# Steady-State Scheduling

Loris Marchal

November 14, 2012

## 1 The context

### Platform

Platform : heterogeneous and distributed :
- processors with different capabilities ;
- communication links of different characteristics.

### Applications

Application made of a very (very) large number of tasks, the tasks can be clustered into a finite number of types, all tasks of a same type having the same characteristics.

### Principle

When we have a very large number of identical tasks to execute, we can imagine that, after some initiation phase, we will reach a (long) steady-state, before a termination phase.

If the steady-state is long enough, the initiation and termination phases will be negligible.

## 2 Routing packets with fixed communication routes

### The problem

Problem : sending a set of message flows.

In a communication network, several flow of packets must be dispatched, each packet flow must be sent from a route to a destination, while following a given path linking the source to the destination.

**Notations**

– $(V, A)$ an oriented graph, representing the communication network.

– A set of $n_c$ flows which must be dispatched.

– The $k$-th flow is denoted $(s_k, t_k, P_k, n_k)$, where
  – $s_k$ is the source of packets ;
  – $t_k$ is the destination ;
  – $P_k$ is the path to be followed ;
  – $n_k$ is the number of packets in the flow.
  We denote by $a_{k,i}$ the $i$-th edge in the path $P_k$.

**Hypotheses**

– A packet goes through an edge $A$ in a unit of time.

– At a given time, a single packet traverses a given edge.

**Objective**

We must decide which packet must go through a given edge at a given time, in order to minimize the overall execution time.

**Lower bound on the duration of schedules**

We call **congestion** of edge $a \in A$, and we denote by $C_a$, the total number of packets which go through edge $a$ :

$$C_a = \sum_{k \ | \ a \in P_k} n_k \qquad C_{\max} = \max_a C_a$$

$C_{\max}$ is a lower bound on the execution time of any schedule.

$$C^* \geq C_{\max}$$

A “fluid” (fractional) resolution of our problem will give us a solution which executes in a time $C_{\max}$.

# 3 Resolution of the “fluidified” problem

***Fluidified* (fractional) version : notations**

**Principle** :

– we do not look for an integral solution but for a rational one.

– $n_{k,i}(t)$ (fractional) number of packets waiting at the entrance of the $i$-th edge of the $k$-th path, at time $t$.

– $T_{k,i}(t)$ is the overall time used by the edge $a_{k,i}$ for packets of the $k$-th flow, during the interval of time $[0; t]$.

***Fluidified* (fractional) version : writing the equations**

1. Initiating the communications

$$n_{k,1}(t) = n_k - T_{k,1}(t), \qquad \text{for each } k$$

2. Conservation law

$$n_{k,i+1}(t) = T_{k,i}(t) - T_{k,i+1}(t), \qquad \text{for each } k$$

3. Resource constraints

$$\sum_{(k,i) \ | \ a_{k,i}=a} T_{k,i}(t_2) - T_{k,i}(t_1) \leq t_2 - t_1, \forall a \in A, \forall t_2 \geq t_1 \geq 0$$

4. Objective

$$\text{Minimize } C_{\text{frac}} = \int_0^\infty \mathbb{1}\left(\sum_{k,i} n_{k,i}(t)\right) dt$$

**Lower bound**

- $n_{k,1}(t) = n_k - T_{k,1}(t), \qquad \text{for each } k$
- $n_{k,i+1}(t) = T_{k,i}(t) - T_{k,i+1}(t), \qquad \text{for each } k$
- At any time $t$, $\displaystyle\sum_{j=1}^{i} n_{k,j}(t) = n_k - T_{k,i}(t)$
- For each edge $a$ : $\displaystyle\sum_{(k,i)|a_{k,i}=a} \sum_{j=1}^{i} n_{k,j}(t) = \sum_{(k,i)|a_{k,i}=a} n_k - \sum_{(k,i)|a_{k,i}=a} T_{k,i}(t) \geq C_a - t$

  As long as $t < C_a$, there are packets in the system.

  Therefore, $C_{\text{frac}} \geq \max_a C_a = C_{\max}$

**A candidate for the solution**

For $t \leq C_{\max}$
- $T_{k,i}(t) = \dfrac{n_k}{C_{\max}} t$, for each $k$ and $i$.
- $n_{k,1}(t) = n_k - T_{k,1}(t) = n_k - \dfrac{n_k}{C_{\max}} t = n_k \left(1 - \dfrac{t}{C_{\max}}\right), \qquad \forall k$
- $n_{k,i}(t) = 0$, for each $k$ and $i \geq 2$.

For $t \geq C_{\max}$
- $T_{k,i}(t) = n_k$
- $n_{k,i}(t) = 0$

This solution is a schedule of makespan $C_{\max}$. We still have to show that it is feasible.

**Checking the solution (for $t \leq C_{\max}$)**

1. $n_{k,1}(t) = n_k - T_{k,1}(t), \qquad$ for each $k$
   Satisfied by definition.

2. $n_{k,i+1}(t) = T_{k,i}(t) - T_{k,i+1}(t), \qquad$ for each $k$
   $T_{k,i}(t) - T_{k,i+1}(t) = \frac{n_k}{C_{\max}}t - \frac{n_k}{C_{\max}}t = 0 = n_{k,i+1}(t)$

3. $$\sum_{(k,i)\ \mid\ a_{k,i}=a} T_{k,i}(t_2) - T_{k,i}(t_1) \leq t_2 - t_1, \forall a \in A, \forall t_2 \geq t_1 \geq 0$$
   $$\sum_{(k,i)\ \mid\ a_{k,i}=a} T_{k,i}(t_2) - T_{k,i}(t_1) = \sum_{(k,i)\ \mid\ a_{k,i}=a} \frac{n_k}{C_{\max}}(t_2 - t_1) = \frac{C_a}{C_{\max}}(t_2 - t_1) \leq t_2 - t_1$$

# 4 Building a schedule

**Definition of a round**

- $\Omega \approx$ duration of a round (will be defined later).
- $m_k$ : number of packets of $k$-th flow distributed in a single round.
  $$m_k = \left\lceil \frac{n_k \Omega}{C_{\max}} \right\rceil.$$
- $D_a = \sum_{(k,i)|a_{k,i}=a} 1 = |\{k|a \in P_k\}|$
  $$D_{\max} = \max_a D_a \leq n_c$$
- Period of the schedule : $\Omega + D_{\max}$.

**Schedule**

During the time interval $[j(\Omega + D_{\max}); (j+1)(\Omega + D_{\max})]$ :

The link $a$ forwards $m_k$ packets of the $k$-th flow if there exists $i$ such that $a_{k,i} = a$.

The link $a$ remains idle for a duration of :

$$\Omega + D_{\max} - \sum_{(k,i)|a_{k,i}=a} m_k$$

(If less than $m_k$ packets are waiting in the entrance of $a$ at time $j(\Omega + D_{\max})$, $a$ forwards what is available and remains idle longer.)

**Feasibility of the schedule**

$$\begin{aligned}\sum_{(k,i)|a_{k,i}=a} m_k &= \sum_{(k,i)|a_{k,i}=a} \left\lceil \frac{n_k \Omega}{C_{\max}} \right\rceil \\ &\leq \sum_{(k,i)|a_{k,i}=a} \left( \frac{n_k \Omega}{C_{\max}} + 1 \right) \\ &\leq \frac{C_a}{C_{\max}} \Omega + D_a \\ &\leq \Omega + D_{\max}\end{aligned}$$

**Behavior of the sources**

- $N_{k,i}(t)$ : number of packets of the $k$-th flow waiting at the entrance of the $i$-th edge, at time $t$.

- $a_{k,1}$ sends $m_k$ packets during $[0, \Omega + D_{\max}]$.
  $N_{k,1}(\Omega + D_{\max}) = n_k - m_k$

- $a_{k,1}$ sends $m_k$ packets during $[\Omega + D_{\max}, 2(\Omega + D_{\max})]$.
  $N_{k,1}(2(\Omega + D_{\max})) = n_k - 2m_k$

- We let $T = \left\lceil \frac{C_{\max}}{\Omega} \right\rceil (\Omega + D_{\max})$

$$N_{k,1}(T) \leq n_k - \frac{T}{\Omega + D_{\max}} m_k \leq n_k - \frac{n_k \Omega}{C_{\max}} \frac{C_{\max}}{\Omega} = 0$$

**Propagation delay**

- $a_{k,1}$ sends $m_k$ packets during $[0, \Omega + D_{\max}]$.
  $N_{k,1}(\Omega + D_{\max}) = n_k - m_k$ $\qquad N_{k,2}(\Omega + D_{\max}) = m_k$
  $N_{k,i\geq 3}(\Omega + D_{\max}) = 0$

- $a_{k,1}$ sends $m_k$ packets during $[\Omega + D_{\max}, 2(\Omega + D_{\max})]$.
  $N_{k,1}(2(\Omega + D_{\max})) = n_k - 2m_k$ $\qquad N_{k,2}(2(\Omega + D_{\max})) = m_k$
  $N_{k,3}(2(\Omega + D_{\max})) = m_k$ $\qquad N_{k,i\geq 4}(2(\Omega + D_{\max})) = 0$

- The delay between the time a packet traverses the first edge of the path $P_k$ and the time it traverses its last edge is, at worst :

$$(|P_k| - 1)(\Omega + D_{\max})$$

  We let $L = \max_k |P_k|$.

**Makespan of the schedule**

$$\begin{aligned}
C_{\text{total}} &\leq T + (L-1)(\Omega + D_{\max}) \\
&= \left\lceil \frac{C_{\max}}{\Omega} \right\rceil (\Omega + D_{\max}) + (L-1)(\Omega + D_{\max}) \\
&\leq \left( \frac{C_{\max}}{\Omega} + 1 \right) (\Omega + D_{\max}) + (L-1)(\Omega + D_{\max}) \\
&= C_{\max} + LD_{\max} + \frac{D_{\max}C_{\max}}{\Omega} + L\Omega
\end{aligned}$$

The lower bound is minimized by $\Omega = \sqrt{\frac{D_{\max}C_{\max}}{L}}$

$$C_{\text{total}} \leq C_{\max} + 2\sqrt{C_{\max}D_{\max}L} + D_{\max}L$$

**Asymptotic optimality**

$$C_{\max} \leq C^* \leq C_{\text{total}} \leq C_{\max} + 2\sqrt{C_{\max}D_{\max}L} + D_{\max}L$$

$$1 \leq \frac{C_{\text{total}}}{C_{\max}} \leq 1 + 2\sqrt{\frac{D_{\max}L}{C_{\max}}} + \frac{D_{\max}L}{C_{\max}}$$

$$\text{With } \Omega = \sqrt{\frac{D_{\max}C_{\max}}{L}}$$

**Resources needed**

$$\begin{aligned}
\sum_{(k,i)|a_{k,i}=a,k\geq 2} m_k \leq \sum_{(k,i)|a_{k,i}=a,k\geq 2} \left( \frac{n_k}{C_{\max}} \sqrt{\frac{D_{\max}C_{\max}}{L}} + 1 \right) \\
\leq \sqrt{\frac{D_{\max}C_{\max}}{L}} + D_{\max}
\end{aligned}$$

**Conclusion**

- We forget the initiation and termination phases
- Rational resolution of the steady-state
- Round whose size is the square-root of the solution :
  - Each round "loses" a constant amount of time
  - The sum of the waisted times increases less quickly than the schedule
  - Buffers of size the square-root of the solution

**Principles**
- focus on steady-state, forget transient phase
- optimize throughput during central steady-state
- in this article : trade-off between the loss in steady-state, and the loss in initialization and clean-up phases (period length = square root of optimal makespan)
- other solution : get optimal steady-state schedules
- as soon as the number of packets is large, the solution is asymptotically optimal :

$$\frac{C_{\max}}{C_{\max}^{\mathrm{opt}}} \xrightarrow[n \to \infty]{} 1$$

# 5 Steady-state scheduling for a similar problem

- Let's get a more realistic network model :
  - Given topology (graph)
  - Sending a unit-size message from $P_i$ to $P_j$ takes a time $c_{i,j}$ (edge weight). For a message of size $S$, it will take $S \times c_{i,j}$. Note that we might have $c_{i,j} \neq c_{j,i}$.
  - Each processor can send (and receive) a single message at a time (bidirectional one-port model).
  - During a communication of size $S$ from $P_i$ to $P_j$ starting at time $t$ (i.e., during $[t, t + Sc_{i,j}]$ :
    - $P_i$ cannot start another sending operation
    - $P_j$ cannot start another reception
    - $P_j$ cannot forward the message, or start a computation depending of this message

We consider here a new problem : Scatter
- scatter : one source processor sends a distinct message to a set of target processors
- series of scatter : similar to scatter big messages using pipelining

**Notations for average (fractional) numbers**
- $n(P_i \to P_j, k)$ : average number of messages of type $k$ (that is, targeting $P_k$) send through edge $(i, j)$ during one time unit
- $s(P_i \to P_j)$ : average occupation time of edge $(i, j)$ during one time unit

**Constraints**
- one-port : outgoing messages, incoming message
- relation between $n$ and $s$
- conservation law
- throughput definition

We get a linear program. Note that all valid solution can be described as $n$ and $s$, and must follow the linear program. Hence the throughput of an optimal solution of the linear program is a lower bound on the achievable throughput.

From a solution of the linear program to a real solution :

- Rational numbers : compute the lowest common multiple (lcm) of all numbers of messages, and multiply all quantities by this number
  - lcm polynomial in the input parameters of the linear program
  - potentially large period, may be shorten using approximate solution
- One-port model : from local constraint to a valid global schedule (example from the JPDC article)
  - graphs of communication (split a node in receiver/sender)
  - one-port model : a valid pattern is a matching in this graph
  - algorithm to decompose the graph in a weighted sum of matching, such that the sum of the weight is no more than the weight of a node in the graph
  - extract matchings to organize communications (if needed, avoid splitting messages by multiplying by lcm again)
- Initialization and clean-up phases :
  - Initialization : the source processors first sends all needed messages to everybody, OR compute the first activation of communications using a graph traversal...
  - Clean-up : similar.

**Asymptotic optimality**

- Every valid schedule has a throughput lower than $\rho^*$, throughput of an optimal solution of the linear program
- Let $T_i$ be the time needed for initialization and clean-up ($T_i$ constant in the number of messages send).
- Throughput for a time $T$ : $\frac{T+T_i}{T}\rho^*$
- Asymptotically optimal

**Conclusion**

- Benefits :
  - Simplicity (description : one period)
  - Efficiency (asymptotic optimality)
  - Adaptability ? (measure bandwidth during one period, change the schedule for the next one
- Drawbacks :
  - Complexity (statically allocate specific path to each packet)
  - Bad performance for small batches
  - Need for large buffers