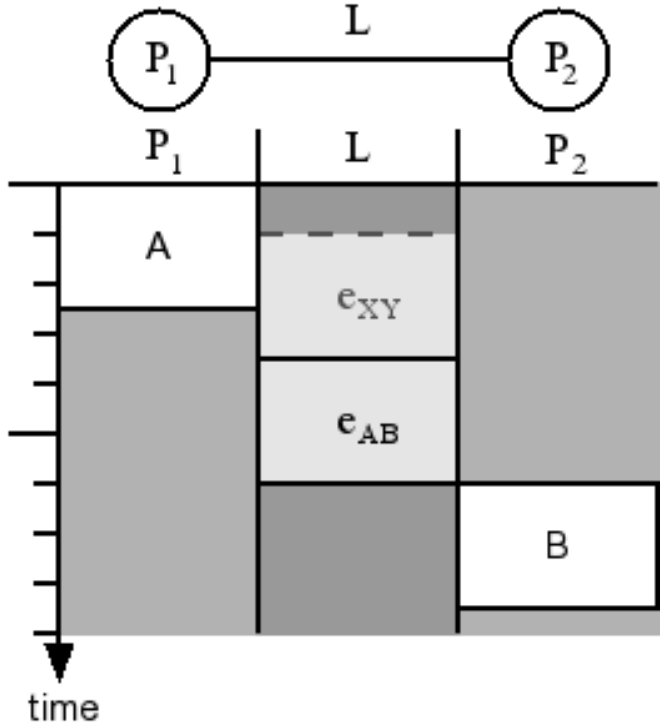
task duplication



Contention-aware task duplication

- Avoiding communication even more important under contention model
- But, strangely, task duplication algorithms for contention-aware scheduling not much investigated

=> Part II of this talk



Outline

Part I:

- Classic task scheduling
- Contention aware task scheduling
- Considering processor involvement

Part II:

- Task duplication under contention model
- Algorithm
- Experimental results
- Conclusions

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Classic task scheduling

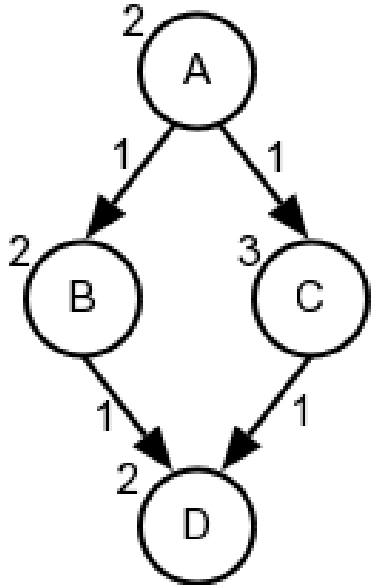
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Classic task scheduling

Schedule definitions: DAG: *G(V,E)*, node *n*, edge *e*

- start time: $t_s(n)$; finish time: $t_f(n)$
- processor assignment: *proc(n)*



Constraints:

- Processor constraint:

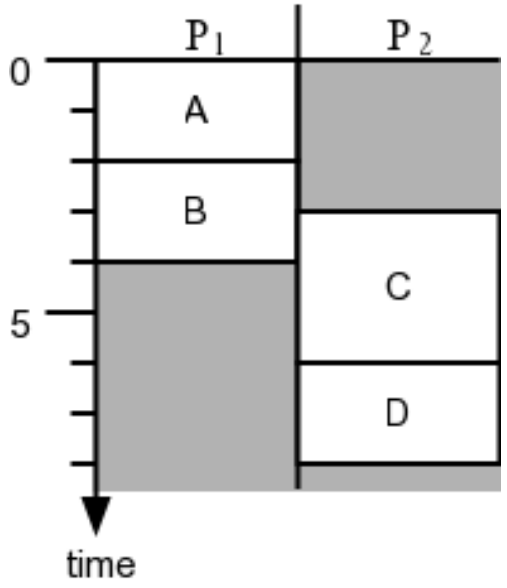
  $proc(n_i)=proc(n_j) \Rightarrow t_s(n_i) \geq t_f(n_j)$ or $t_s(n_j) \geq t_f(n_i)$

- Precedence constraint:

  for all edges e_{ji} from n_j to n_i

  $t_s(n_i) \geq t_f(n_j)$

  $+ c(e_{ji})$ if remote, i.e. $proc(n_i) \neq proc(n_j)$

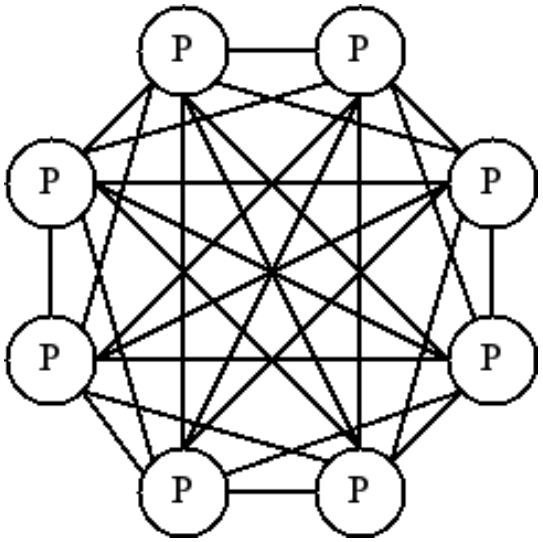


Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Classic system model

system model



e.g. 8 processors

Properties:

- Dedicated system
- Dedicated processors
- Zero-cost local communication
- Communication subsystem
- Concurrent communication
- Fully connected

Goal: find schedule with shortest schedule length (makespan)

=> NP-hard problem => many heuristics

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

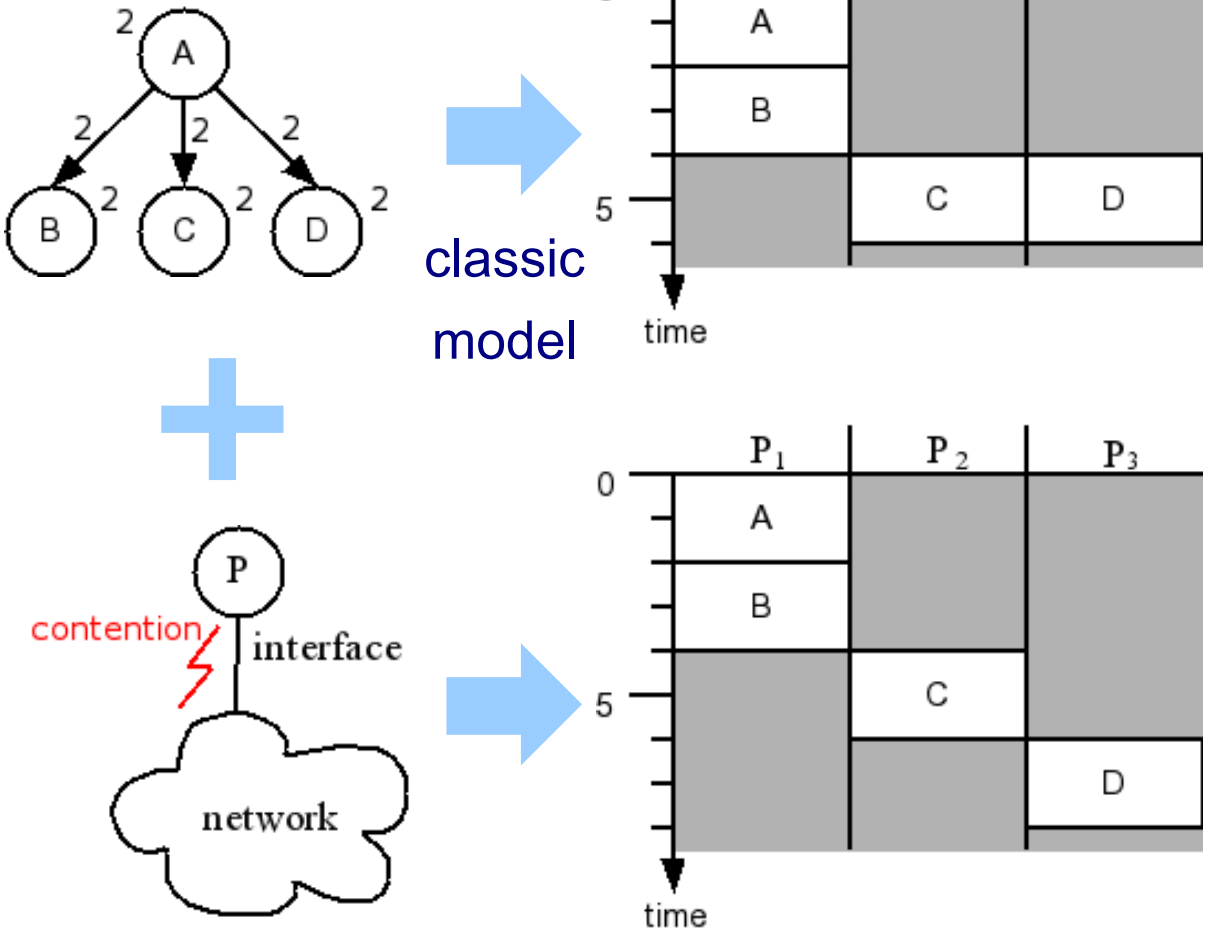
Contention aware task scheduling

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Communication contention

contention example



- End-point contention
  - For Interface
- Most networks *not* fully connected
- Network contention
  - For network links

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Network model

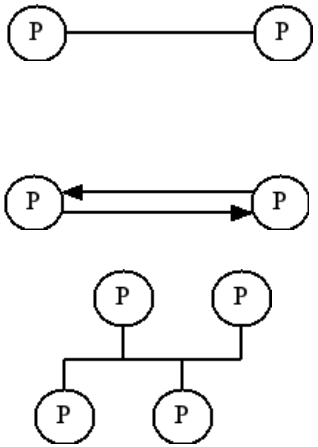
New network graph:

Vertices: processors (P) and switches (S)

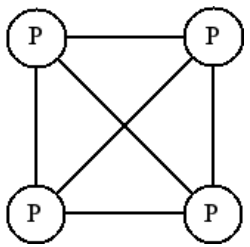
- Static and dynamic networks
- End-point and network contention

Edges: communication links (L)

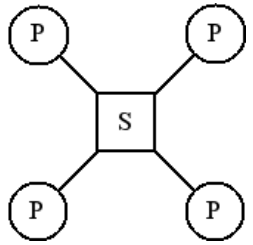
- Undirected edges
  - Half duplex
- Directed edges
  - Full duplex
- Hyperedges
  - Bus



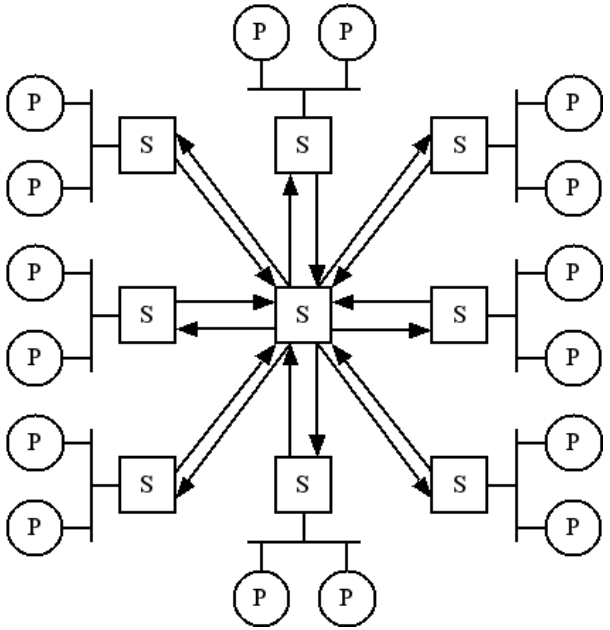
fully connected



switched LAN



example: 8 dual-processor cluster



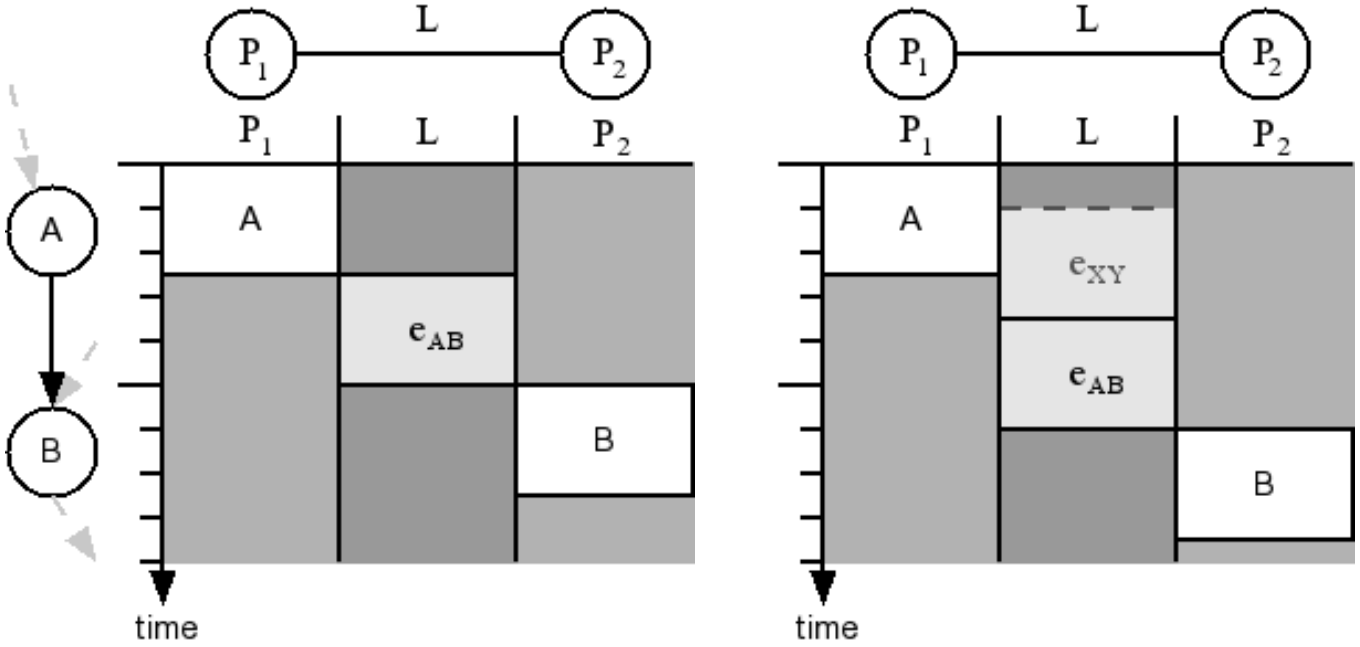
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Contention aware scheduling

Contention awareness:

- Schedule **edges** on links
- Integration of edge scheduling into task scheduling
  - Only impact on start time of node:
  - $t_s(n_i) \geq t_f(e_{ji})$ (precedence constraint)



without contention

with contention

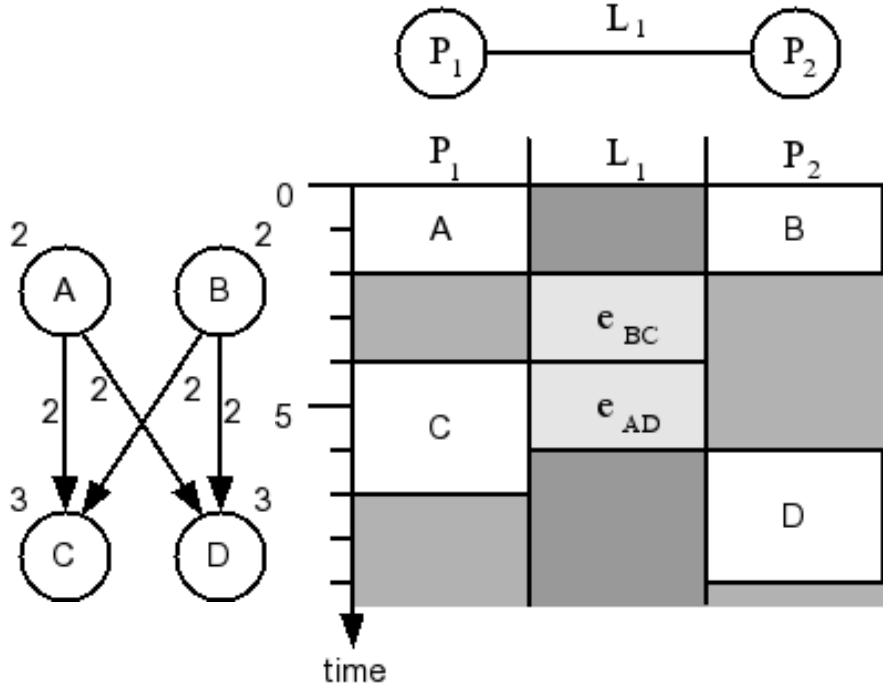
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Contention aware list scheduling

List Scheduling

1. Make node list
   - according to priority, respecting precedence constraints
2. For each *n* in node list:
   a) Find *P* that allows earliest start time of current *n*, by tentatively scheduling all incoming edges on links
   b) Schedule *n* on chosen *P* and incoming edges on respective links



Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Processor involvement

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Processor Involvement

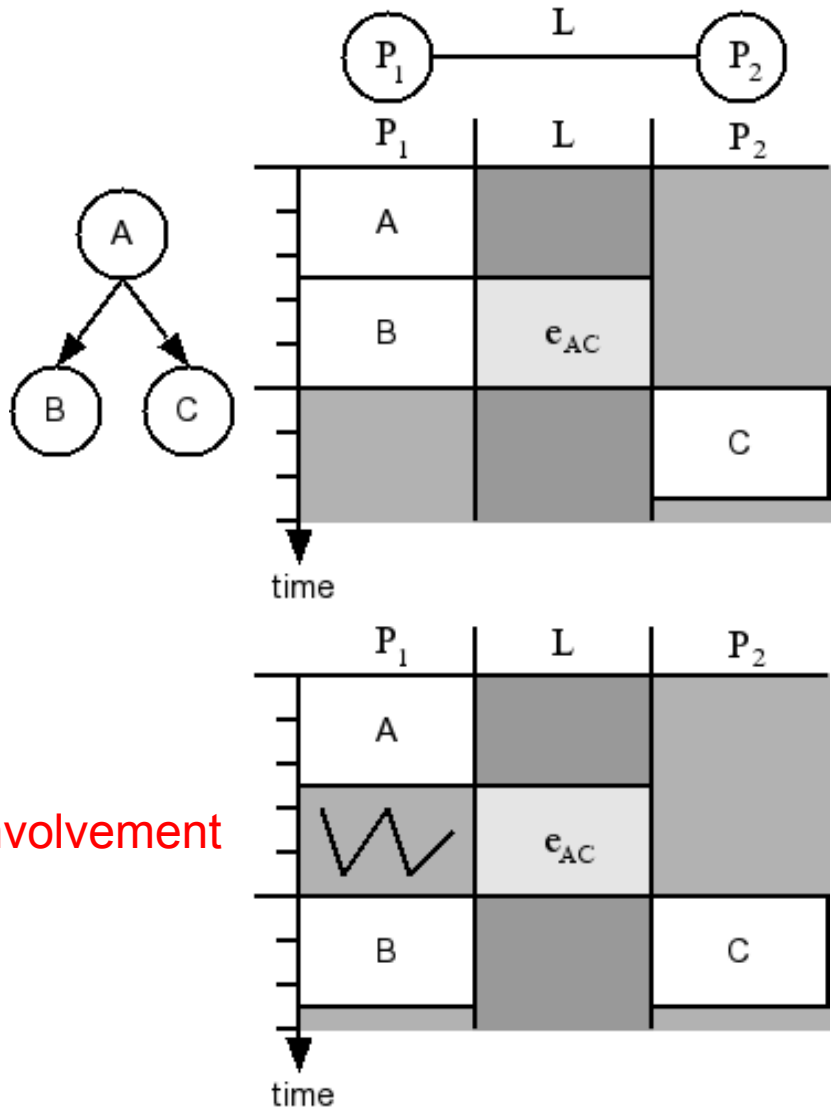
- Results suggest: contention model still not accurate enough

System model properties:

- ...
- Communication subsystem
- Concurrent communication ✓
- Fully connected ✓

Discrepancy real system ↔ model

⇨ Involvement of processor in communication

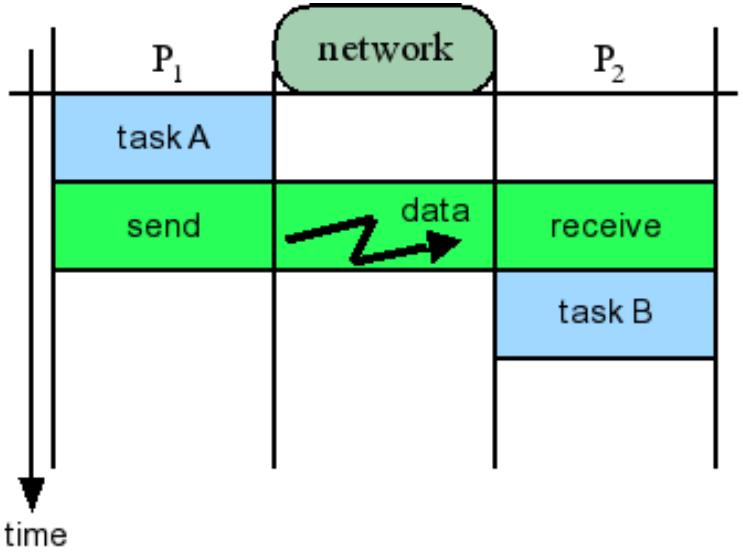


Oliver Sinnen – o.sinnen@auckland.ac.nz

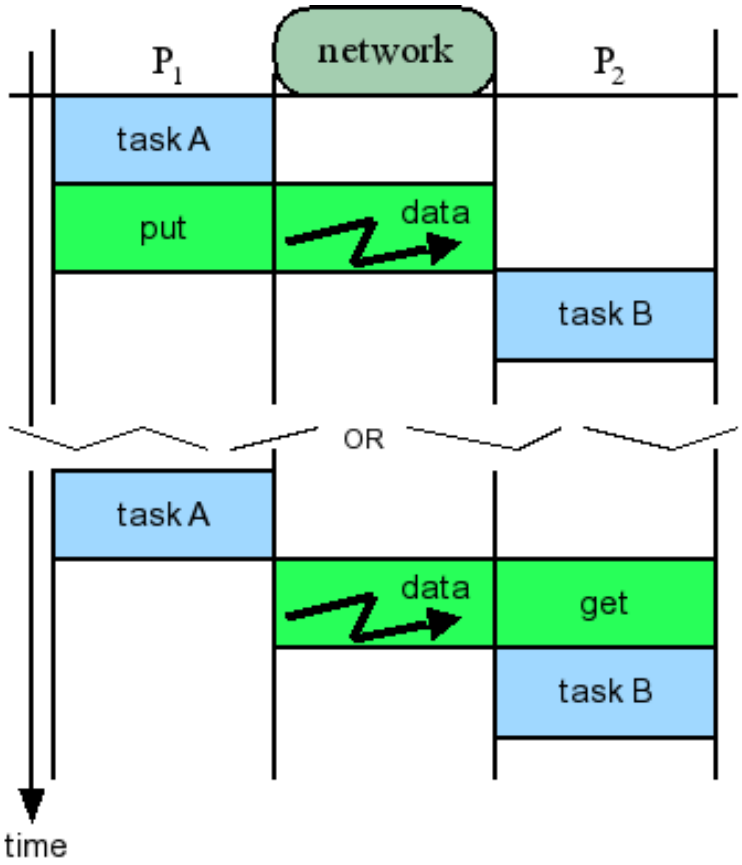
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Types of processor involvement

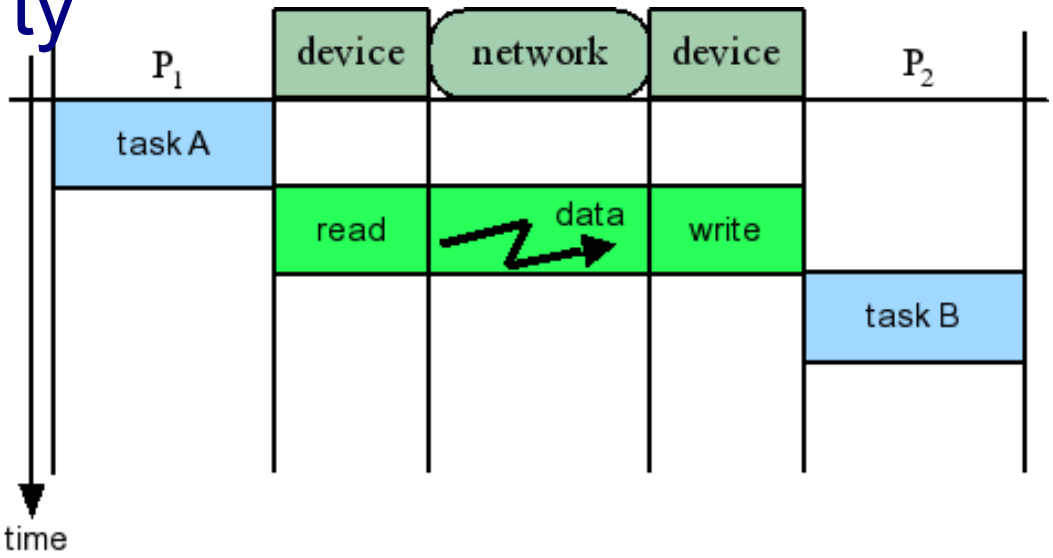
two sided



one sided



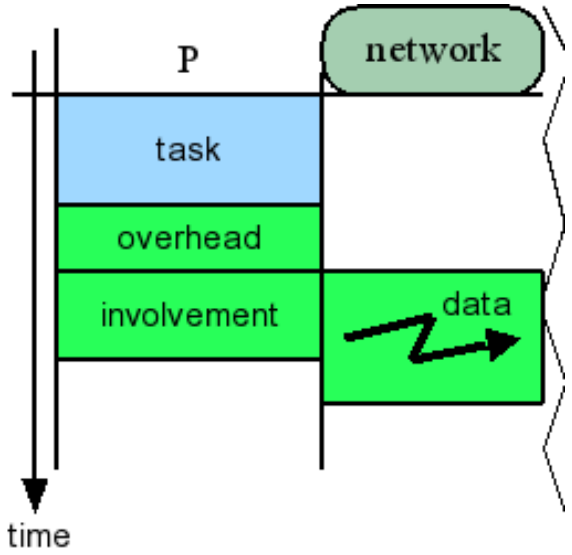
3rd party



Characteristic of involvement

Distinction of overhead and involvement

- Overhead:
  - pre and post processing
  - e.g. setup
- Involvement:
  - direct involvement while communication is already in network
  - e.g. memory copy



Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

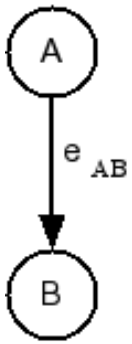
Scheduling edge on processor

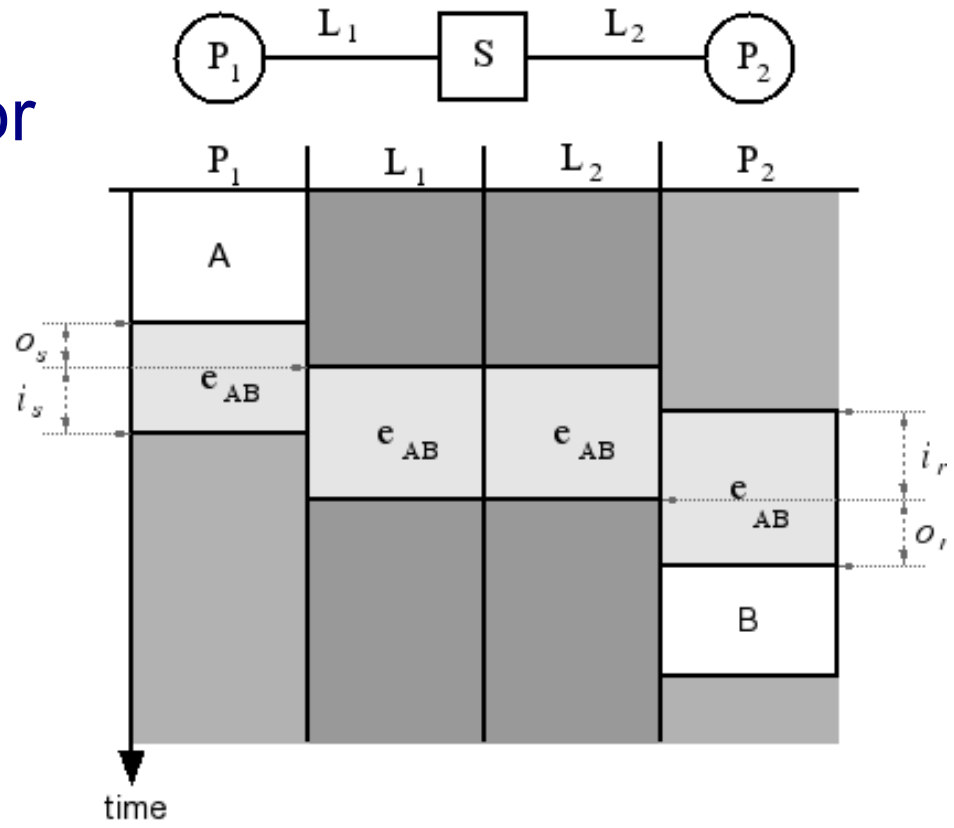
How to consider processor involvement in task scheduling?

⇒ Scheduling edge on processor

Size of edge determined by

- Involvement type
- Involvement parameters:
  - $o_{r,s}$, $i_{r,s}$





Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Involvement scheduling

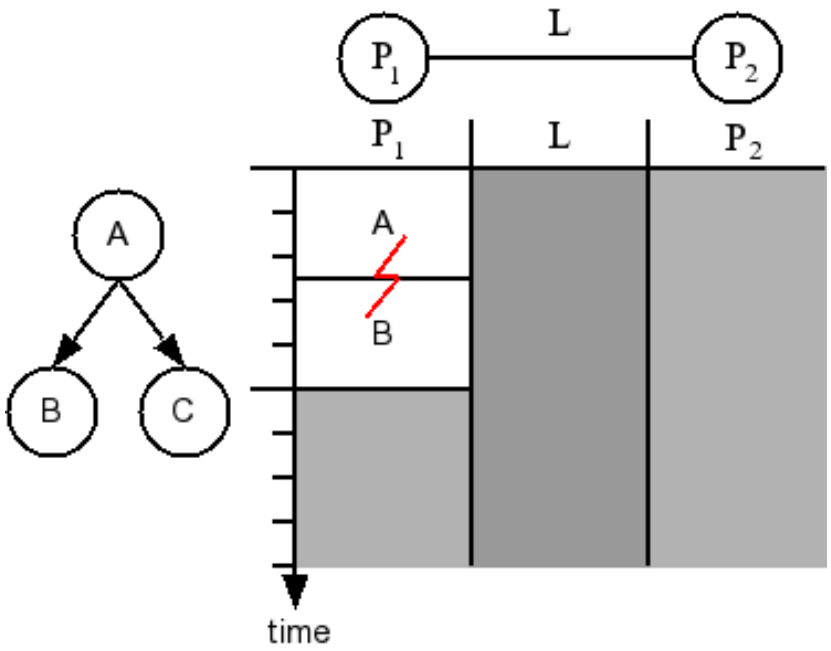
- Integrates with contention scheduling
- **But** more difficult as contention scheduling

Problem:

- Edge is only scheduled on processor *if* communication is remote

Solutions:

1. Provisional scheduling
2. Using given processor allocation



Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

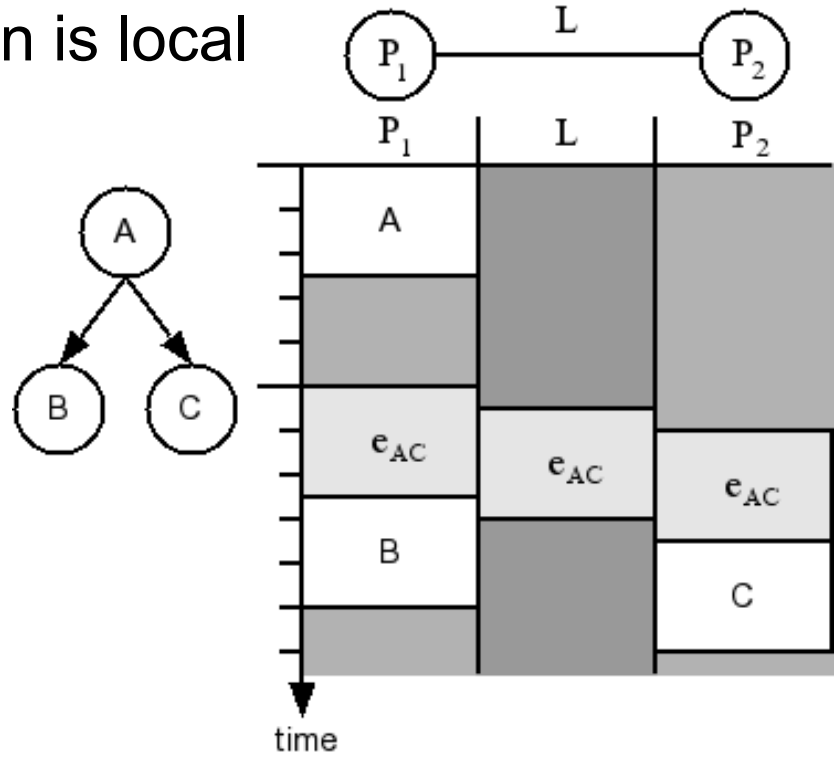
Scheduling edge on processor

1. Provisional scheduling

- Provisionally schedule all leaving edges
  - Corresponds to “worst-case” that all communications are remote
- When destination node is scheduled:
  - Schedule edge on route and destination processor
  - *Or*: remove edge if communication is local

Disadvantage:

- Leads to gaps in schedule
  - Partial solution: use insertion



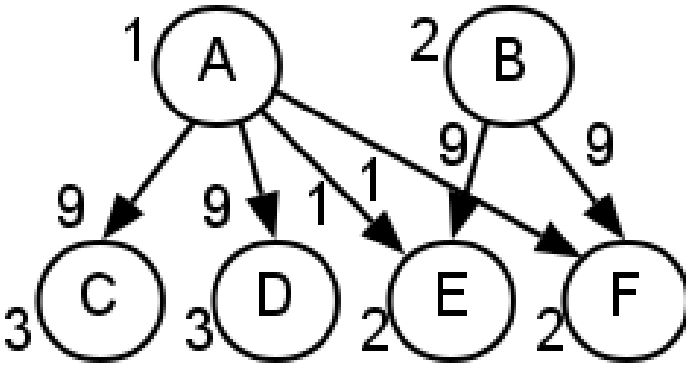
Part II:
Task duplication under contention model

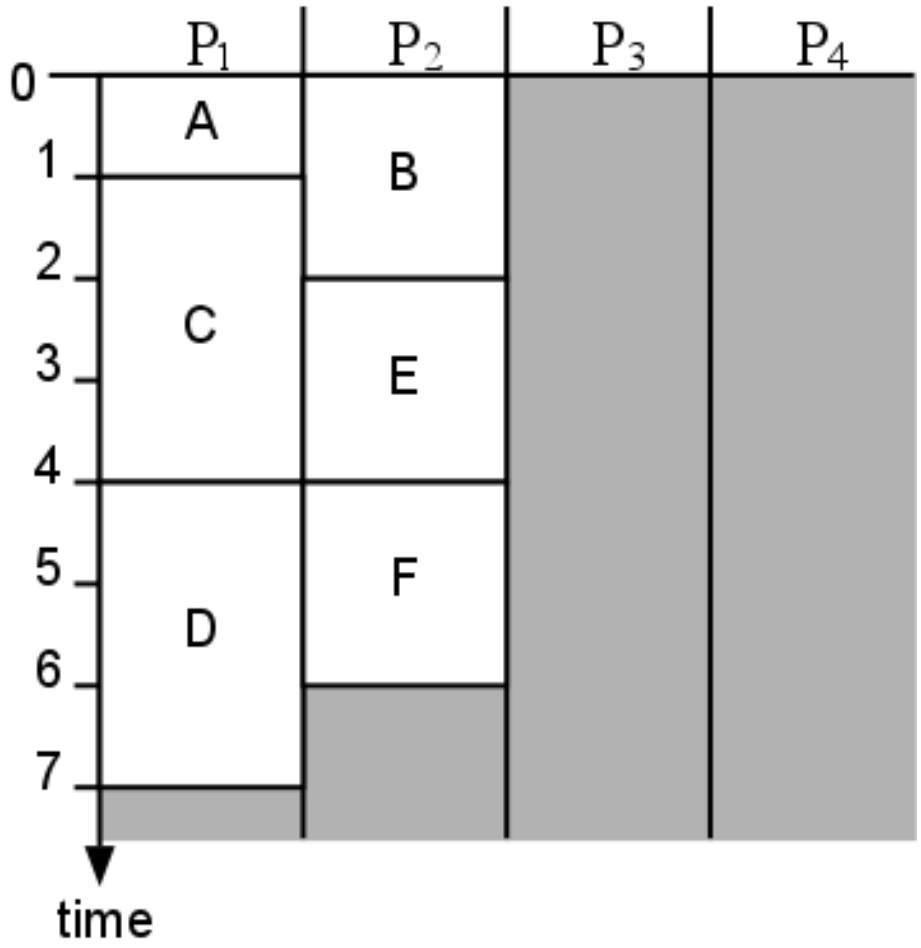
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Task duplication (classic model)

- Duplicating tasks to avoid remote communication

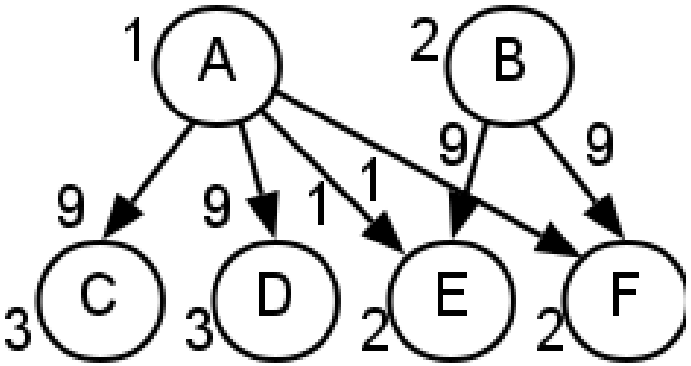


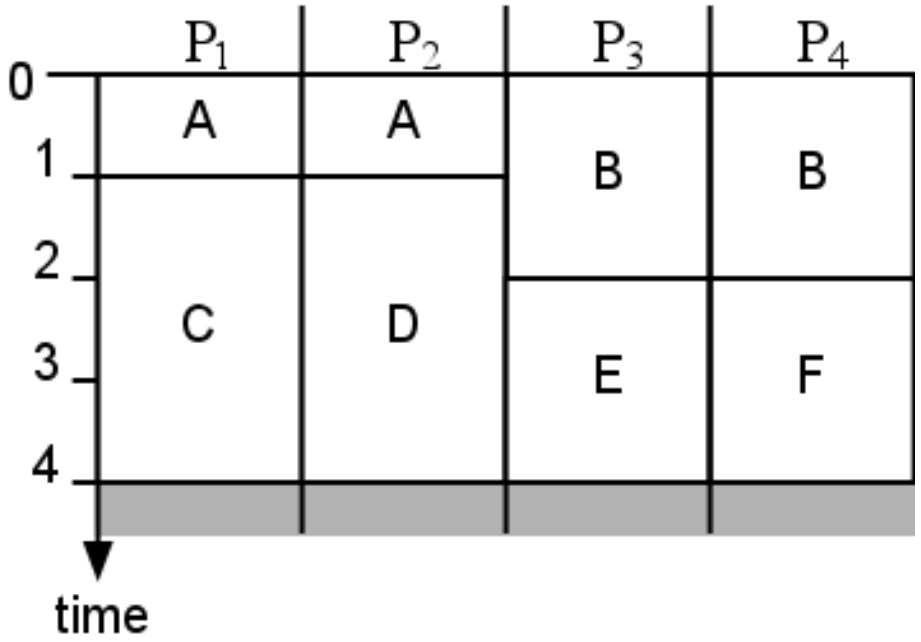


without duplication

Task duplication (classic model)

- Duplicating tasks to avoid remote communication

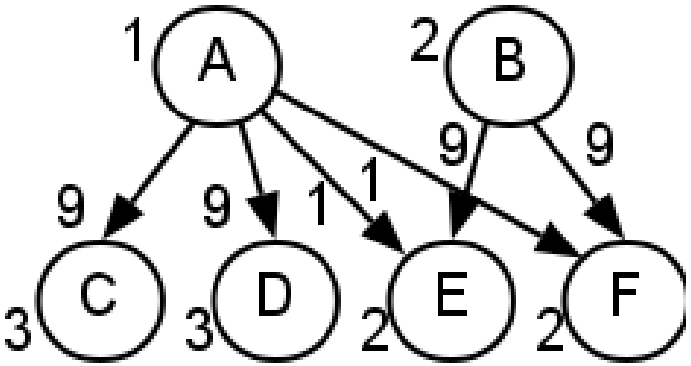


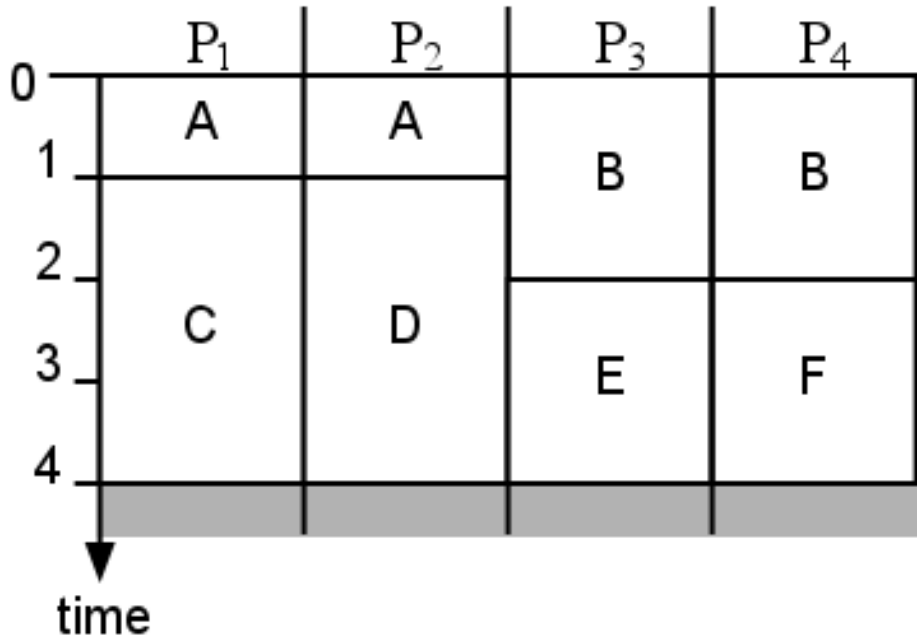


with duplication

Task duplication (classic model)

- Duplicating tasks to avoid remote communication





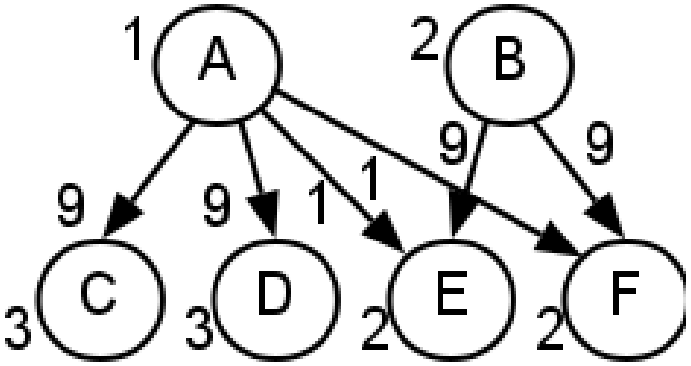
with duplication

- Not relevant from which task A (on P_1 or P_2) tasks E and F receive communication

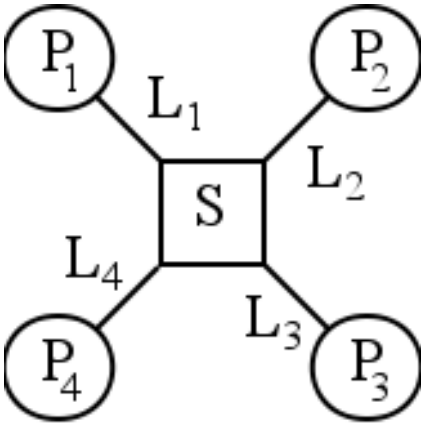
Oliver Sinnen – o.sinnen@auckland.ac.nz

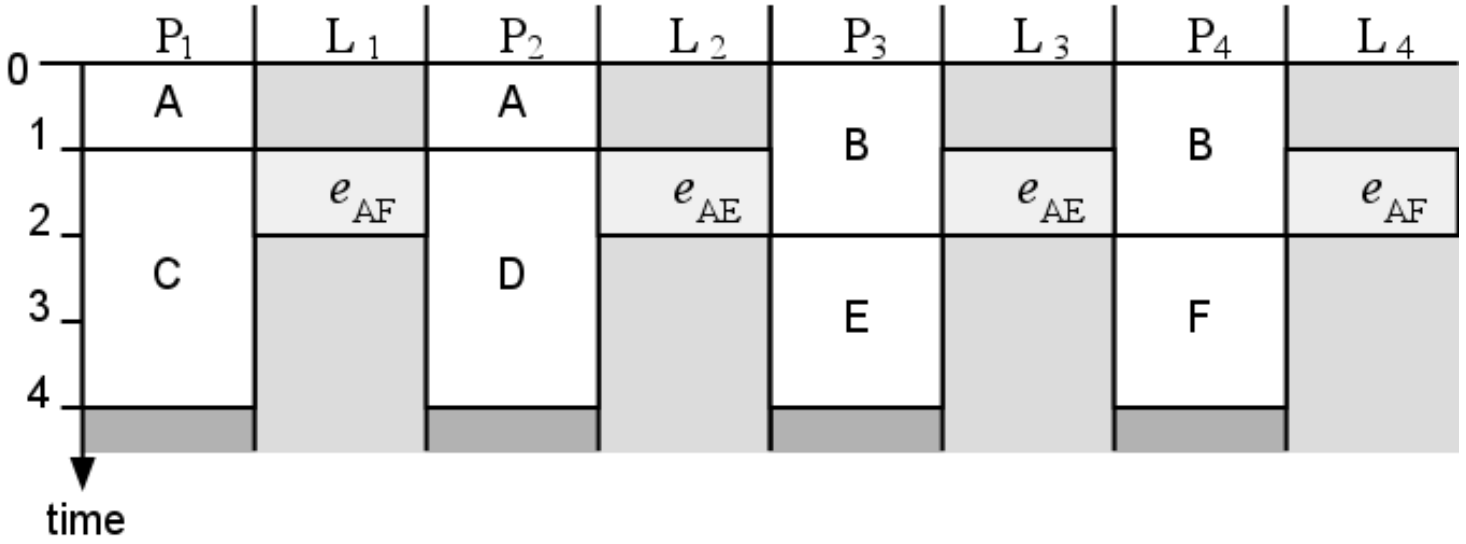
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Task duplication (contention model)



System network graph

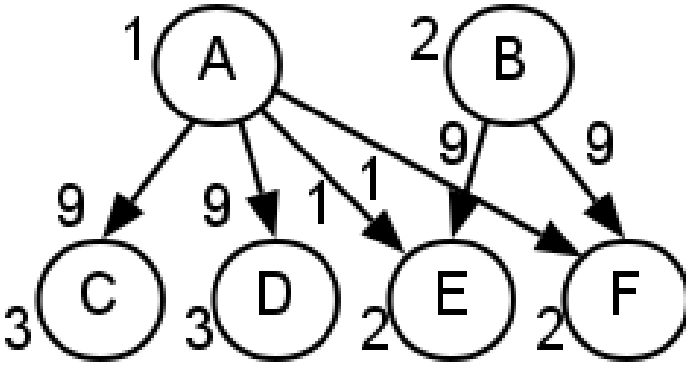




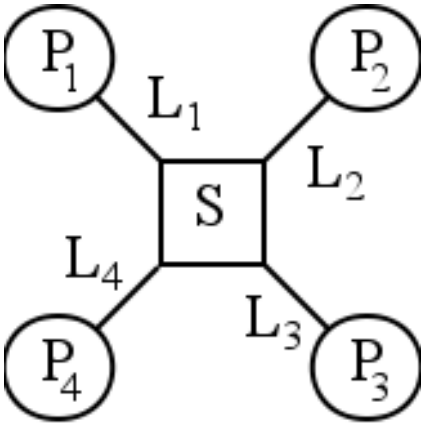
Oliver Sinnen – o.sinnen@auckland.ac.nz

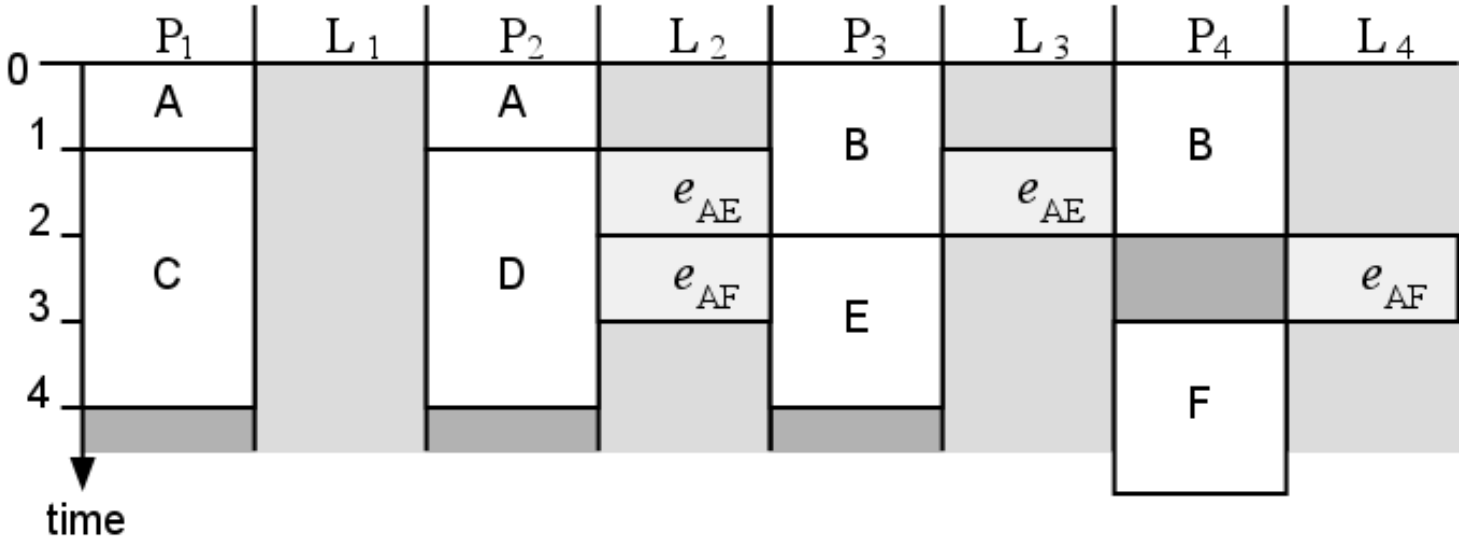
THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Task duplication (contention model)



System network graph





Relevant which duplicated task sends a communication

Contention-aware duplication

Necessary:

- Defining which duplicated tasks sends a communication

Possible:

- Sending same communication e_{ij} multiple times from one task n_i to *different* destinations of duplicated nodes n_j
- Scheduling same edge e_{ij} multiple times on single link L

Algorithm:

- Must decide which duplicated task sends a communication
- Using tentative scheduling to decide

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Algorithm

Algorithm
General structure

- List scheduling
  - Using bottom-levels to order nodes
  - Bottom-level: length of longest path starting in a node, in terms of node and edge weights
- Insertion technique
  - Schedule nodes and edges between already scheduled tasks and edges
  - Especially important for redundant node/edge removal

Algorithm Duplication aspects

- Critical parent
  - Duplicate only critical parent
  - Task from where communication arrives latest
- Recursive duplication
  - Try duplicating critical parent of critical parent etc.
- Tentative scheduling
  - On each processor, in-edges are tentatively scheduled
  - for each in-edge, from every duplicated origin node
- Redundant node/edge removal
  - Duplication can make tasks redundant → remove them and their in-edges

Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Experiments

Experimental evaluation

Workload:

- Scheduled large set of graphs: sizes, densities, CCR, graph structures (trees, random, series-parallel etc.)

Four algorithms:

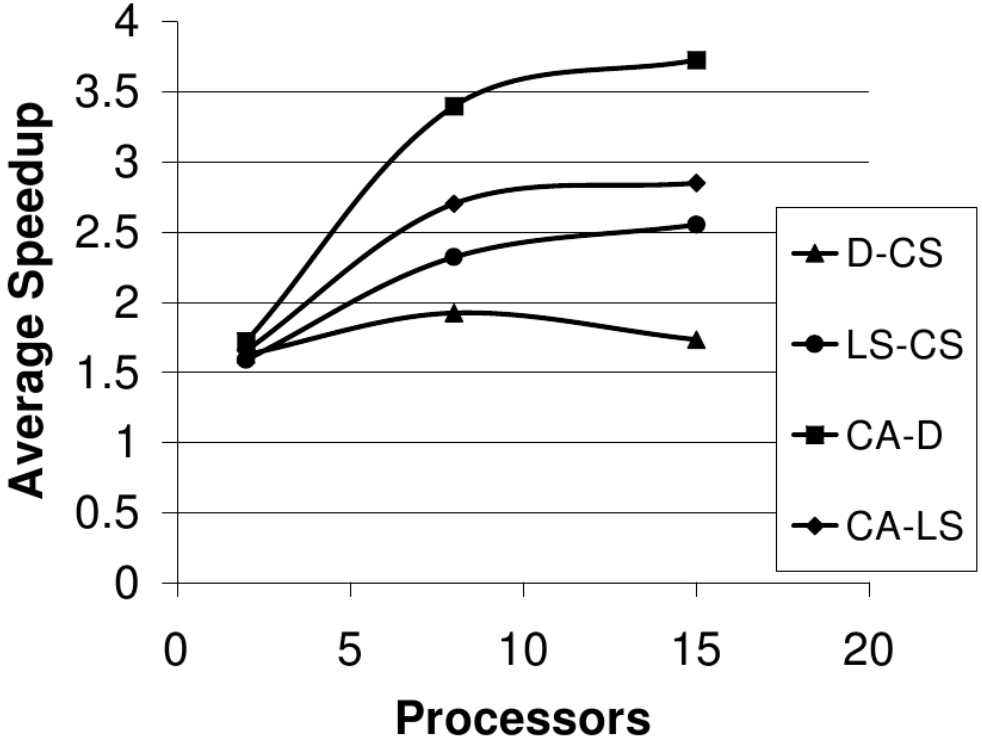
- Contention-ware:
  - **Duplication algorithm (CA-D)**
  - List scheduling (**CA-LS**)
- Classic model:
  - Duplication algorithm (**D-CS**)
  - List scheduling (**LS-CS**)
  - *Contention is simulated (CS), by rescheduling under contention model*

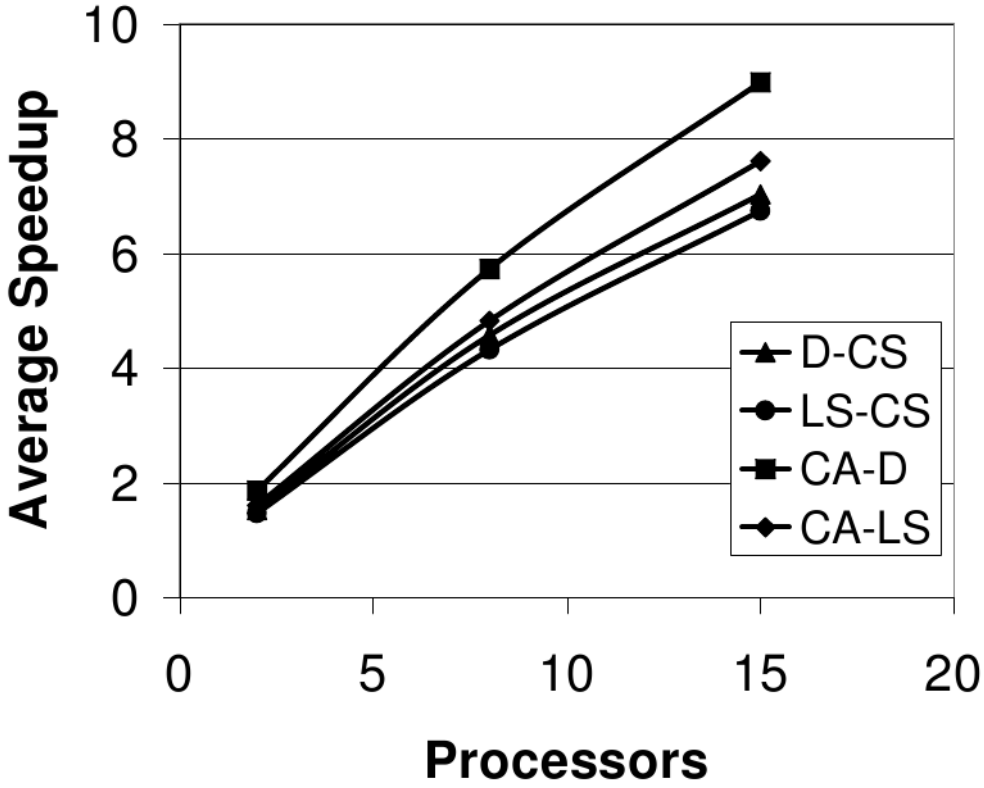
Target systems:

- Star-network with bi-directional links (one port model), various number of processors

Experimental results

- Series-Parallel graphs (left) and Random graph (right)
  - Different sizes, densities, weights ...



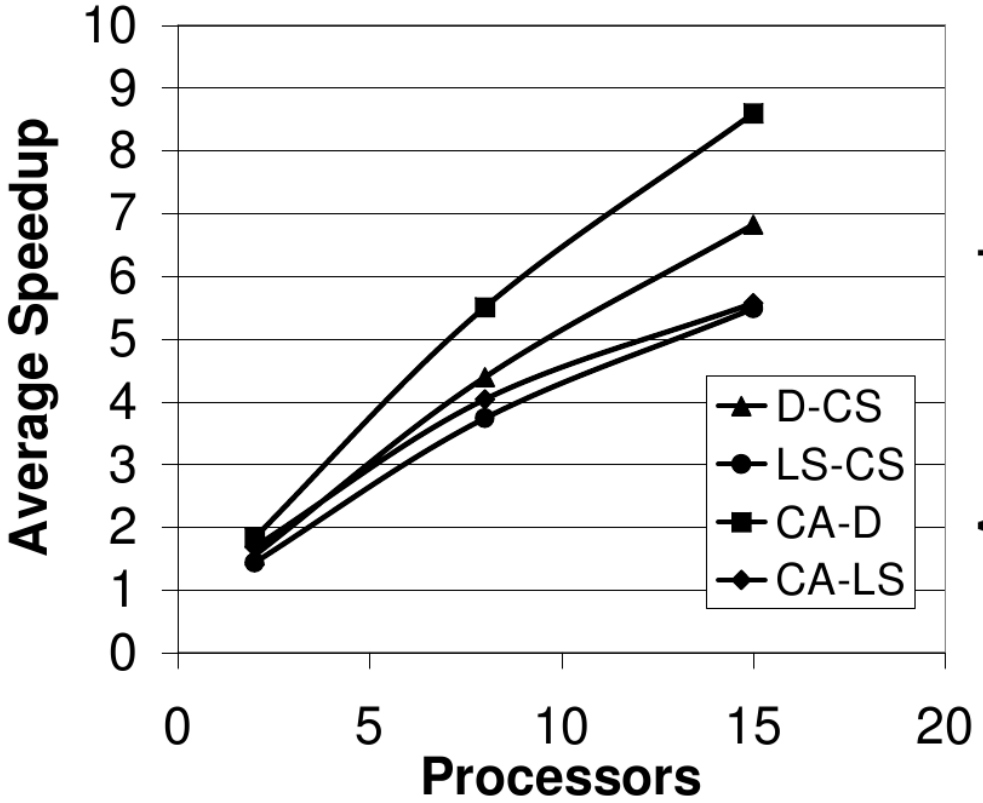


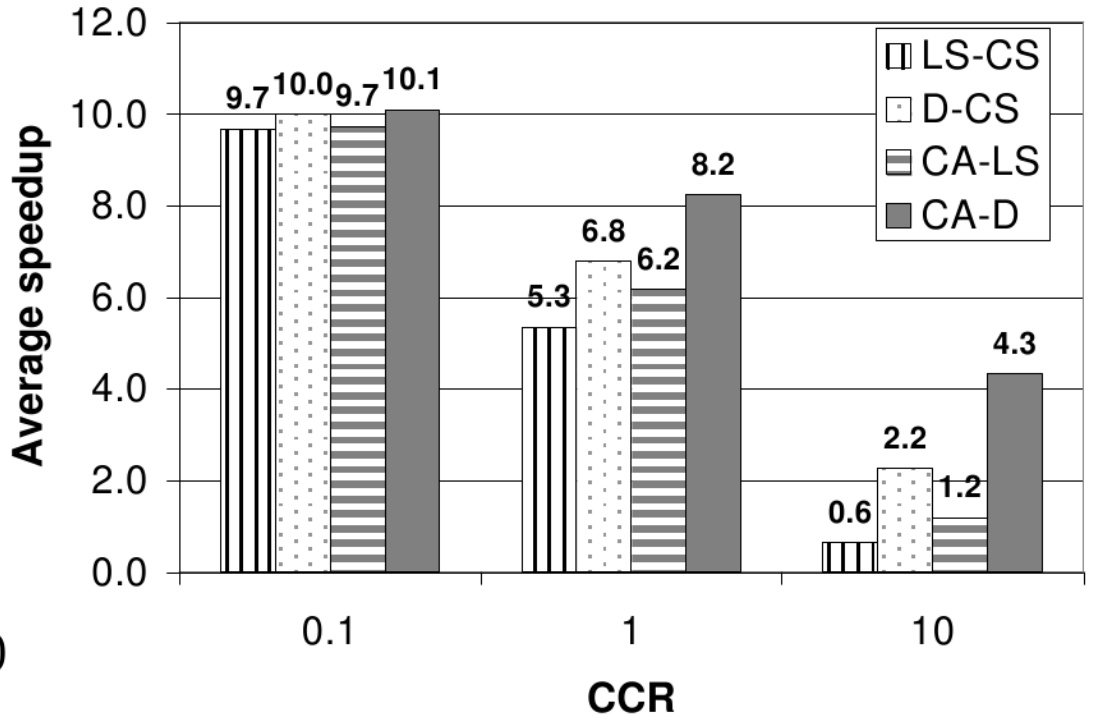
Oliver Sinnen – o.sinnen@auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND
Te Whare Wānanga o Tāmaki Makaurau

Experimental results

- Out-trees (left) and all graphs on 15 processors (right)





Conclusions

- Contention-aware scheduling
  - Easy to integrate
- Considering processor involvement
  - More complex

=> very good results

- Investigated duplication for contention-aware scheduling

=> Very good speedup improvements

=> as expected, even more than under classic model

www.ece.auckland.ac.nz/~sinnen/