Steady-State Scheduling (2)

Frédéric Vivien

e-mail: Frederic.Vivien@ens-lyon.fr

9 octobre 2006

Overview

1. Packet routing without fixed path
2. Broadcast
3. Master-slave tasking

Steady-State Scheduling

Changing the objective:

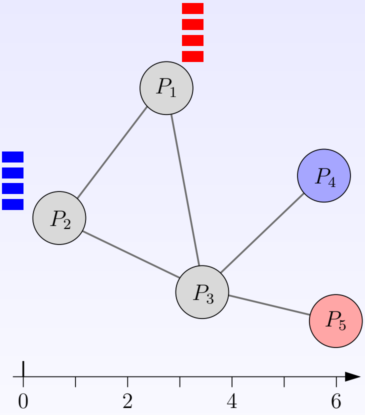
- Makespan minimization: reasonable for small set of tasks
- On distributed heterogeneous platforms: large amount of work
- No difference if program runs for 3 hours or 3 hours + 5 secondes
- Total completion time may not be the right metric
- Efficient resource utilization during steady-state:

  throughput maximization
- Neglect initialization and clean-up phases

Overview

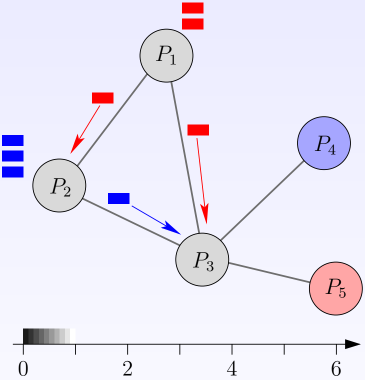
1. Packet routing without fixed path
2. Broadcast
3. Master-slave tasking

Packet routing without fixed path



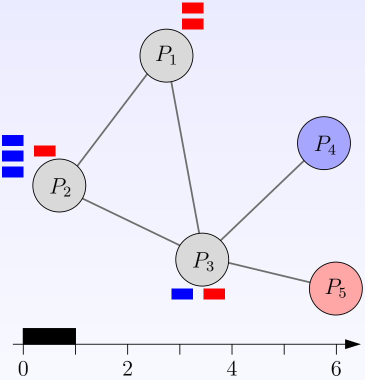
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l

Packet routing without fixed path



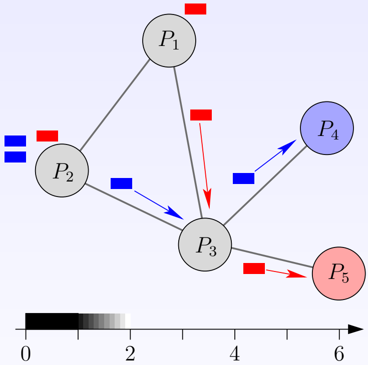
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



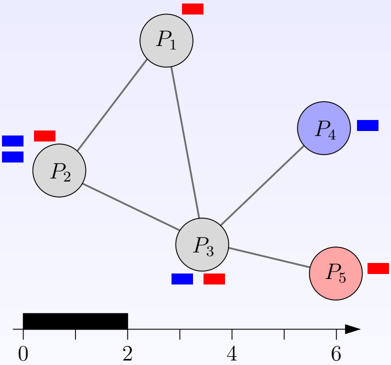
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



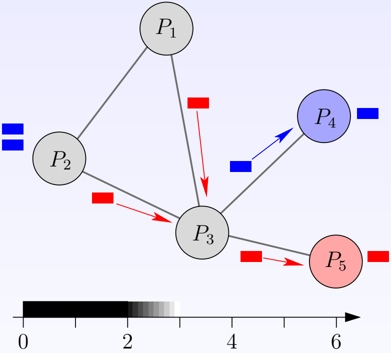
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



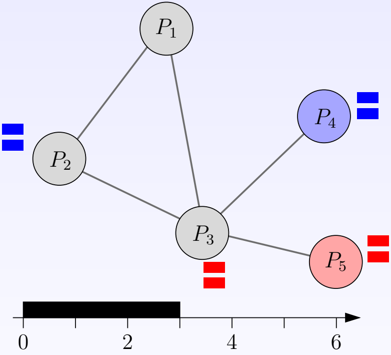
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



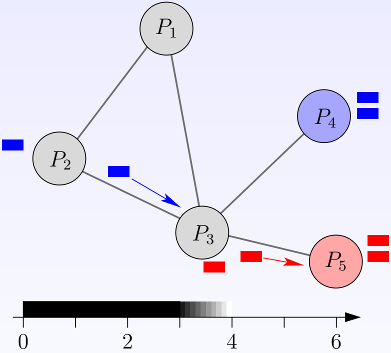
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



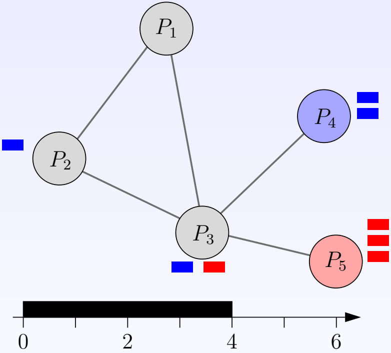
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



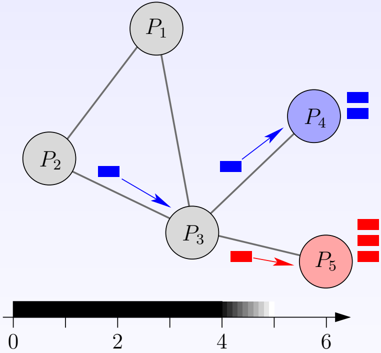
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



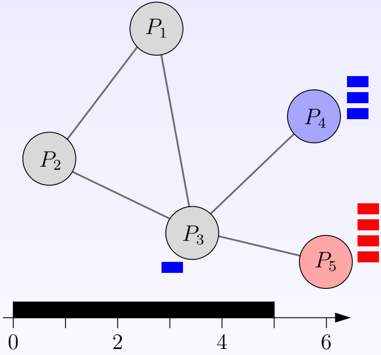
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



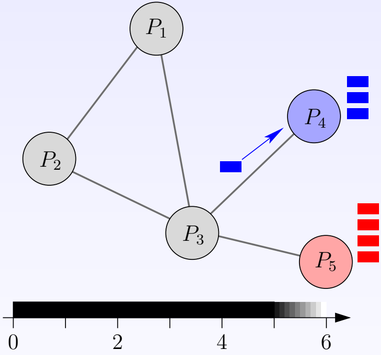
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



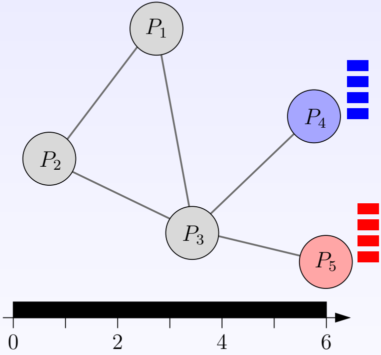
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



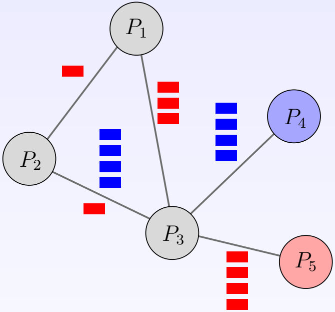
- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

Packet routing without fixed path



- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

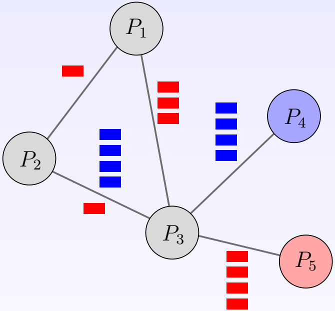
Packet routing without fixed path



- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

- $n^{k,l}_{i,j}$: total number of packets routed from k to l and crossing edge (i,j)

Packet routing without fixed path



- n_c collections of packets to be routed
- packets of a same collection may follow different paths
- $n^{k,l}$: total number of packets to be routed from k to l
- rule: one edge cannot carry two packets at the same time

- $n^{k,l}_{i,j}$: total number of packets routed from k to l and crossing edge (i,j)
- Congestion:

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n^{k,l}_{i,j} \qquad C_{\max} = \max_{i,j} C_{i,j}$$

Equations (1/2)

1. Initialization

$$\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$$

Equations (1/2)

1. Initialization

$$\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$$

2. Reception

$$\sum_{i|(i,l)\in A} n_{i,l}^{k,l} = n^{k,l}$$

Equations (1/2)

1. Initialization

$$\sum_{j|(k,j)\in A} n_{k,j}^{k,l} = n^{k,l}$$

2. Reception

$$\sum_{i|(i,l)\in A} n_{i,l}^{k,l} = n^{k,l}$$

3. Conservation law

$$\sum_{i|(i,j)\in A} n_{i,j}^{k,l} = \sum_{i|(j,i)\in A} n_{j,i}^{k,l} \quad \forall (k,l), j \neq k, j \neq l$$

Equations (2/2)

4. Congestion

$$C_{i,j} = \sum_{(k,l) | n^{k,l} > 0} n_{i,j}^{k,l}$$

Equations (2/2)

4. Congestion

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l}$$

5. Objective function

$$C_{\max} \geq C_{i,j}, \qquad \forall i, j$$

Minimize $C_{\max}$

Equations (2/2)

4. Congestion

$$C_{i,j} = \sum_{(k,l)|n^{k,l}>0} n_{i,j}^{k,l}$$

5. Objective function

$$C_{\max} \geq C_{i,j}, \qquad \forall i, j$$

Minimize $C_{\max}$

Linear program in rational numbers: polynomial-time solution. In practice use Maple, Mupad, lp-solve,. . .

Equations (2/2)

4. Congestion

$$C_{i,j} = \sum_{(k,l) | n^{k,l} > 0} n_{i,j}^{k,l}$$

5. Objective function

$$C_{\max} \geq C_{i,j}, \qquad \forall i, j$$

Minimize $C_{\max}$

Linear program in rational numbers: polynomial-time solution. In practice use Maple, Mupad, lp-solve,. . .

Solution:
number of messages $n_{i,j}^{k,l}$ of each edge to minimize total congestion

Routing algorithm

1. Computing optimal solution $C_{\max}$ of previous linear program

Routing algorithm

1. Computing optimal solution $C_{\max}$ of previous linear program
2. Consider periods of length Ω (to be defined later)

Routing algorithm

1. Computing optimal solution $C_{\max}$ of previous linear program
2. Consider periods of length Ω (to be defined later)
3. During each time-interval $[p\Omega, (p+1)\Omega]$, follow the optimal solution: edge (i,j) forwards:

$$m_{i,j}^{k,l} = \left\lfloor \frac{n_{i,j}^{k,l}\Omega}{C_{\max}} \right\rfloor$$

packets that go from k to l.
(if available)

Routing algorithm

1. Computing optimal solution $C_{\max}$ of previous linear program
2. Consider periods of length Ω (to be defined later)
3. During each time-interval $[p\Omega, (p+1)\Omega]$, follow the optimal solution: edge (i, j) forwards:

   $$m_{i,j}^{k,l} = \left\lfloor \frac{n_{i,j}^{k,l}\Omega}{C_{\max}} \right\rfloor$$

   packets that go from k to l.
   (if available)

4. number of such periods: $\left\lceil \frac{C_{\max}}{\Omega} \right\rceil$

Routing algorithm

1. Computing optimal solution $C_{\max}$ of previous linear program
2. Consider periods of length Ω (to be defined later)
3. During each time-interval $[p\Omega, (p+1)\Omega]$, follow the optimal solution: edge (i,j) forwards:

   $$m_{i,j}^{k,l} = \left\lfloor \frac{n_{i,j}^{k,l}\Omega}{C_{\max}} \right\rfloor$$

   packets that go from k to l.
   (if available)
4. number of such periods: $\left\lceil \frac{C_{\max}}{\Omega} \right\rceil$
5. After time-step

   $$T \equiv \left\lceil \frac{C_{\max}}{\Omega} \right\rceil \Omega \leq C_{\max} + \Omega$$

   sequentially process M residual packets in no longer than ML time-steps, where L is the maximum length of a simple path in the network

Feasibility

$$\sum_{(k,l)} m_{i,j}^{k,l} \leq \sum_{(k,l)} \frac{n_{i,j}^{k,l}\Omega}{C_{\max}} = \frac{C_{i,j}\Omega}{C_{\max}} \leq \Omega$$

Makespan

- Define Ω as $\Omega = \sqrt{C_{\max} n_c}$.

Makespan

- Define Ω as $\Omega = \sqrt{C_{\max} n_c}$.
- Total number of packets still inside network at time-step T is at most
$$2|A|\sqrt{C_{\max} n_c} + |A| n_c$$

Makespan

- Define Ω as $\Omega = \sqrt{C_{\max} n_c}$.
- Total number of packets still inside network at time-step T is at most
$$2|A|\sqrt{C_{\max} n_c} + |A| n_c$$
- Makespan:
$$C_{\max} \leq C^* \leq C_{\max} + \sqrt{C_{\max} n_c} + 2|A|\sqrt{C_{\max} n_c}|V| + |A| n_c |V|$$
$$C^* = C_{\max} + O(\sqrt{C_{\max}})$$

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering/allocation of tasks/messages not needed

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering/allocation of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering/allocation of tasks/messages not needed
- Characterize resource activity during each time-unit:
  - which (rational) fraction of time is spent computing for which application?
  - which (rational) fraction of time is spent receiving or sending to which neighbor?

Efficiency Periodic schedule, described in compact form

Steady-state scheduling

Background Approach pioneered by Bertsimas and Gamarnik

Rationale Maximize throughput

Simplicity Relaxation of makespan minimization problem

- Ignore initialization and clean-up phases
- Precise ordering/allocation of tasks/messages not needed
- Characterize resource activity during each time-unit:

  - which (rational) fraction of time is spent computing for which application?

  - which (rational) fraction of time is spent receiving or sending to which neighbor?

Efficiency Periodic schedule, described in compact form

Adaptability Dynamically record observed performance during current period, and inject this information to compute optimal schedule for next period

Overview

1. Packet routing without fixed path
2. Broadcast
3. Master-slave tasking

Broadcasting data

- Key collective communication operation
- Start: one processor has the data
- End: all processors own a copy

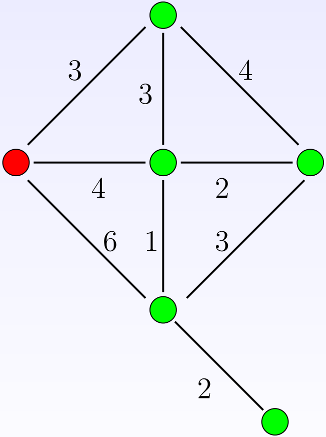
Broadcasting data

- Key collective communication operation
- Start: one processor has the data
- End: all processors own a copy
- Vast literature about broadcast, `MPI_Bcast`
- Standard approach: use a spanning tree

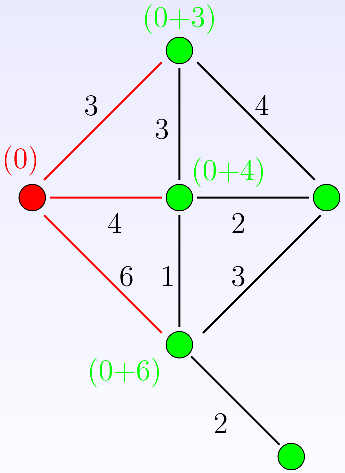
Broadcasting data

- Key collective communication operation
- Start: one processor has the data
- End: all processors own a copy
- Vast literature about broadcast, `MPI_Bcast`
- Standard approach: use a spanning tree
- Finding the best spanning tree: NP-Complete problem (even in the telephone model)

Heuristic: Earliest completing edge first (ECEF)

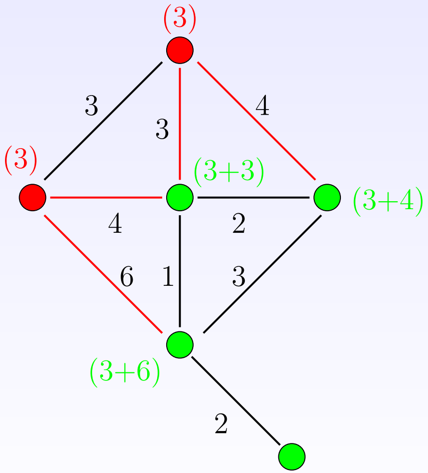


Heuristic: Earliest completing edge first (ECEF)



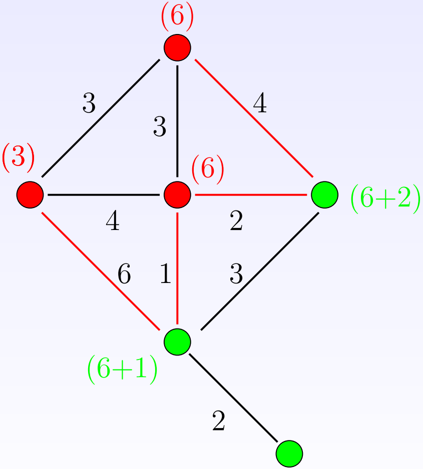
Next node: minimize $(R_i) + c_{ij}$, $P_j \notin \mathcal{T}$

Heuristic: Earliest completing edge first (ECEF)



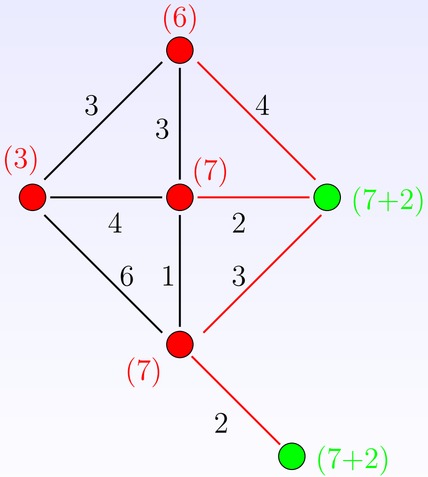
Next node: minimize $(R_i) + c_{ij}$, $P_j \notin \mathcal{T}$

Heuristic: Earliest completing edge first (ECEF)



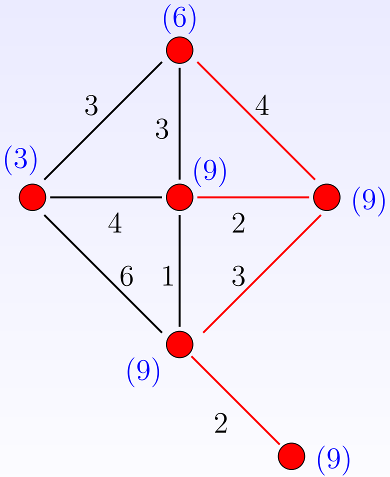
Next node: minimize $(R_i) + c_{ij}$, $P_j \notin \mathcal{T}$

Heuristic: Earliest completing edge first (ECEF)



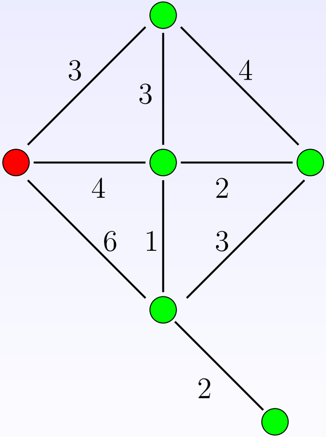
Next node: minimize $(R_i) + c_{ij}$, $P_j \notin \mathcal{T}$

Heuristic: Earliest completing edge first (ECEF)



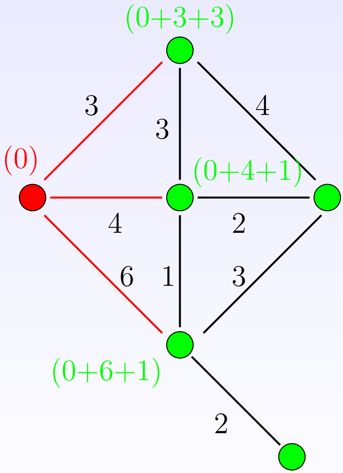
Broadcast finishing times (t)

Heuristic: Look-ahead (LA)



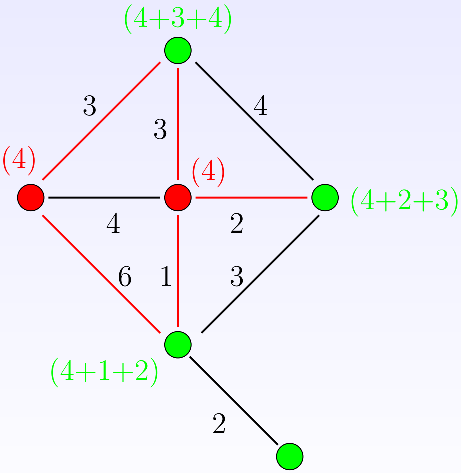
Next node: minimize $\color{red}{(R_i)} + c_{ij} + \color{green}{(\min c_{jk})}$, $P_j, P_k \notin \mathcal{T}$

Heuristic: Look-ahead (LA)



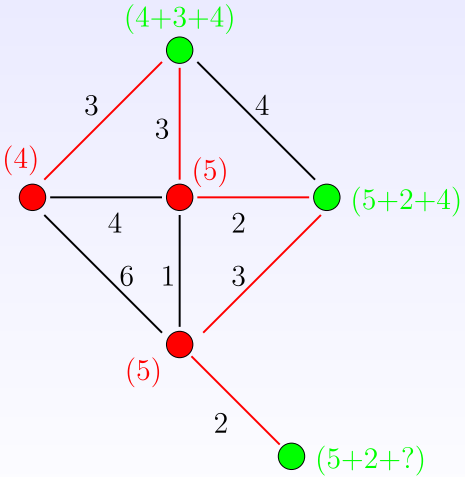
Next node: minimize $(R_i) + c_{ij} + (\min c_{jk})$, $P_j, P_k \notin \mathcal{T}$

Heuristic: Look-ahead (LA)



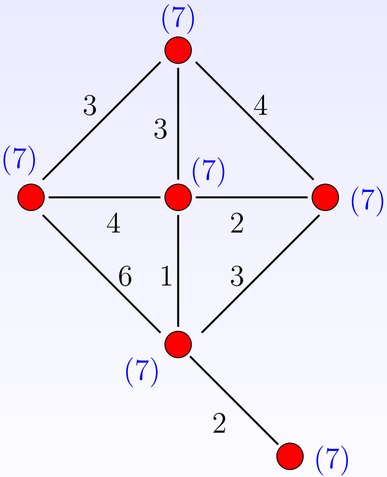
Next node: minimize $(R_i) + c_{ij} + (\min c_{jk})$, $P_j, P_k \notin \mathcal{T}$

Heuristic: Look-ahead (LA)



Next node: minimize $(R_i) + c_{ij} + (\min c_{jk})$, $P_j, P_k \notin \mathcal{T}$

Heuristic: Look-ahead (LA)



Broadcast finishing times (t)

Broadcasting longer messages

- Message size goes from L to, say, $10L$
- Communication costs scale from c_{ij} to $10c_{ij}$

Broadcasting longer messages

- Message size goes from L to, say, $10L$
- Communication costs scale from c_{ij} to $10c_{ij}$
- ECEF heuristic: broadcast time becomes 90
- LA heuristic: broadcast time becomes 70

Broadcasting longer messages

- ~~Message size goes from L to, say, $10L$~~
- ~~Communication costs scale from c_{ij} to $10c_{ij}$~~
- ~~ECEF heuristic: broadcast time becomes 90~~
- ~~LA heuristic: broadcast time becomes 70~~

Eh wait!

What about PIPELINING?!

Introduction to Pipelined Communications

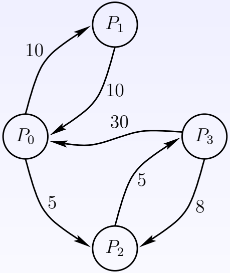
- Complex applications on gridS require collective communication schemes:

one-to-all	Broadcast, Multicast, Scatter
all-to -one	Reduce
all-to-all	Gossip, All-to-All

- Numerous studies of a single communication scheme, mainly about one single broadcast
- Pipelining communications:
  - data parallelism involves a large amount of data
  - not a single communication, but a series of same communication schemes (e.g. a series of broadcasts from the same source)
  - maximize throughput of the steady-state operation

Modeling the platform

- $G = (P, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- $(P_j, P_k) \in E$ denotes a communication link between P_i and P_j
- $c(P_j, P_k)$ denotes the time to transfer one unit-size message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



Pipelining Broadcasts

- Send n messages from P_0 to all other P_i's

Pipelining Broadcasts

- Send n messages from P_0 to all other P_i's
- Let $T_{opt(n)}$ denote the optimal time for broadcasting the n messages

Pipelining Broadcasts

- Send n messages from P_0 to all other P_i's
- Let $T_{opt(n)}$ denote the optimal time for broadcasting the n messages
- Asymptotic optimality: $$\lim_{n \to +\infty} \frac{T_{alg}(n)}{T_{opt}(n)} = 1$$

Pipelining Broadcasts

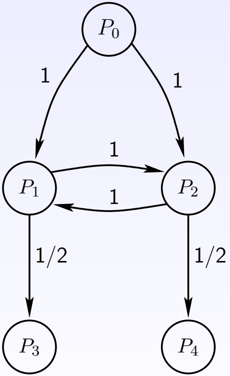
- Send n messages from P_0 to all other P_i's
- Let $T_{opt(n)}$ denote the optimal time for broadcasting the n messages
- Asymptotic optimality: $$\lim_{n \to +\infty} \frac{T_{alg}(n)}{T_{opt}(n)} = 1$$
- Usually, broadcasts are executed along one or several spanning trees

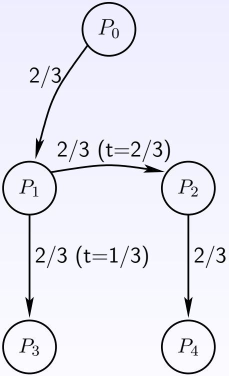
Pipelining Broadcasts

- Send n messages from P_0 to all other P_i's
- Let $T_{opt(n)}$ denote the optimal time for broadcasting the n messages
- Asymptotic optimality: $$\lim_{n \to +\infty} \frac{T_{alg}(n)}{T_{opt}(n)} = 1$$
- Usually, broadcasts are executed along one or several spanning trees
- What is the best broadcast throughput when using a single tree, a DAG, or a general graph?

With a tree

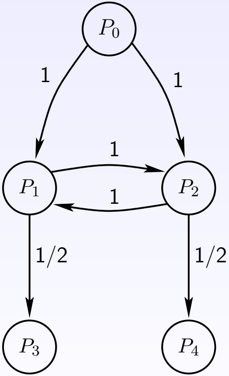
The throughput with the best tree is 2 messages every 3 tops

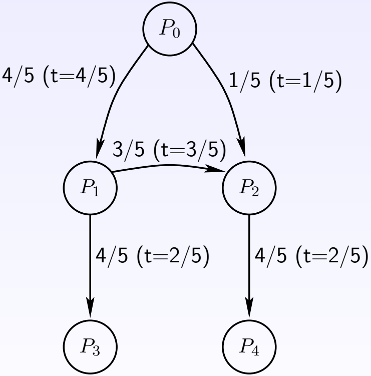




With a DAG

The throughput with the best DAG is 4 messages every 5 tops

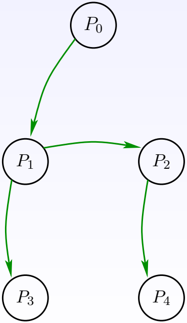




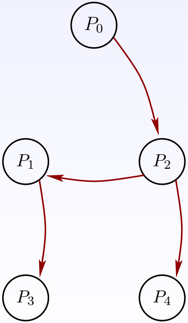
With a general graph

- Throughput with the best graph: 2 messages every 2 tops
- Two different sorts of messages (even/odd numbered)
- $m_1(i)$ denotes the message sent from P_0 to P_1 during period i
- $m_2(i)$ denotes the message sent from P_0 to P_2 during period i

path for m_1 messages



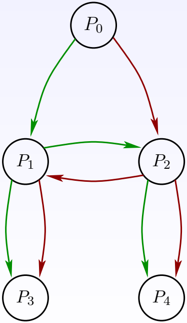
path for m_2 messages



With a general graph

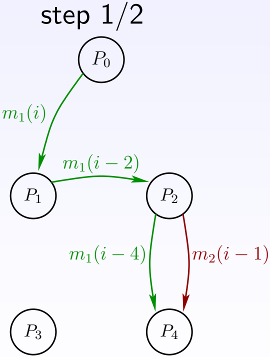
- Throughput with the best graph: 2 messages every 2 tops
- Two different sorts of messages (even/odd numbered)
- $m_1(i)$ denotes the message sent from P_0 to P_1 during period i
- $m_2(i)$ denotes the message sent from P_0 to P_2 during period i

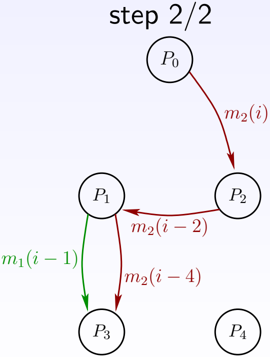
all communications



With a general graph

- Throughput with the best graph: 2 messages every 2 tops
- Two different sorts of messages (even/odd numbered)
- $m_1(i)$ denotes the message sent from P_0 to P_1 during period i
- $m_2(i)$ denotes the message sent from P_0 to P_2 during period i





Problem Statement

- Input: $G = (P, E, c)$
- Output:
  - The best throughput $\frac{p}{q}$
  - A “compact” description of the behavior of the nodes.

During q time steps

- step 1: $P_{i_1}^{(1)}$ sends 1 mess to $P_{j_1}^{(1)}$
- step 1: $P_{i_2}^{(1)}$ sends 1 mess to $P_{j_2}^{(1)}$
- $\vdots$
- step q: $P_{i_n}^{(q)}$ sends 1 mess to $P_{j_n}^{(q)}$

This is not likely to be polynomial since the size of the description is a priori of order $O(nq)$

During q time steps

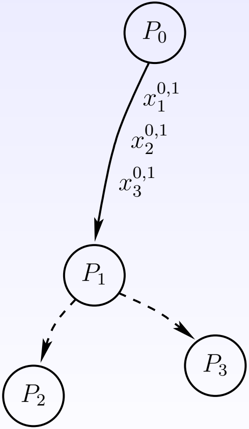
- step 1: $P_{i_1}^{(1)}$ sends $\alpha_{i_1}^{(1)}$ mess to $P_{j_1}^{(1)}$
- step 1: $P_{i_2}^{(1)}$ sends $\alpha_{i_2}^{(1)}$ mess to $P_{j_2}^{(1)}$
- $\vdots$
- step q: $P_{i_n}^{(q)}$ sends $\alpha_{i_n}^{(q)}$ mess to $P_{j_n}^{(q)}$

The size of such a description may be polynomial

Broadcast: Linear Program (1)

$x_i^{j,k}$ denotes the fraction of the message from P_0 to P_i that uses edge (P_j, P_k)
The conditions are

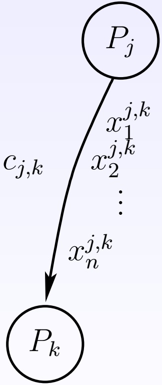
- $\forall i, \quad \sum x_i^{0,k} = 1$
- $\forall i, \quad \sum x_i^{j,i} = 1$
- $\forall j \neq 0, i, \quad \sum_k x_i^{j,k} = \sum_k x_i^{k,j}$



Broadcast: Linear Program (2)

$t_{j,k}$ denotes the time to transfer all the messages between P_j and P_k

- $t_{j,k} \leq \sum x_i^{j,k} c_{j,k}$????
- too pessimistic since $x_{i_1}^{j,k}$ and $x_{i_2}^{k,j}$ may be the same message
- not good for a lower bound



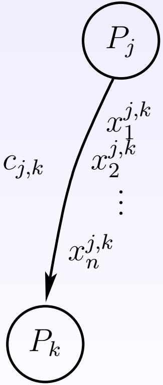
Broadcast: Linear Program (2)

$t_{j,k}$ denotes the time to transfer all the messages between P_j and P_k

- $t_{j,k} \;\leq\; \sum x_i^{j,k} c_{j,k}$????
- too pessimistic since $x_{i_1}^{j,k}$ and $x_{i_2}^{k,j}$ may be the same message
- not good for a lower bound

or

- $\forall i, \;\; t_{j,k} \geq x_i^{j,k} c_{j,k}$????
- too optimistic since it supposes that all the messages are sub-messages of the largest one
- OK for a lower bound, may not be feasible



Broadcast: Linear Program (3)

one-port model, for a unit message

- at most one sending operation: $\displaystyle\sum_{(P_j,P_k)\in E} t_{j,k} \leq t_j^{out}$
- at most one receiving operation: $\displaystyle\sum_{(P_k,P_j)\in E} t_{k,j} \leq t_j^{in}$

and at last,

- $\forall j, \quad t_j^{out} \leq t^{broadcast}$
- $\forall j, \quad t_j^{in} \leq t^{broadcast}$

Broadcast: Linear Program (4)

$$
\text{MINIMIZE } t^{broadcast},\\
\text{SUBJECT TO}\\
\begin{cases}
\forall i, & \sum x_i^{0,k} = 1\\
\forall i, & \sum x_i^{j,i} = 1\\
\forall i,\ \forall j \neq 0, i, & \sum x_i^{j,k} = \sum x_i^{k,j}\\
\forall i, j, k & t_{j,k} \geq x_i^{j,k} c_{j,k}\\
\forall j, & \sum_{(P_j,P_k)\in E} t_{j,k} \leq t_j^{out}\\
\forall j, & \sum_{(P_k,P_j)\in E} t_{k,j} \leq t_j^{in}\\
\forall j, & t_j^{out} \leq t^{broadcast}\\
\forall j, & t_j^{in} \leq t^{broadcast}
\end{cases}
$$

Caveats

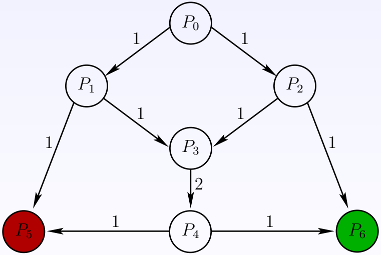
- The linear program provides a lower bound for the broadcasting time of a unit-size divisible message
- It is not obvious that this lower bound is feasible since we considered that all the messages using the same communication link are sub-messages of the largest one.

Consider the multicast of a message:

- Some nodes not involved in receiving the messages
- Ue the same equations, but if P_i does not belong to the multicast set, then $\sum x_i^{0,k} = 1$ and $\sum x_i^{j,i} = 1$ are removed

Lower Bound ??? Multicast Example (1)

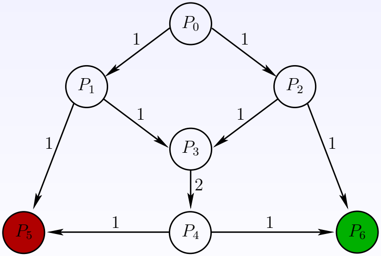
Consider the following platform, where the multicast set consists in the colored nodes:



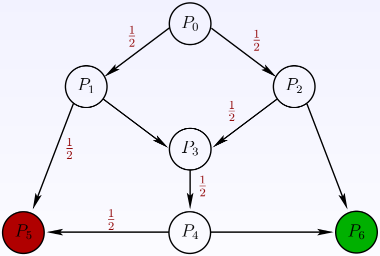
The linear program provides the following solution with throughput 1:

Lower Bound ??? Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

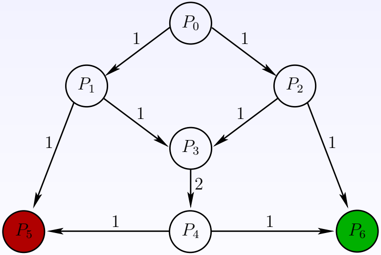


The linear program provides the following solution with throughput 1:

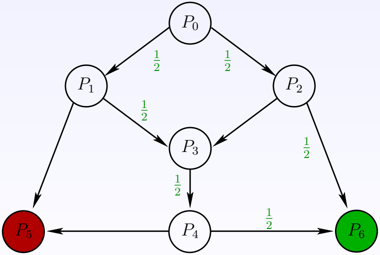


Lower Bound ??? Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

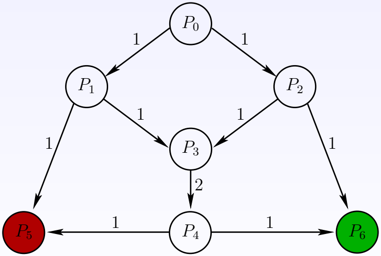


The linear program provides the following solution with throughput 1:

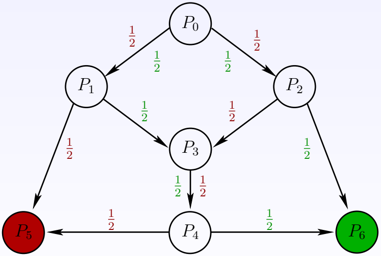


Lower Bound ??? Multicast Example (1)

Consider the following platform, where the multicast set consists in the colored nodes:

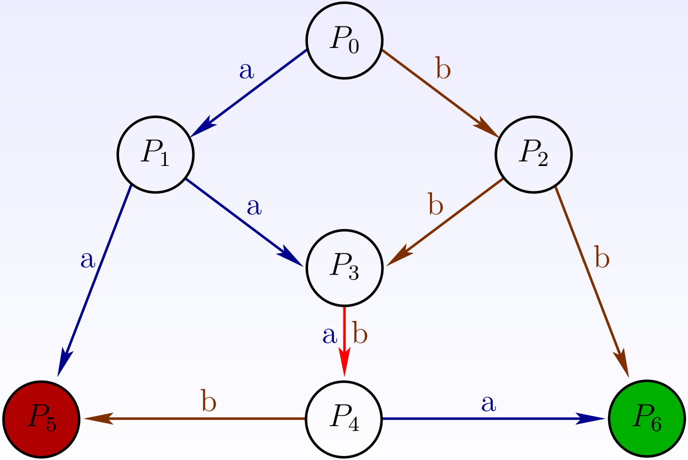


The linear program provides the following solution with throughput 1:



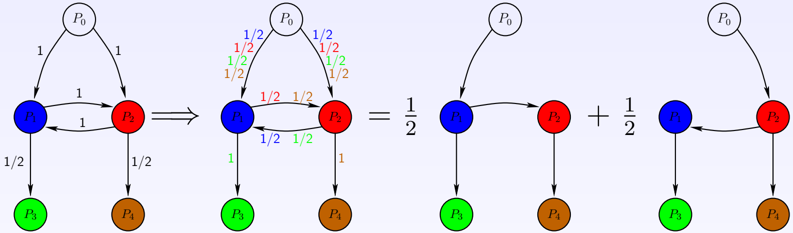
Lower Bound ??? Multicast Example (2)

Nevertheless, the obtained throughput is not feasible:



Lower Bound ??? Broadcast Example

For broadcast, the bound is nevertheless tight:



2 disjoint broadcast trees T_1 and T_2, of weight $\frac{1}{2}$ $\Longrightarrow$ 1 message broacast at every top.

- How to find the trees ?
- How to keep the number of (weighted) trees relatively low ?

How many paths from P_0 to P_i (1)

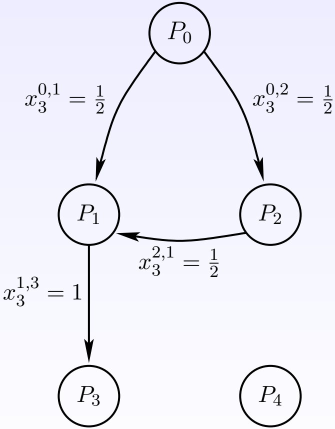
$x_i^{j,k}$ denotes the fraction of the message from P_0 to P_i that uses edge (P_j, P_k)
We know that

$$\left\{\begin{array}{ll} \text{fraction of messages leaving } P_0 & \sum x_i^{0,k} = 1 \\ \text{fraction of messages arriving at } P_i & \sum x_i^{j,i} = 1 \\ \text{conservation law at } P_i \neq P_0,\ P_i & \sum x_i^{j,k} = \sum x_i^{k,j} \end{array}\right.$$

The x_i's define a flow in G of total weight 1.

How many paths from P_0 to P_i (2)

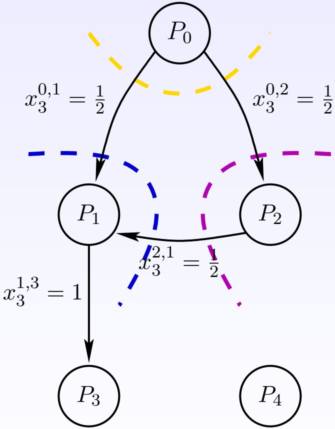
- The x_3's define a flow in G of total weight 1
- In order to disconnect P_3 from P_0, a total weight of 1 has to be removed



P_0
$x_3^{0,1} = \frac{1}{2}$
$x_3^{0,2} = \frac{1}{2}$
P_1
P_2
$x_3^{2,1} = \frac{1}{2}$
$x_3^{1,3} = 1$
P_3
P_4

How many paths from P_0 to P_i (2)

- The x_3's define a flow in G of total weight 1
- In order to disconnect P_3 from P_0, a total weight of 1 has to be removed


P_0
$x_3^{0,1} = \frac{1}{2}$
$x_3^{0,2} = \frac{1}{2}$
P_1
P_2
$x_3^{2,1} = \frac{1}{2}$
$x_3^{1,3} = 1$
P_3
P_4

A nice graph theorem

- $c(P_0, P_i)$ minimum weight to remove to disconnect $= 1$
- $c(P_0) = \min c(P_0, P_i) = 1$
- $n_{j,k}{=}\max_i \left\{ x_i^{j,k} \right\}$ is the fraction of messages through (P_j, P_k).

Theorem (Weighted version of Edmond's branching Theorem)

Given a directed weighted $G = (P, E, n)$, $P_0 \in P$ the source, we can find P_0-arborescences, $T_1, \ldots, T_k$, and weights $\lambda_1, \ldots, \lambda_k$ with $\forall j, k,\ \sum \lambda_i \delta(T_i) \leq n_{j,k}$ with

$$\sum \lambda_i = c(P_0) = 1,$$

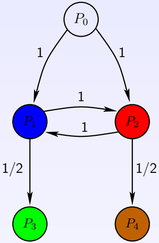
in strongly polynomial time, and $k \leq |E| + |V|^3$.

This theorem provides:

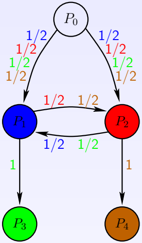
- the set of trees, their weights
- and the number of trees is "low": $\leq |E| + |V|^3$.

A nice graph theorem (2)

1. Linear program:

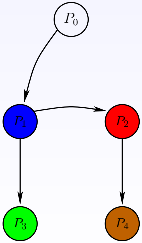


$\Longrightarrow$

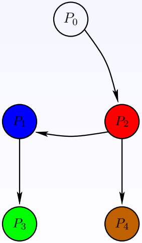


2. Schrijver's algorithm for weighted Edmond's theorem

$\Longrightarrow \frac{1}{2}$



$+ \quad \frac{1}{2}$



Compact description of the solution?

- Period duration $= 2$ $(=$ lcm(denominators tree coeff.)$)$

Compact description of the solution?

- Period duration $= 2$ $(=$ lcm(denominators tree coeff.)$)$
- P_0 sends even-numbered messages to P_1 and odd-numbered messages to P_2

Compact description of the solution?

- Period duration $= 2$ $(=$ lcm(denominators tree coeff.)$)$
- P_0 sends even-numbered messages to P_1 and odd-numbered messages to P_2
- Complete description for time-steps $2i$ and $2i+1$:
  - P_0 sends m_{2i} to P_1 and m_{2i+1} to P_2
  - P_1 sends m_{2i-2} (recvd. from P_0 at previous step) to P_2 and P_3
  - P_1 sends m_{2i-3} (recvd. from P_2 at previous step) to P_3
  - P_2 sends m_{2i-1} (recvd. from P_0 at previous step) to P_1 and P_4
  - P_2 sends m_{2i-4} (recvd. from P_1 at previous step) to P_4

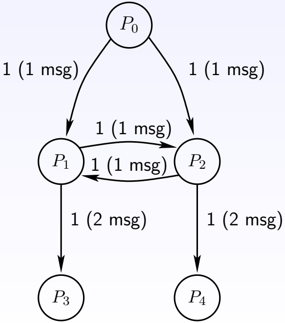
Compact description of the solution?

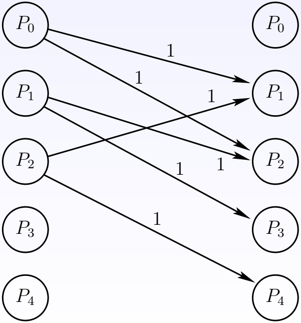
- Period duration $= 2$ $(=$ lcm(denominators tree coeff.)$)$
- P_0 sends even-numbered messages to P_1 and odd-numbered messages to P_2
- Complete description for time-steps $2i$ and $2i+1$:
  - P_0 sends m_{2i} to P_1 and m_{2i+1} to P_2
  - P_1 sends m_{2i-2} (recvd. from P_0 at previous step) to P_2 and P_3
  - P_1 sends m_{2i-3} (recvd. from P_2 at previous step) to P_3
  - P_2 sends m_{2i-1} (recvd. from P_0 at previous step) to P_1 and P_4
  - P_2 sends m_{2i-4} (recvd. from P_1 at previous step) to P_4
- Solution size: number of communications within one period bounded by:

$$\begin{array}{cc} \text{number of trees} & \leq |E| + |V|^3 \\ \times & \\ \text{number of edges of one tree} & \leq |V| \end{array}$$

From local to global (1)

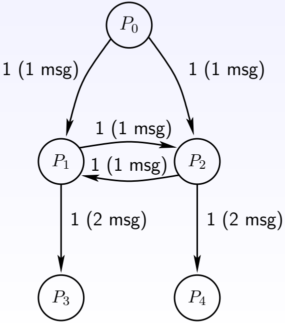
1. Set of communications to execute within period T
2. One-port equations → local constraints
3. Pairwise-disjoint communications to be scheduled simultaneously
   ⇒ extract a collection of matchings

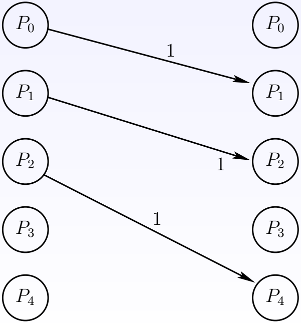




From local to global (1)

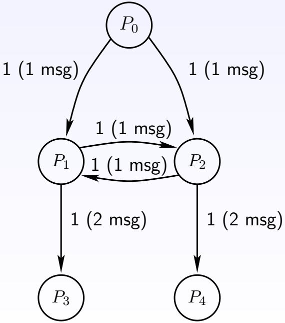
1. Set of communications to execute within period T
2. One-port equations → local constraints
3. Pairwise-disjoint communications to be scheduled simultaneously
   $\Rightarrow$ extract a collection of matchings

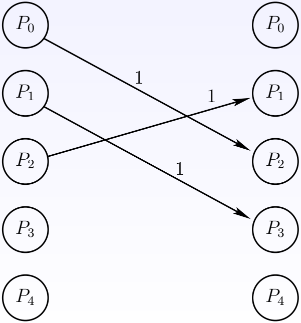




From local to global (1)

1. Set of communications to execute within period T
2. One-port equations → local constraints
3. Pairwise-disjoint communications to be scheduled simultaneously
   $\Rightarrow$ extract a collection of matchings





From local to global (2)

Solution

- Peel off bipartite communication graph

From local to global (2)

Solution

- Peel off bipartite communication graph
- **Idea 1:** Split each communication of length L into L communications of length 1 and use König's edge-coloring algorithm (but not polynomial)

From local to global (2)

Solution

- Peel off bipartite communication graph
- **Idea 1:** Split each communication of length L into L communications of length 1 and use König's edge-coloring algorithm (but not polynomial)
- **Idea 2:** Use Schrijver's weighted edge-coloring algorithm:

From local to global (2)

Solution

- Peel off bipartite communication graph
- **Idea 1:** Split each communication of length L into L communications of length 1 and use König's edge-coloring algorithm (but not polynomial)
- **Idea 2:** Use Schrijver's weighted edge-coloring algorithm:
  - extract a matching and substract minimum weight from participating edges

From local to global (2)

Solution

- Peel off bipartite communication graph
- **Idea 1:** Split each communication of length L into L communications of length 1 and use König's edge-coloring algorithm (but not polynomial)
- **Idea 2:** Use Schrijver's weighted edge-coloring algorithm:
  - extract a matching and substract minimum weight from participating edges
  - zero out at least one edge for each matching

From local to global (2)

Solution

- Peel off bipartite communication graph
- **Idea 1:** Split each communication of length L into L communications of length 1 and use König's edge-coloring algorithm (but not polynomial)
- **Idea 2:** Use Schrijver's weighted edge-coloring algorithm:
  - extract a matching and substract minimum weight from participating edges
  - zero out at least one edge for each matching
  - strongly polynomial

Conclusion

Complexity of steady-state problems

Ask biased question:

Can we determine best throughput and characterize a solution achieving it, all that in polynomial time?

1. Broadcast: yes
2. Multicast: no, NP-complete
3. Scatter: yes (easier)
4. Reduce: yes (complicated too)

Makespan minimization versus throughput

Everything NP-hard.

Bibliography – Broadcast

- Complexity:
  On broadcasting in heterogeneous networks, S. Khuller and Y.A. Kim, 15th ACM SODA (2004), 1011–1020
- Heuristics:
  Efficient collective communication in distributed heterogeneous systems, P.B. Bhat, C.S. Raghavendra and V.K. Prasanna, JPDC 63 (2003), 251–263
- Steady-state:
  Pipelining broadcasts on heterogeneous platforms, O. Beaumont et al., IEEE TPDS 16, 4 (2005), 300-313

Overview

1. Packet routing without fixed path
2. Broadcast
3. Master-slave tasking

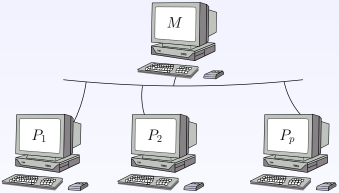
Master-slave platform

Master-slave tasking Simple yet efficient

Master-slave platform

Master-slave tasking Simple yet efficient

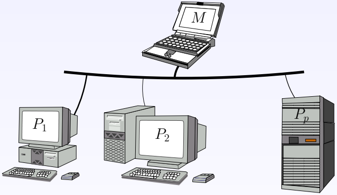
Standard implementation Independent tasks are executed by identical processors (the slaves) under the supervision of a special processor (the master)



Master-slave platform

Master-slave tasking Simple yet efficient

Standard implementation Independent tasks are executed by identical processors (the slaves) under the supervision of a special processor (the master)



Heterogeneous version Computing times and communication times are different from slave to slave

Model

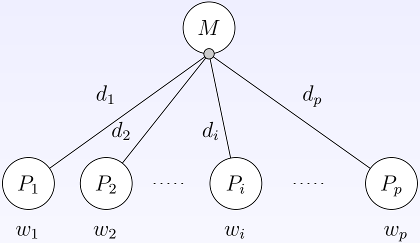
- Set of independent tasks to be executed by p slaves

Model

- Set of independent tasks to be executed by p slaves
- All tasks are identical: each represents the same amount of computations

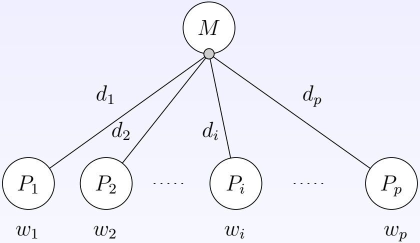
Model

- Set of independent tasks to be executed by p slaves
- All tasks are identical: each represents the same amount of computations



Model

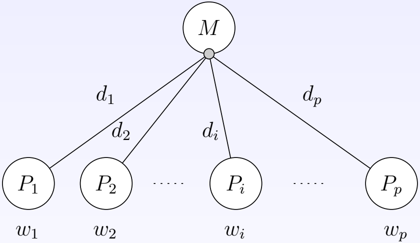
- Set of independent tasks to be executed by p slaves
- All tasks are identical: each represents the same amount of computations



- Need d_i time-units to transfer a task from M to P_i, and w_i time-units to execute it on P_i

Model

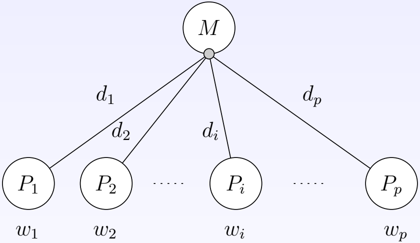
- Set of independent tasks to be executed by p slaves
- All tasks are identical: each represents the same amount of computations



- Need d_i time-units to transfer a task from M to P_i, and w_i time-units to execute it on P_i
- Communications obey the one-port model: M can only send one task at a given time-step

Model

- Set of independent tasks to be executed by p slaves
- All tasks are identical: each represents the same amount of computations



- Need d_i time-units to transfer a task from M to P_i, and w_i time-units to execute it on P_i
- Communications obey the one-port model: M can only send one task at a given time-step
- Overlap computations and communications

Complexity results

Definition MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$): Given a master-slave platform with parameters $(d_1, w_1), \ldots, (d_p, w_p)$, what it the minimum time to process n tasks?

Complexity results

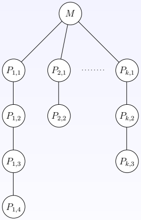
Definition MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$): Given a master-slave platform with parameters $(d_1, w_1), \ldots, (d_p, w_p)$, what it the minimum time to process n tasks?

MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$) can be solved at cost $O(n^2 p^2)$ by a complicated greedy algorithm

Complexity results

Definition MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$): Given a master-slave platform with parameters $(d_1, w_1), \ldots, (d_p, w_p)$, what it the minimum time to process n tasks?

MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$) can be solved at cost $O(n^2p^2)$ by a complicated greedy algorithm

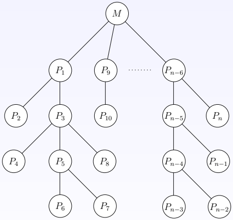


If the interconnection network is a linear chain or a harpoon, problem still polynomial

Complexity results

Definition MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$): Given a master-slave platform with parameters $(d_1, w_1), \ldots, (d_p, w_p)$, what it the minimum time to process n tasks?

MasterSlave($P_1(d_1, w_1), \ldots, P_p(d_p, w_p), T^{(1)}, \ldots, T^{(n)}$) can be solved at cost $O(n^2p^2)$ by a complicated greedy algorithm



If the interconnection network is a linear chain or a harpoon, problem still polynomial
However, for tree-shaped platforms, problem becomes NP-complete

A model not well-suited. . .

- Hardness comes from the metric: makespan minimization

A model not well-suited. . .

- Hardness comes from the metric: makespan minimization
- Not suited to large-scale distributed platforms

A model not well-suited. . .

- Hardness comes from the metric: makespan minimization
- Not suited to large-scale distributed platforms
  - Modeling a collection of clusters, and acquiring all various parameters: long, tedious and error-prone

A model not well-suited. . .

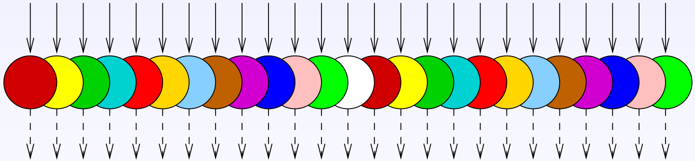
- Hardness comes from the metric: makespan minimization
- Not suited to large-scale distributed platforms
  - Modeling a collection of clusters, and acquiring all various parameters: long, tedious and error-prone
  - Given difficulty and time needed to deploy applications on such platforms, number of tasks expected to be very large

A model not well-suited. . .

- Hardness comes from the metric: makespan minimization
- Not suited to large-scale distributed platforms
  - Modeling a collection of clusters, and acquiring all various parameters: long, tedious and error-prone
  - Given difficulty and time needed to deploy applications on such platforms, number of tasks expected to be very large
- Concentrate on steady-state, and target complex platforms (with cycles and multiple paths) while designing efficient (asymptotically optimal) schedulings

Application graph

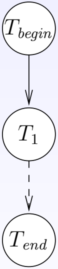
n problem instances $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$, where n is large



Application graph

n problem instances $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$, where n is large

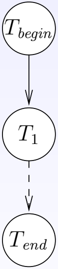
Each problem corresponds to a copy of the same task graph $G_A = (V_A, E_A)$, the application graph



Application graph

n problem instances $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \ldots, \mathcal{P}^{(n)}$, where n is large

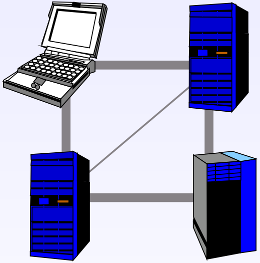
Each problem corresponds to a copy of the same task graph $G_A = (V_A, E_A)$, the application graph



T_{begin} et T_{end} are fictitious tasks, used to model the scattering of input files and the gathering of output files

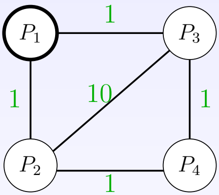
Platform graph

Target platform represented by platform graph $G_P = (V_P, E_P)$



Platform graph

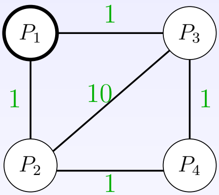
Target platform represented by platform graph $G_P = (V_P, E_P)$



Edge $P_i \rightarrow P_j$ is labeled with $c_{i,j}$: time needed to send a unit-length message from P_i to P_j

Platform graph

Target platform represented by platform graph $G_P = (V_P, E_P)$



Edge $P_i \to P_j$ is labeled with $c_{i,j}$: time needed to send a unit-length message from P_i to P_j

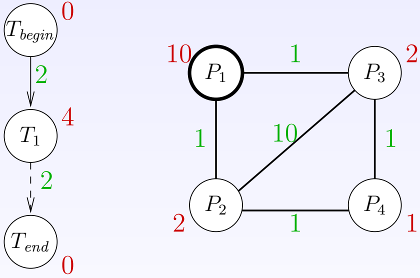
Communication model: full overlap, one-port for incoming and outgoing messages

Computations and communications

P_i requires $w_{i,k}$ time-units to process task T_k $(k \in \{begin, 1, end\})$.

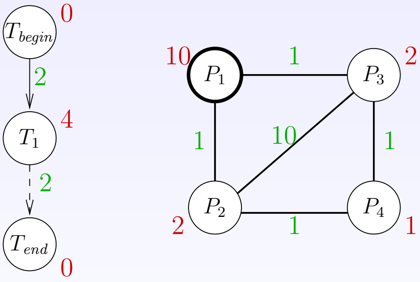
Computations and communications

P_i requires $w_{i,k}$ time-units to process task T_k ($k \in \{begin, 1, end\}$).



Computations and communications

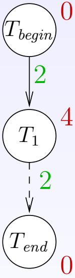
P_i requires $w_{i,k}$ time-units to process task T_k ($k \in \{begin, 1, end\}$).

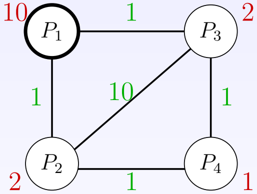


Edge $e_{k,l} : T_k \to T_l$ in G_A is labeled with $data_{k,l}$: data volume generated by T_k and used by T_l

Computations and communications

P_i requires $w_{i,k}$ time-units to process task T_k ($k \in \{begin, 1, end\}$).





Edge $e_{k,l} : T_k \rightarrow T_l$ in G_A is labeled with $data_{k,l}$: data volume generated by T_k and used by T_l

Transfer time of a file $e_{k,l}$ from P_i to P_j: $data_{k,l} \times c_{i,j}$

Definitions

Allocation An allocation is a pair of mappings: $\pi : V_A \mapsto V_P$ and $\sigma : E_A \mapsto \{\text{paths in } G_P\}$

Definitions

Allocation An allocation is a pair of mappings: $\pi : V_A \mapsto V_P$ and $\sigma : E_A \mapsto \{\text{paths in } G_P\}$

Schedule A schedule associated to an allocation (π, σ) is a pair of mappings: $t_\pi : V_A \mapsto \mathbb{R}$ and application $t_\sigma : E_A \times E_P \mapsto \mathbb{R}$, satisfying to:

- precedence constraints
- resource constraints on processors
- resource constraints on network links
- one-port constraints

Activity variables

$cons(P_i, T_k)$: average number of tasks of type T_k processed by P_i every time-unit

$$\forall P_i, \forall T_k \in V_A,\ 0 \leq cons(P_i, T_k) \times w_{i,k} \leq 1$$

Activity variables

$cons(P_i, T_k)$: average number of tasks of type T_k processed by P_i every time-unit

$$\forall P_i, \forall T_k \in V_A,\ 0 \leq cons(P_i, T_k) \times w_{i,k} \leq 1$$

$sent(P_i \rightarrow P_j, e_{k,l})$: average number of files of type $e_{k,l}$ sent from P_i to P_j every time-unit

$$\forall P_i, P_j,\ 0 \leq sent(P_i \rightarrow P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leq 1$$

Steady-state equations

One-port for outgoing communications P_i sends messages to its neighbors sequentially

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leq 1$$

Steady-state equations

One-port for outgoing communications P_i sends messages to its neighbors sequentially

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leq 1$$

One-port for ingoing communications P_i receives messages sequentially

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leq 1$$

Steady-state equations

One-port for outgoing communications P_i sends messages to its neighbors sequentially

$$\forall P_i, \sum_{P_i \to P_j} \sum_{e_{k,l} \in E_A} \left(sent(P_i \to P_j, e_{k,l}) \times data_{k,l} \times c_{i,j} \right) \leq 1$$

One-port for ingoing communications P_i receives messages sequentially

$$\forall P_i, \sum_{P_j \to P_i} \sum_{e_{k,l} \in E_A} \left(sent(P_j \to P_i, e_{k,l}) \times data_{k,l} \times c_{j,i} \right) \leq 1$$

Overlap Computations and communications take place simultaneously

$$\forall P_i, \sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leq 1$$

Conservation law

Consider a processor P_i and an edge $e_{k,l}$ of the application graph:

Files of type $e_{k,l}$ received: $\displaystyle\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l})$

Files of type $e_{k,l}$ generated: $cons(P_i, T_k)$

Files of type $e_{k,l}$ consumed: $cons(P_i, T_l)$

Files of type $e_{k,l}$ sent: $\displaystyle\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l})$

In steady state:

$$\forall P_i, \forall e_{k,l} : T_k \to T_l \in E_A,$$

$$\sum_{P_j \to P_i} sent(P_j \to P_i, e_{k,l}) + cons(P_i, T_k) =$$

$$\sum_{P_i \to P_j} sent(P_i \to P_j, e_{k,l}) + cons(P_i, T_l)$$

Upper bound for the throughput

$$\textsc{Maximize}\ \rho = \sum_{i=1}^{p} cons(P_i, T_{end}),$$

$$\textsc{under the constraints}$$

$$\left\{\begin{array}{ll}
\text{(1a)} & \forall P_i, \forall T_k \in V_A,\ 0 \leq cons(P_i, T_k) \times w_{i,k} \leq 1 \\
\text{(1b)} & \forall P_i, P_j,\ 0 \leq sent(P_i \rightarrow P_j, e_{k,l}) \times (data_{k,l} \times c_{i,j}) \leq 1 \\
\text{(1c)} & \forall P_i,\ \displaystyle\sum_{P_i \rightarrow P_j} \sum_{e_{k,l} \in E_A} \big(sent(P_i \rightarrow P_j, e_{k,l}) \times data_{k,l} \times c_{i,j}\big) \leq 1 \\
\text{(1d)} & \forall P_i,\ \displaystyle\sum_{P_j \rightarrow P_i} \sum_{e_{k,l} \in E_A} \big(sent(P_j \rightarrow P_i, e_{k,l}) \times data_{k,l} \times c_{j,i}\big) \leq 1 \\
\text{(1e)} & \forall P_i,\ \displaystyle\sum_{T_k \in V_A} cons(P_i, T_k) \times w_{i,k} \leq 1 \\
\text{(1f)} & \forall P_i, \forall e_{k,l} \in E_A : T_k \rightarrow T_l, \\
& \qquad\qquad \displaystyle\sum_{P_j \rightarrow P_i} sent(P_j \rightarrow P_i, e_{k,l}) + cons(P_i, T_k) = \\
& \qquad\qquad\qquad\qquad \displaystyle\sum_{P_i \rightarrow P_j} sent(P_i \rightarrow P_j, e_{k,l}) + cons(P_i, T_l)
\end{array}\right.$$

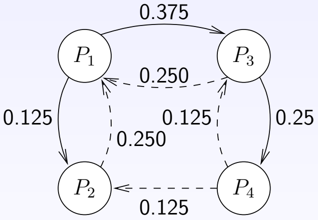
How to design a schedule achieving this throughput?

Back to the example

Computations

	$cons(P_i, T_1)$
P_1	0.025
P_2	0.125
P_3	0.125
P_4	0.250
Total	21 tasks / 40 seconds

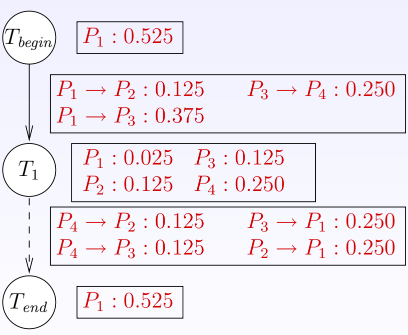
Communications

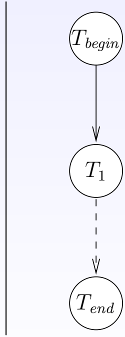


$sent(P_i \rightarrow P_j, e_{k,l})$

Decomposition into a set of allocations (1/2)

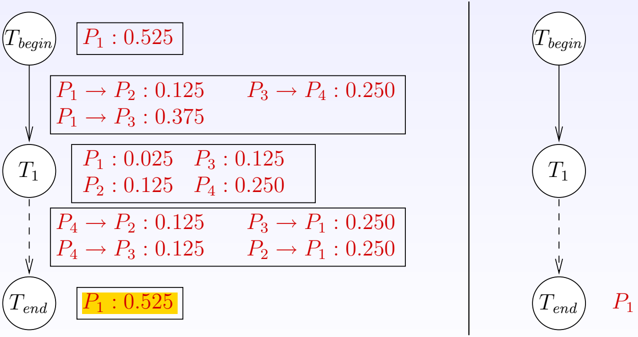
Steady state = superposition of several allocations





Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



$P_1 : 0.525$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

$P_1 : 0.025$ $P_3 : 0.125$
$P_2 : 0.125$ $P_4 : 0.250$

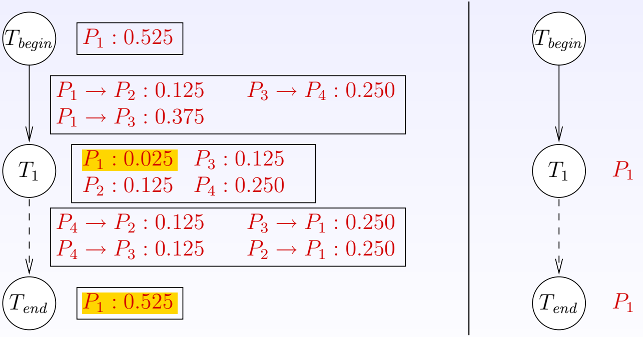
$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

$P_1 : 0.525$

P_1

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



$P_1 : 0.525$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

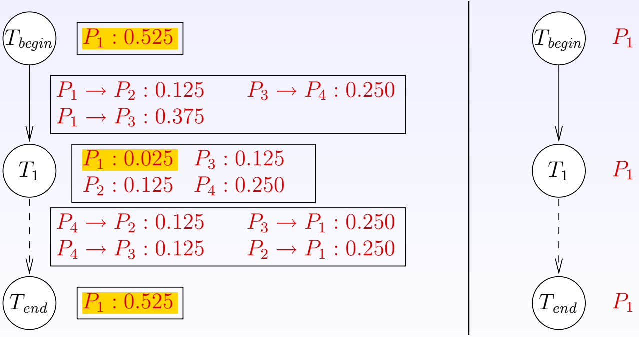
$P_1 : 0.025$ $P_3 : 0.125$
$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

$P_1 : 0.525$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



$P_1 : 0.525$

$P_1 \rightarrow P_2 : 0.125$ $\quad P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

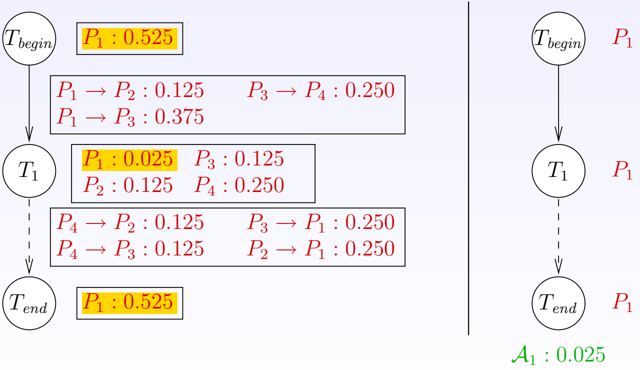
$P_1 : 0.025$ $\quad P_3 : 0.125$
$P_2 : 0.125$ $\quad P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $\quad P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $\quad P_2 \rightarrow P_1 : 0.250$

$P_1 : 0.525$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin}

$P_1 : 0.525$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

T_1

$P_1 : 0.025$ $P_3 : 0.125$
$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

T_{end}

$P_1 : 0.525$

T_{begin} P_1

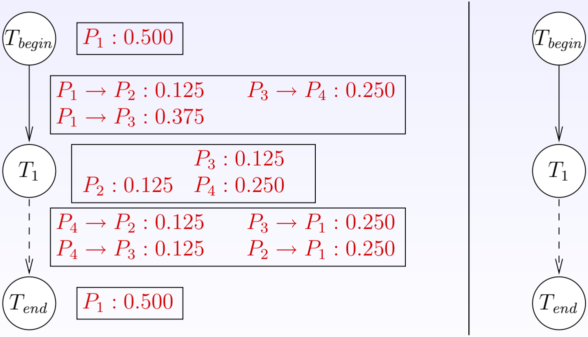
T_1 P_1

T_{end} P_1

$\mathcal{A}_1 : 0.025$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



$P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $\quad P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

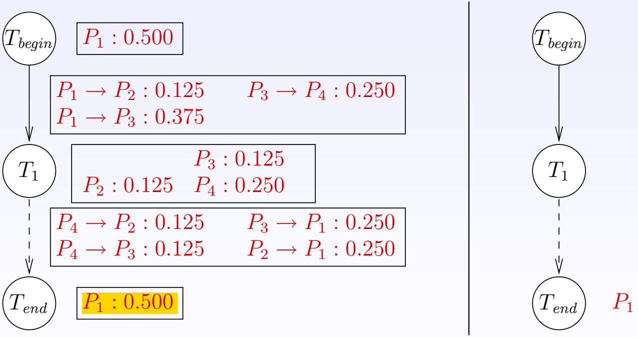
$P_3 : 0.125$
$P_2 : 0.125$ $\quad P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $\quad P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $\quad P_2 \rightarrow P_1 : 0.250$

$P_1 : 0.500$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



$P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$

$P_3 : 0.125$
$P_2 : 0.125$ $P_4 : 0.250$

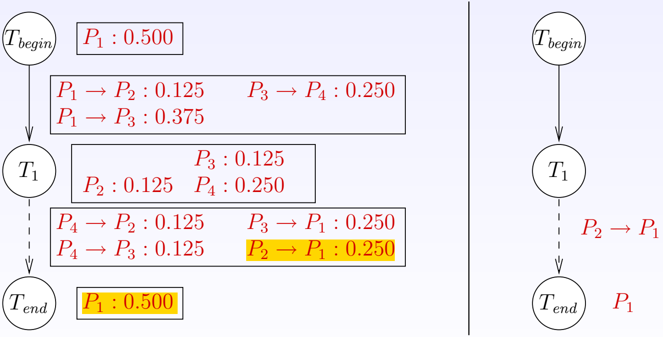
$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

$P_1 : 0.500$

P_1

Decomposition into a set of allocations (1/2)

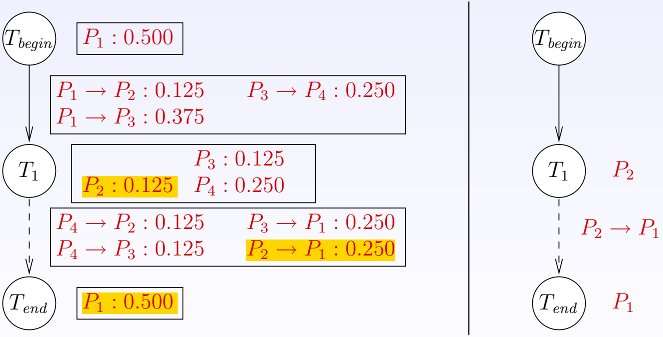
Steady state = superposition of several allocations



T_{begin}
$P_1 : 0.500$
$P_1 \rightarrow P_2 : 0.125$
$P_3 \rightarrow P_4 : 0.250$
$P_1 \rightarrow P_3 : 0.375$
T_1
$P_3 : 0.125$
$P_2 : 0.125$
$P_4 : 0.250$
$P_4 \rightarrow P_2 : 0.125$
$P_3 \rightarrow P_1 : 0.250$
$P_4 \rightarrow P_3 : 0.125$
$P_2 \rightarrow P_1 : 0.250$
T_{end}
$P_1 : 0.500$
T_{begin}
T_1
$P_2 \rightarrow P_1$
T_{end}
P_1

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin}

$P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.375$

T_1

$P_3 : 0.125$

$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

T_{end}

$P_1 : 0.500$

T_{begin}

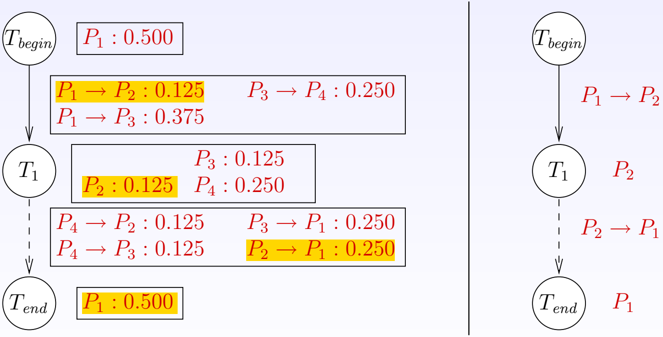
T_1 P_2

$P_2 \rightarrow P_1$

T_{end} P_1

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin}

$P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.375$

T_1

$P_3 : 0.125$

$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

T_{end}

$P_1 : 0.500$

T_{begin}

$P_1 \rightarrow P_2$

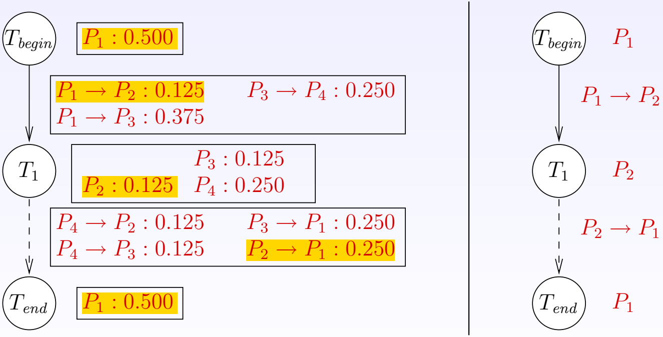
T_1 P_2

$P_2 \rightarrow P_1$

T_{end} P_1

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin} $P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.375$

T_1 $P_3 : 0.125$

$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

T_{end} $P_1 : 0.500$

T_{begin} P_1

$P_1 \rightarrow P_2$

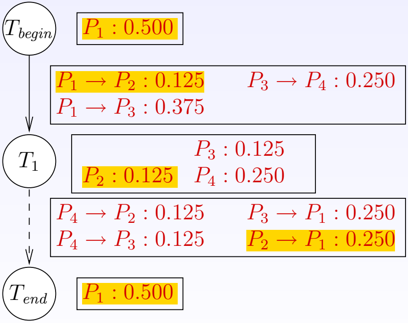
T_1 P_2

$P_2 \rightarrow P_1$

T_{end} P_1

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin}

$P_1 : 0.500$

$P_1 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.375$

T_1

$P_3 : 0.125$

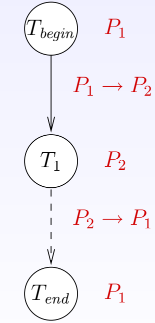
$P_2 : 0.125$ $P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.250$

T_{end}

$P_1 : 0.500$



T_{begin} P_1

$P_1 \rightarrow P_2$

T_1 P_2

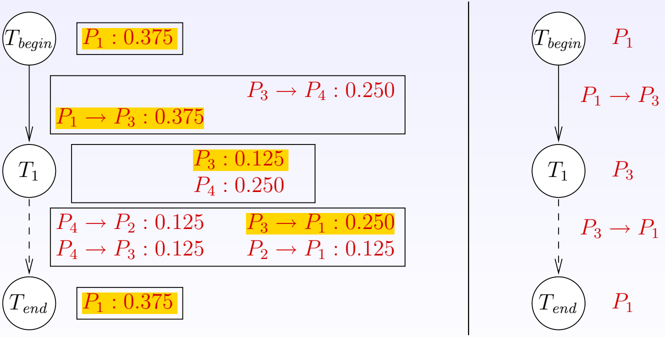
$P_2 \rightarrow P_1$

T_{end} P_1

$\mathcal{A}_2 : 0.125$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin} $P_1 : 0.375$

$P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.375$

T_1 $P_3 : 0.125$

$P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.250$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.125$

T_{end} $P_1 : 0.375$

T_{begin} P_1

$P_1 \rightarrow P_3$

T_1 P_3

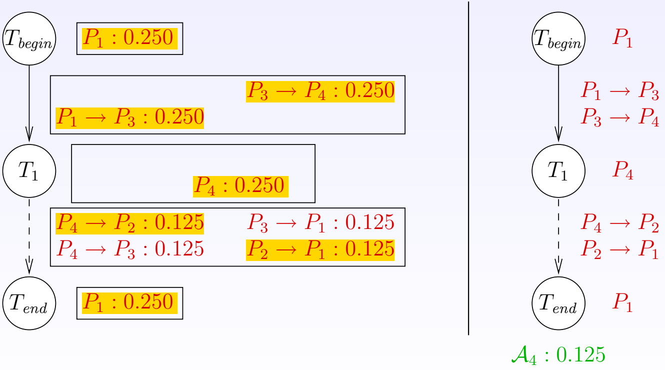
$P_3 \rightarrow P_1$

T_{end} P_1

$\mathcal{A}_3 : 0.125$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin}

$P_1 : 0.250$

$P_3 \rightarrow P_4 : 0.250$

$P_1 \rightarrow P_3 : 0.250$

T_1

$P_4 : 0.250$

$P_4 \rightarrow P_2 : 0.125$ $P_3 \rightarrow P_1 : 0.125$

$P_4 \rightarrow P_3 : 0.125$ $P_2 \rightarrow P_1 : 0.125$

T_{end}

$P_1 : 0.250$

T_{begin} P_1

$P_1 \rightarrow P_3$

$P_3 \rightarrow P_4$

T_1 P_4

$P_4 \rightarrow P_2$

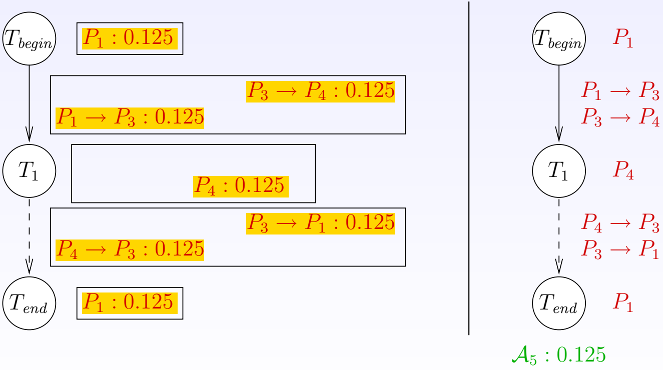
$P_2 \rightarrow P_1$

T_{end} P_1

$\mathcal{A}_4 : 0.125$

Decomposition into a set of allocations (1/2)

Steady state = superposition of several allocations



T_{begin} $P_1 : 0.125$

$P_3 \rightarrow P_4 : 0.125$

$P_1 \rightarrow P_3 : 0.125$

T_1 $P_4 : 0.125$

$P_3 \rightarrow P_1 : 0.125$

$P_4 \rightarrow P_3 : 0.125$

T_{end} $P_1 : 0.125$

T_{begin} P_1

$P_1 \rightarrow P_3$

$P_3 \rightarrow P_4$

T_1 P_4

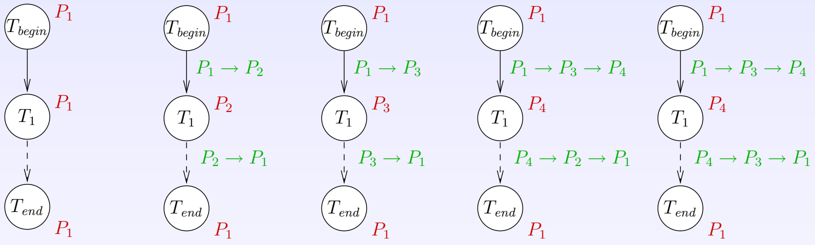
$P_4 \rightarrow P_3$

$P_3 \rightarrow P_1$

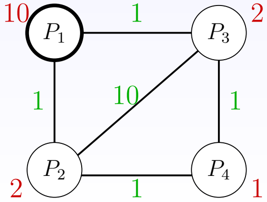
T_{end} P_1

$\mathcal{A}_5 : 0.125$

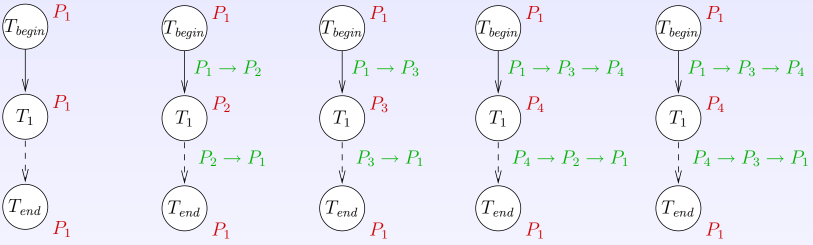
Decomposition into a set of allocations (2/2)



$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$
0,025	0,125	0,125	0,125	0,125

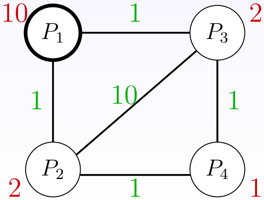


Decomposition into a set of allocations (2/2)

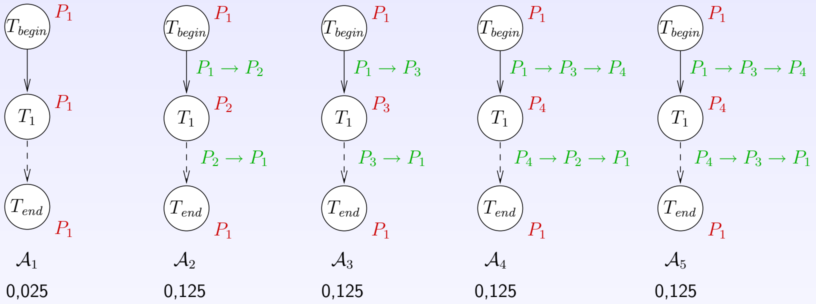


$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$	$\mathcal{A}_5$
0,025	0,125	0,125	0,125	0,125

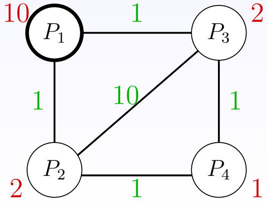
This decomposition is always possible



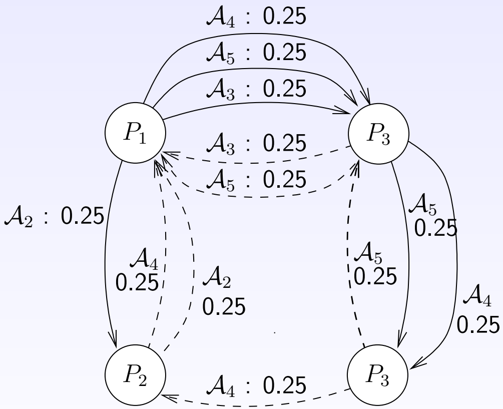
Decomposition into a set of allocations (2/2)



How to orchestrate these allocations?

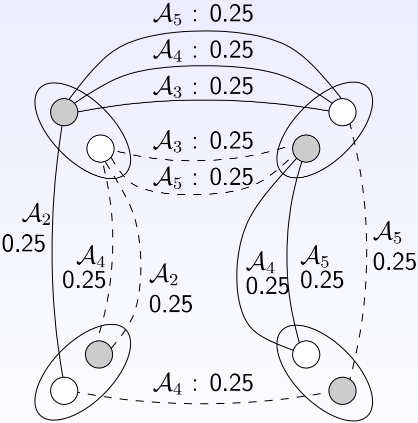


Communication graph

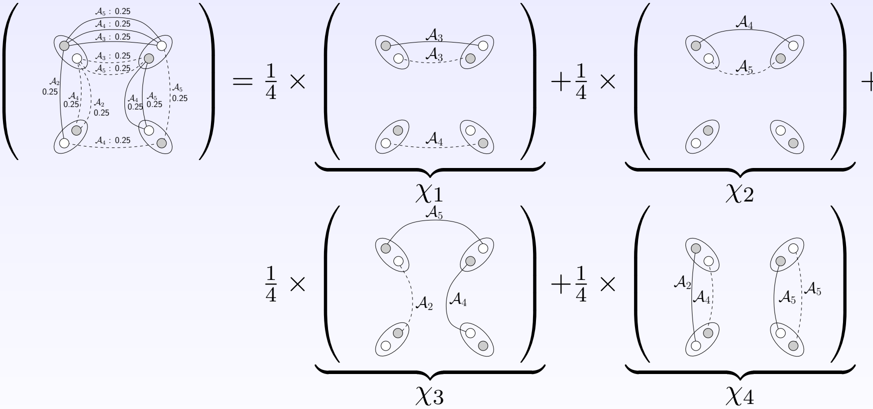


Fraction of time spent transferring some $e_{k,l}$ file from P_i to P_j for a given allocation

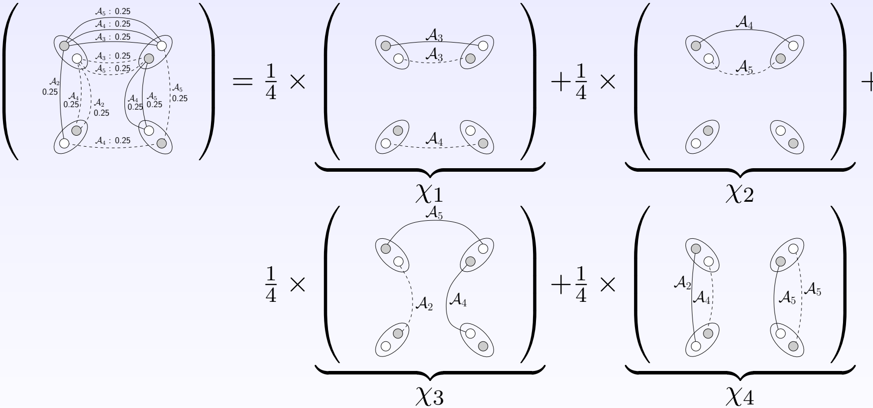
One-port constraints = matching



Edge coloring (decomposition into matchings)

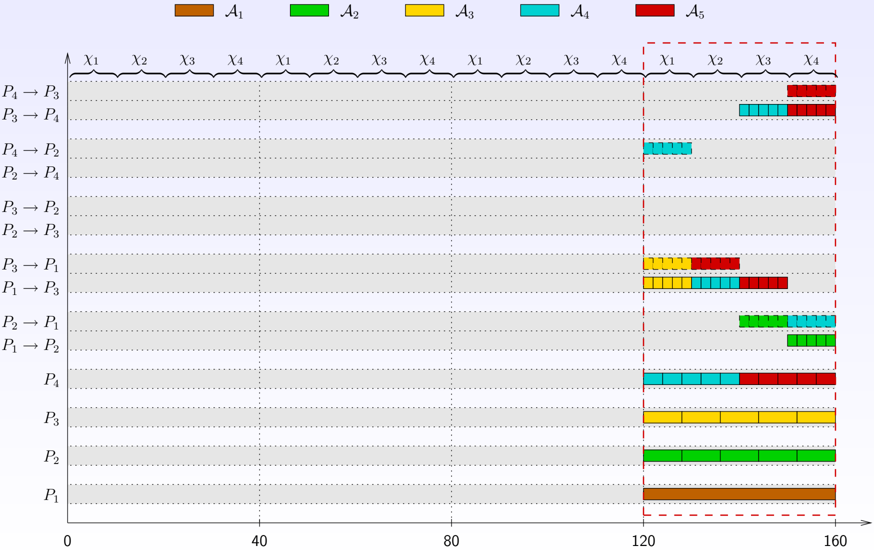


Edge coloring (decomposition into matchings)

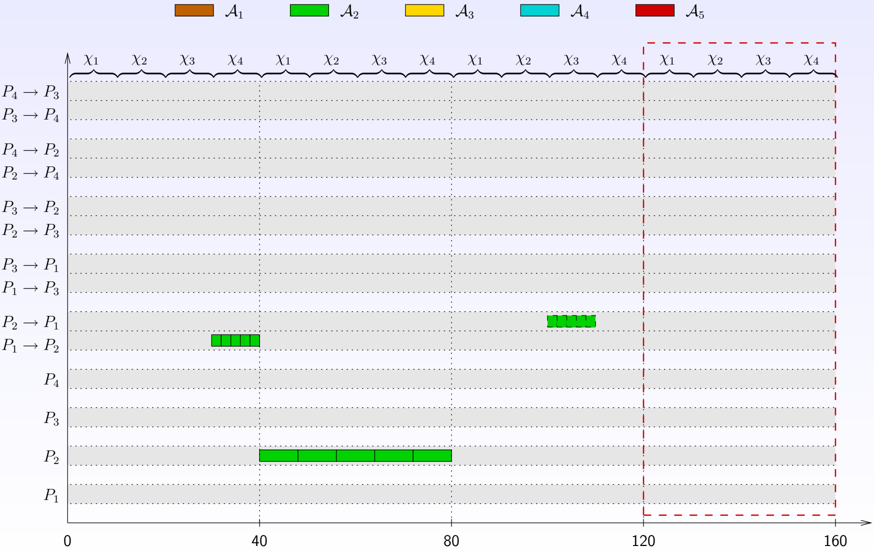


This decomposition is always possible

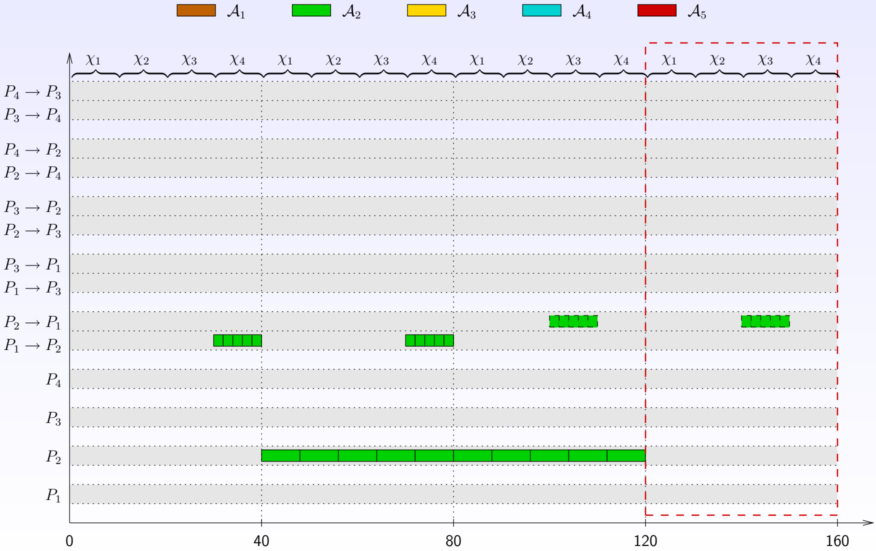
Cyclic scheduling achieving optimal throughput



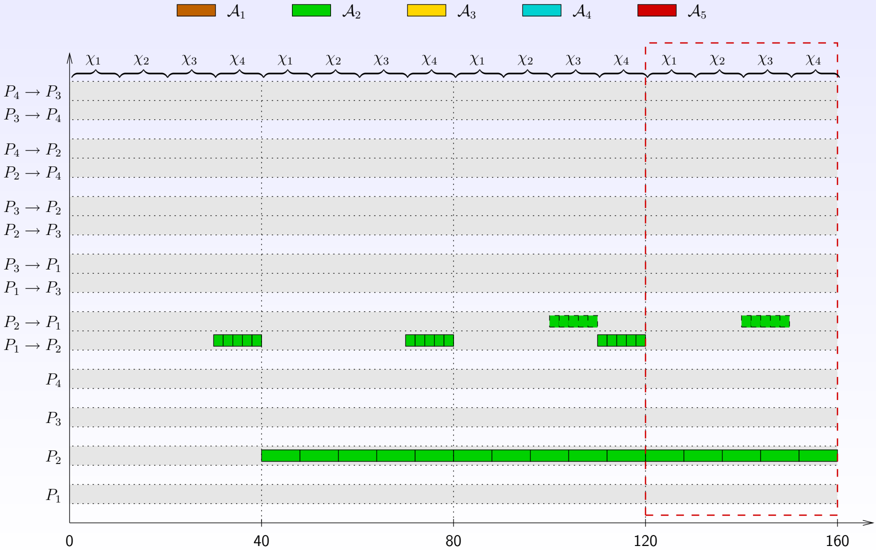
Cyclic scheduling achieving optimal throughput



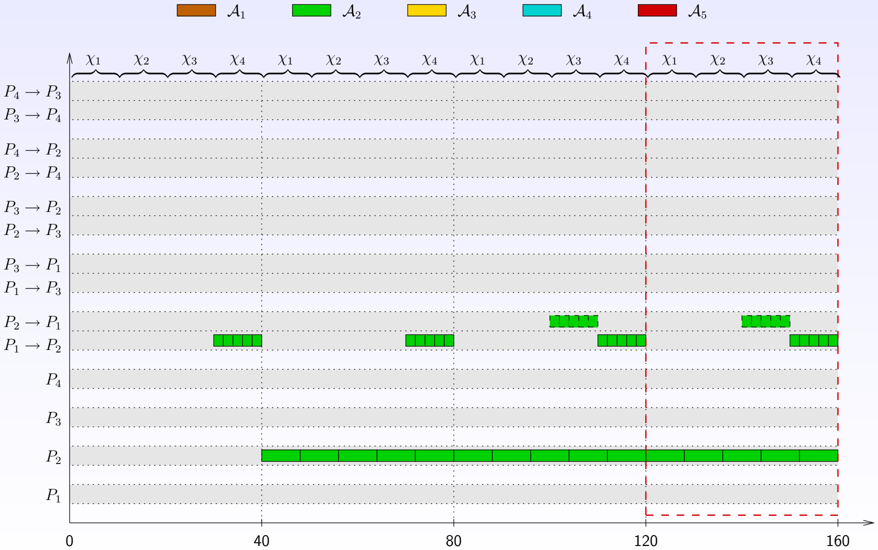
Cyclic scheduling achieving optimal throughput



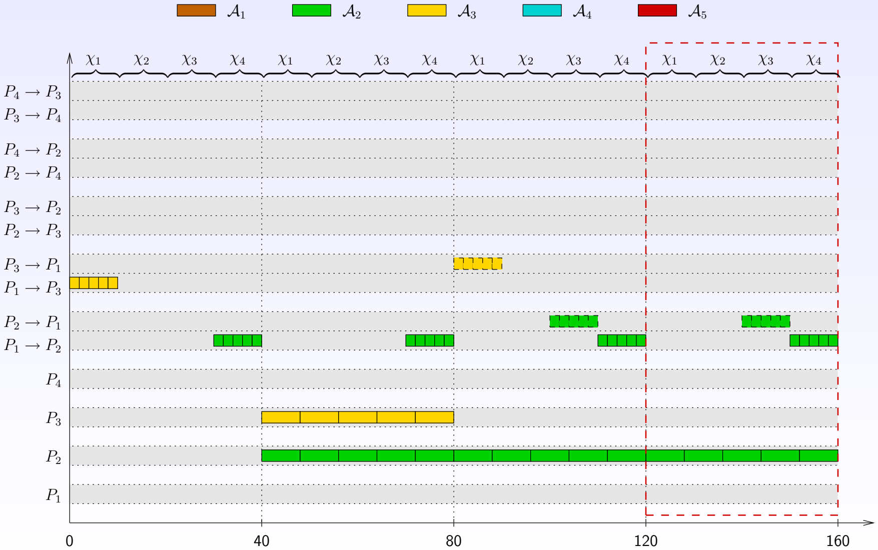
Cyclic scheduling achieving optimal throughput



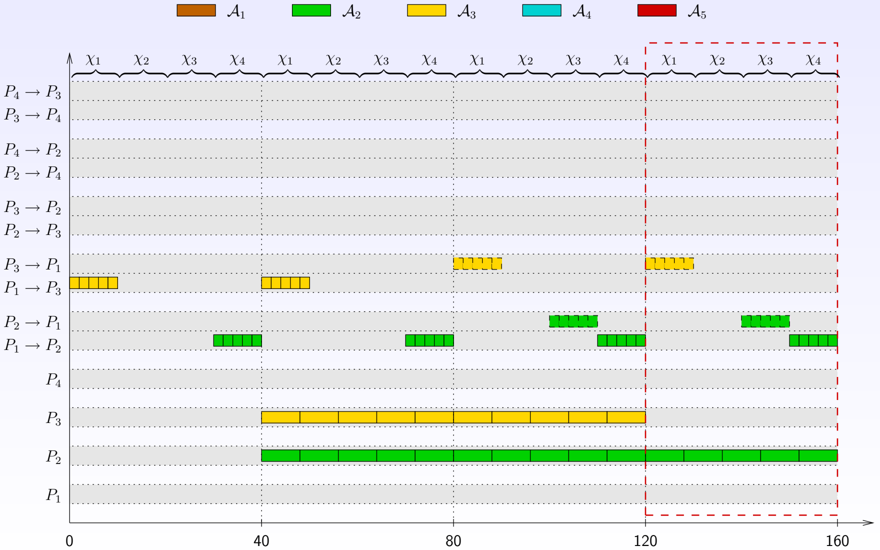
Cyclic scheduling achieving optimal throughput



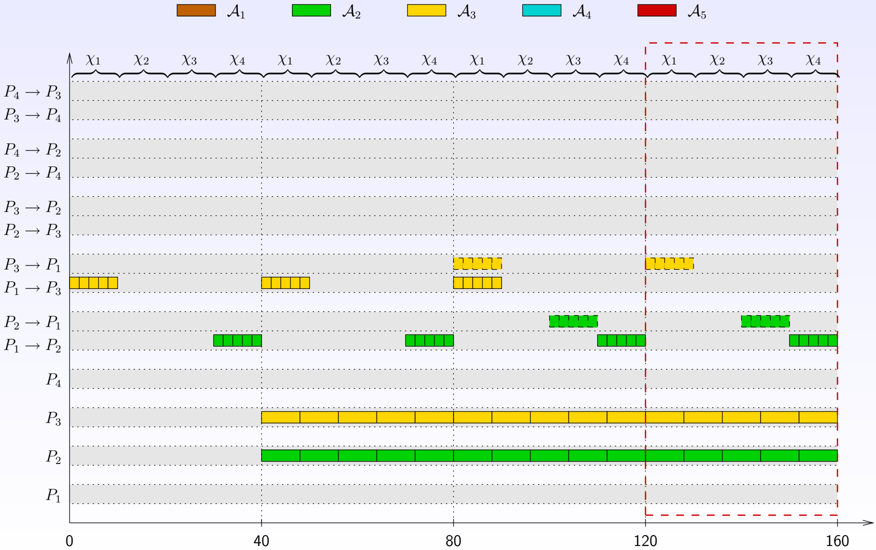
Cyclic scheduling achieving optimal throughput



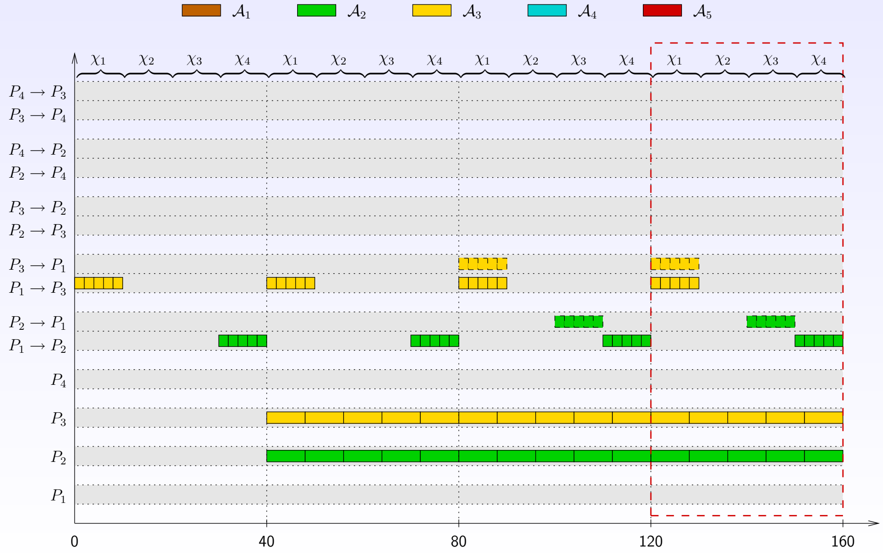
Cyclic scheduling achieving optimal throughput



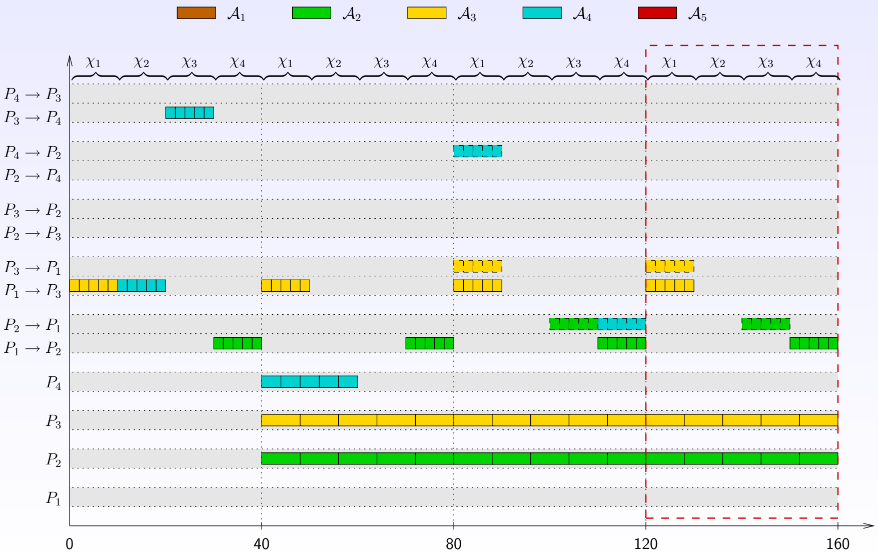
Cyclic scheduling achieving optimal throughput



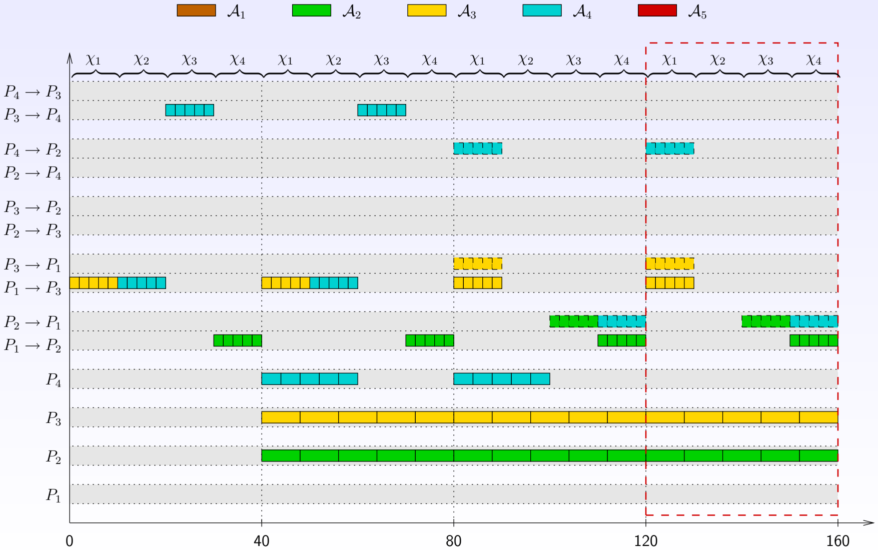
Cyclic scheduling achieving optimal throughput



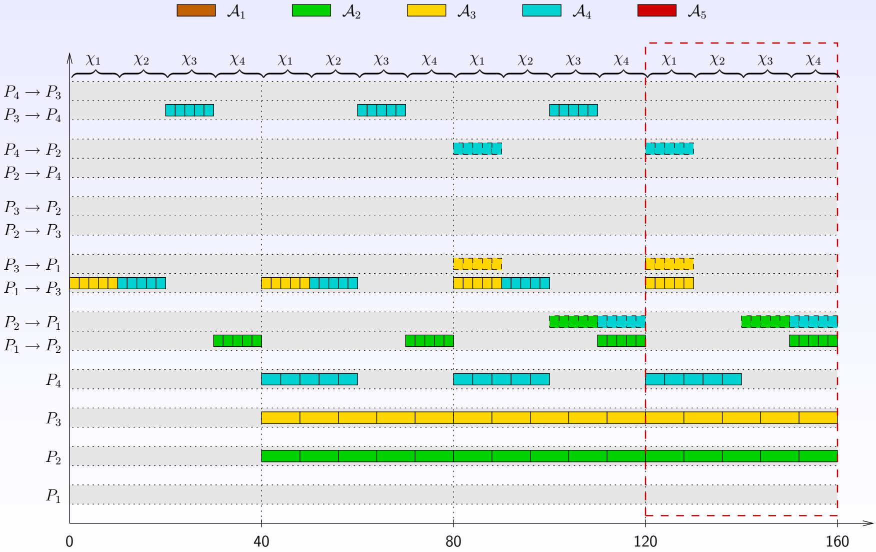
Cyclic scheduling achieving optimal throughput



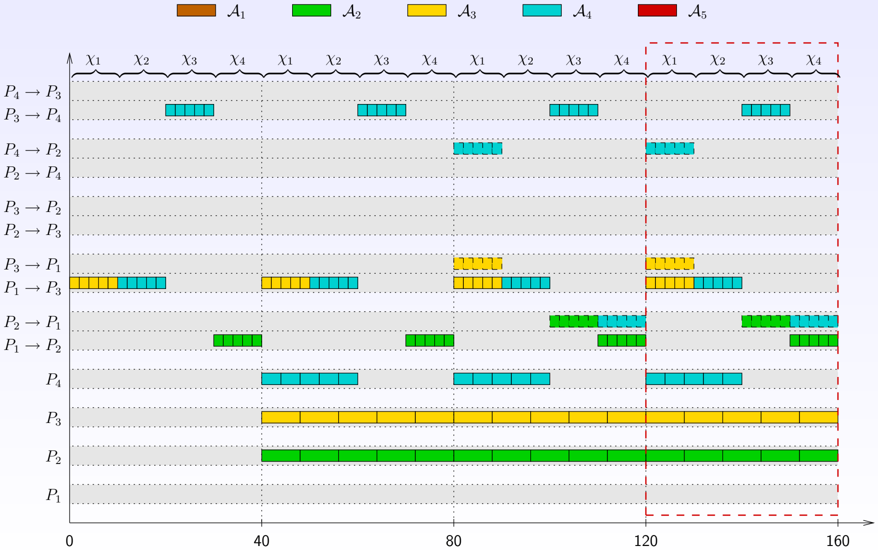
Cyclic scheduling achieving optimal throughput



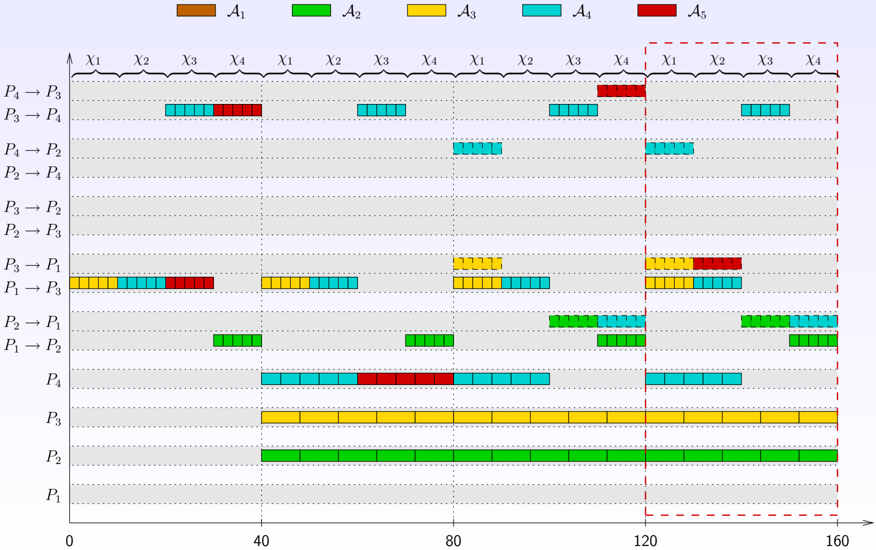
Cyclic scheduling achieving optimal throughput



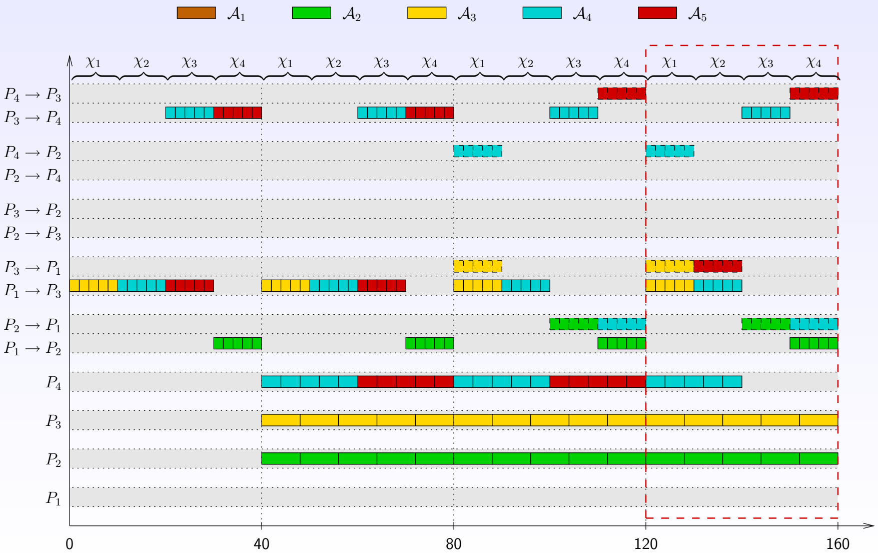
Cyclic scheduling achieving optimal throughput



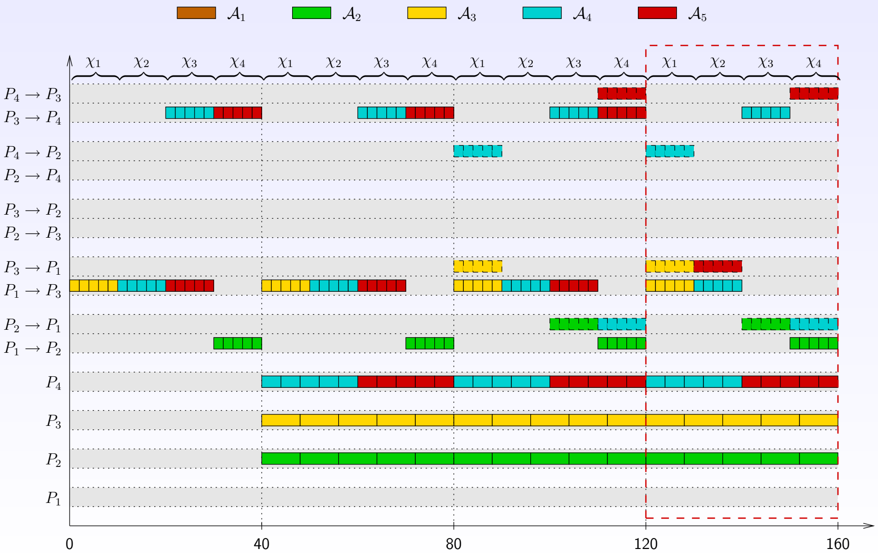
Cyclic scheduling achieving optimal throughput



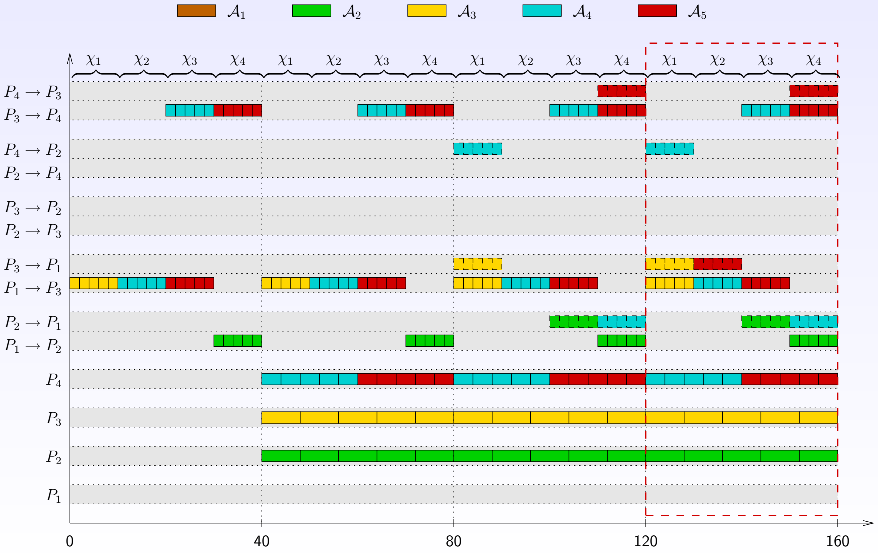
Cyclic scheduling achieving optimal throughput



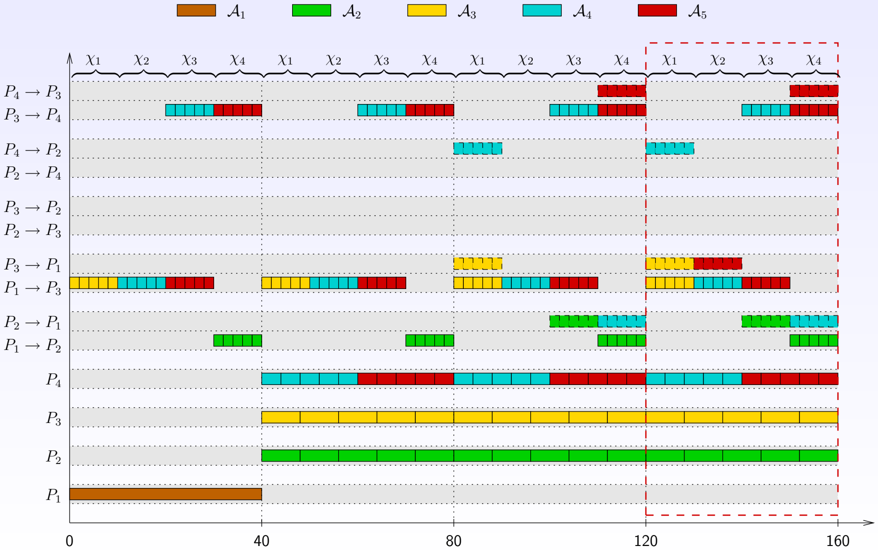
Cyclic scheduling achieving optimal throughput



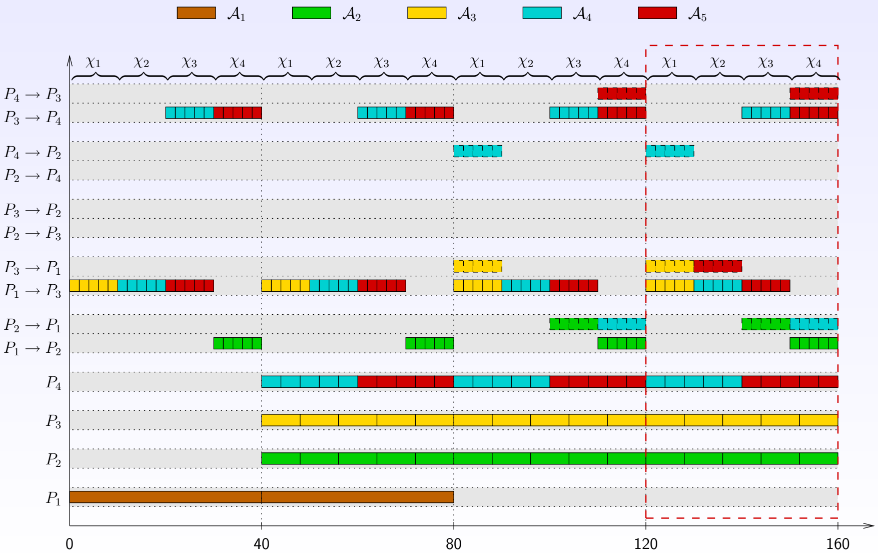
Cyclic scheduling achieving optimal throughput



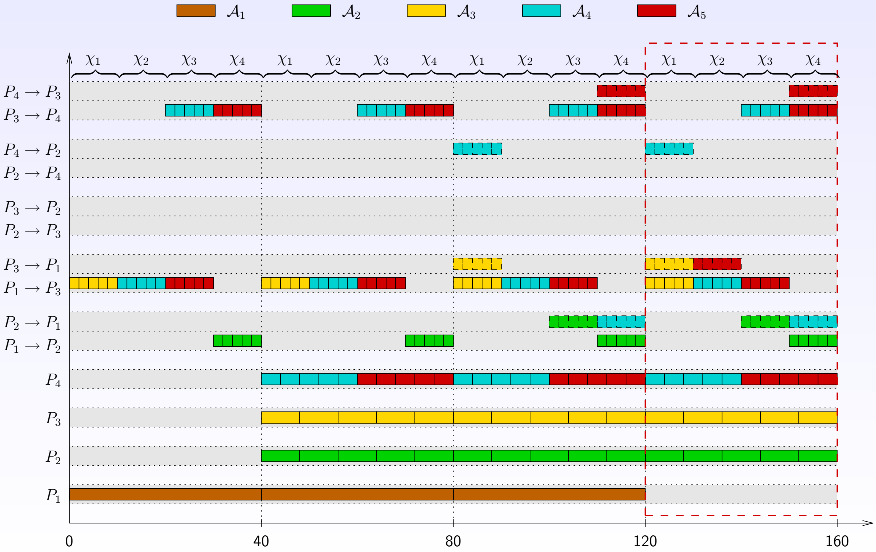
Cyclic scheduling achieving optimal throughput



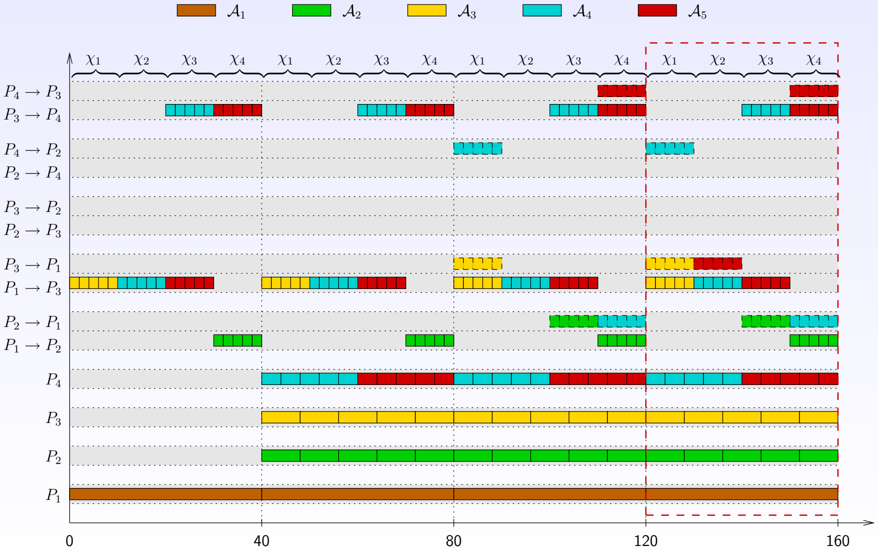
Cyclic scheduling achieving optimal throughput



Cyclic scheduling achieving optimal throughput



Cyclic scheduling achieving optimal throughput



Asymptotically optimal schedule

- The technique used in the example is
  - general
  - polynomial
- The resulting schedule is asymptotically optimal: within T time-steps, it differs from the optimal schedule by a constant number of tasks (independent of T)

Extensions to collections of general task graphs

- More difficult but possible
- Maximizing throughput NP-hard ☹
- Most application DAGs have polynomial number of joins
  $\Rightarrow$ polynomial solution ☺

Bibliography

- Packet routing:
  Asymptotically optimal algorithms for job shop scheduling and packet routing, D. Bertsimas and D. Gamarnik,
  In *Journal of Algorithms 33, 2 (1999), 296-318*

- Steady-state for independent tasks:
  Scheduling strategies for master-slave tasking on heterogeneous processor platforms, C. Banino et al.,
  In *IEEE TPDS 15, 4 (2004), 319-330*

- Steady-state for DAGs: complete reseach report
  Assessing the impact and limits of steady-state scheduling for mixed task and data parallelism on heterogeneous platforms., O. Beaumont et al.,
  LIP research report, RR-2004-20, http://www.ens-lyon.fr/LIP/Pub/Rapports/RR/RR2004/RR2004-20.ps.gz

- With bounded multi-port model:
  Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput, B. Hong and V.K. Prasanna