Modeling the networks

Frédéric Vivien

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

6 novembre 2006

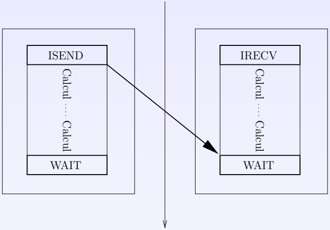
Outline

Outline

In an ideal world

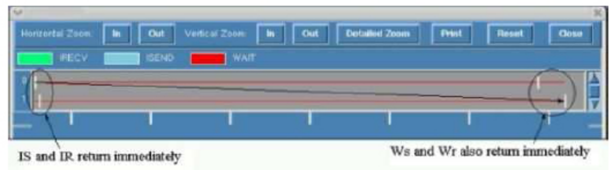
- While sending and receiving messages, one can compute.
- The computational power is not at all affected by the communications.
- Communications are realized using non blocking primitives.

Experimental scheme

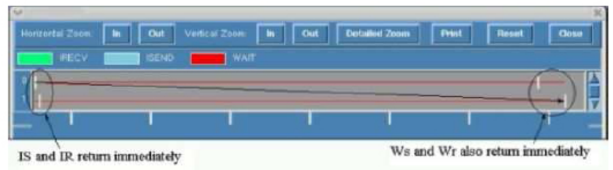


- Two processes (on two different processors): the first one sends a message to the second one.
- The two processes execute “at the same time” their asynchronous send and receive commands.
- After a (long) computational time, the two processes put themselves in wait of the termination of the communications.

Actual behavior: 1 kilobyte message

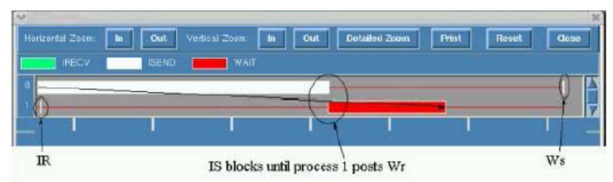


Actual behavior: 1 kilobyte message

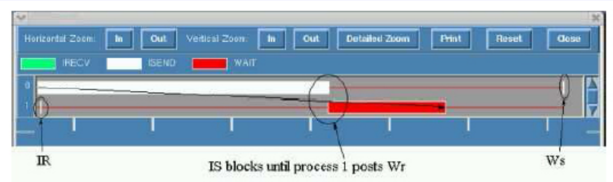


Expected behavior: the apparent behavior of the primitives is non-blocking.

Actual behavior: 60 kilobyte message (1)

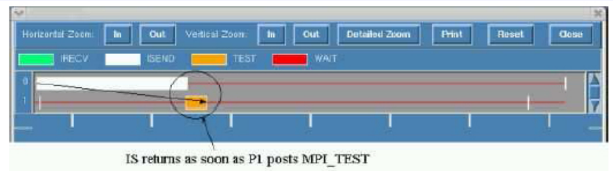


Actual behavior: 60 kilobyte message (1)



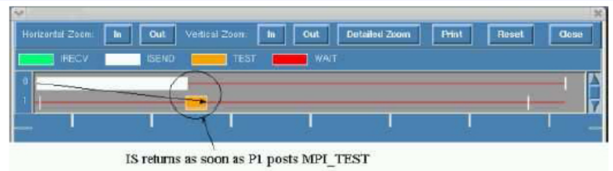
The sending, which was supposed to be non-blocking, is in fact blocking.

Actual behavior: 60 kilobyte message (2)



60 kilobyte message with a non-blocking test of communication completion in the middle of the computational phase of the receiving process.

Actual behavior: 60 kilobyte message (2)



60 kilobyte message with a non-blocking test of communication completion in the middle of the computational phase of the receiving process.

The sending is blocked until the receiver posts the termination test.

Actual behavior: other results

- The symptoms are the same, when one is sending one 60 kilobyte message or three 20 kilobyte messages.
- When one is enlarging the size of the TCP/IP socket buffer to 64 kilobytes, the sending has a non-blocking behavior.

Interpretation

- Once we have reached a certain (accumulated) size of messages, the system buffers do not allow anymore to copy locally the messages. MPI moves then to a rendezvous protocol which needs a synchronization of the two processes which are supposed to communicate.
- This behavior is not a bug. It is inherently linked to the limits of the architectures used.

Outline

Classical hypotheses

- One can compute and communicate simultaneously (using threads and/or asynchronous communications).
- Computations and communications do not interfere which each other.

Classical hypotheses

- One can compute and communicate simultaneously (using threads and/or asynchronous communications).
- Computations and communications do not interfere which each other.

What happens in practice ?

Classical hypotheses

- One can compute and communicate simultaneously (using threads and/or asynchronous communications).
- Computations and communications do not interfere which each other.

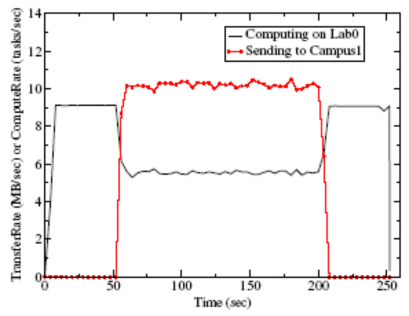
What happens in practice ?

Are the existing influences significant ?

Experimental setup

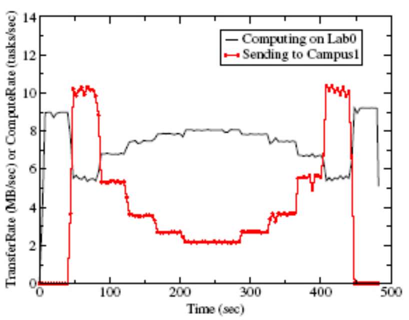
- Program written in Java using native threads.
- Processor architecture: Intel.
- Operating systems: FreeBSD, Linux 2.4 (Debian, RedHat), and Solaris (SunOS).
- Computers in the laboratory (Grail/UCSD), on the campus (UCSD), at UCSB, and on remote sites (Tennessee, Brésil, France).

Simultaneous computations and sends (1)



A processor simultaneously sends messages and computes (constant throughput).

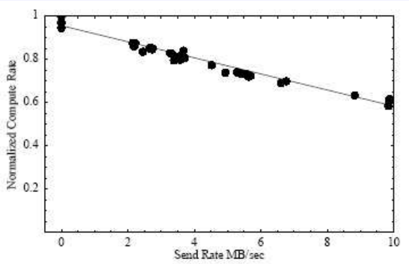
Simultaneous computations and sends (2)



A processor simultaneously sends messages and computes (varying throughput).

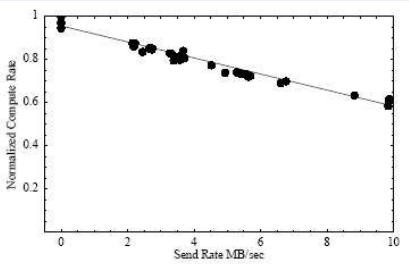
(the throughput is defined by introducing some contention on the receiver)

Simultaneous computations and sends (3)



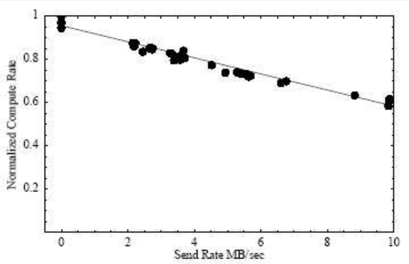
A processor simultaneously sends messages and computes: averages.

Simultaneous computations and sends (3)



A processor simultaneously sends messages and computes: averages.
Linear regression with least squares:
Compute rate $\approx$ -0.037 $\times$ Communication rate + 0.96.

Simultaneous computations and sends (3)



A processor simultaneously sends messages and computes: averages.
Linear regression with least squares:
Compute rate $\approx$ -0.037 $\times$ Communication rate $+$ 0.96.
IR $=$ 0.037 is the Interference Rate

Influence of receiving on the computations

- Average Interference rate: 0.052.
  On average, when the node receives messages at 10 MB/s, its computational power decreases by 52%.

Influence of receiving on the computations

- Average Interference rate: 0.052.
  On average, when the node receives messages at 10 MB/s, its computational power decreases by 52%.
- When a single reception takes place, the maximal throughput achieved is roughly 85%-95% of the maximal throughput achievable (when one receives simultaneously from several senders).

Influence of receiving on the computations

- Average Interference rate: 0.052.
  On average, when the node receives messages at 10 MB/s, its computational power decreases by 52%.
- When a single reception takes place, the maximal throughput achieved is roughly 85%-95% of the maximal throughput achievable (when one receives simultaneously from several senders).
- Once the maximal throughput is reached, the impact on the available computational power is maximized, whatever the number of receiver threads.

Influence of receiving on the computations

- Average Interference rate: 0.052.
  On average, when the node receives messages at 10 MB/s, its computational power decreases by 52%.
- When a single reception takes place, the maximal throughput achieved is roughly 85%-95% of the maximal throughput achievable (when one receives simultaneously from several senders).
- Once the maximal throughput is reached, the impact on the available computational power is maximized, whatever the number of receiver threads.
- Estimating the impact of a set of receptions:
  $$1 - \sum_i \mathsf{IR}(i) \times \mathsf{TR}(i) \qquad \text{where TR is the transfer rate}$$

Influence of sending on the computations

- Average interference rate: 0.034.
  On average, when the node sends messages at 10 MB/s, its computational power decreases by 34%.

Influence of sending on the computations

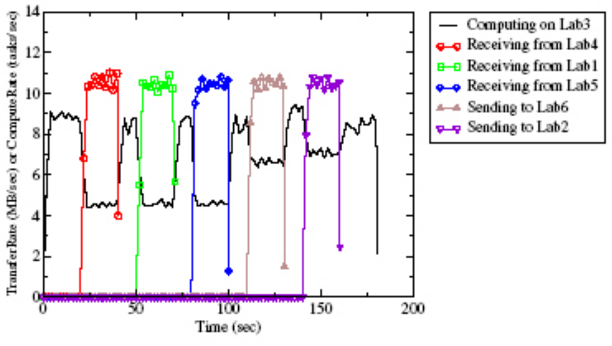
- Average interference rate: 0.034.
  On average, when the node sends messages at 10 MB/s, its computational power decreases by 34%.
- Receiving messages has more influence on the computational power than sending messages.

Influence of sending on the computations

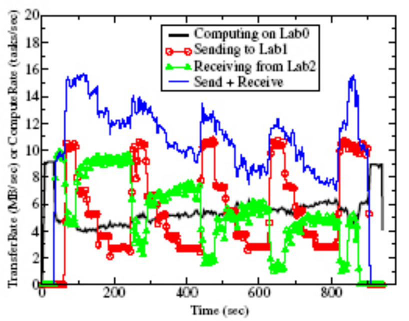
- Average interference rate: 0.034.
  On average, when the node sends messages at 10 MB/s, its computational power decreases by 34%.
- Receiving messages has more influence on the computational power than sending messages.
- Estimating the impact of a set of message sendings:

$$1 - \sum_i \mathrm{IR}(i) \times \mathrm{TR}(i) \qquad \text{where TR is the transfer rate}$$

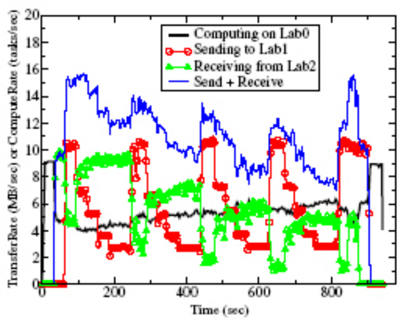
Receptions have more influence than sendings



Simultaneous sends and receives



Simultaneous sends and receives



The estimate: $1 - \mathrm{IR}_s \times \mathrm{TR}_s - \mathrm{IR}_r \times \mathrm{TR}_r$

gives (s denoting the sends and r the receives):

$\mathrm{IR}_r = 0.0427$ instead of 0.0502, $\mathrm{IR}_s = 0.0265$ instead of 0.0327

There is a synergistic influence between sends and receives.

Other observations

- The less the memory used by the application, the less important the interferences: the interference rates are application dependent !
- The size of messages has no influence.
- Remote processors have significantly higher interference rates (more than 0.070 for receives), but the achievable bandwidths are far less important.

Conclusion

- To compute and communicate simultaneously can make one "lose" more than half of the computational power.

Conclusion

- To compute and communicate simultaneously can make one "lose" more than half of the computational power.
- Once the interference rates are measured, for a given application, between each pair of processors, one can deduce a good approximation of the available computational powers.

Does this change anything in practice ?

Conclusion

- To compute and communicate simultaneously can make one "lose" more than half of the computational power.
- Once the interference rates are measured, for a given application, between each pair of processors, one can deduce a good approximation of the available computational powers.
- What about other platforms ? languages ? (same thing for the C language)

Does this change anything in practice ?

The studied problem

- Application: n identical, independent, tasks.

The studied problem

- Application: n identical, independent, tasks.
- Considered platform : a tree of heterogeneous computational resources (processors, clusters, etc.), interconnected by communications links of different characteristics.

The studied problem

- Application: n identical, independent, tasks.
- Considered platform : a tree of heterogeneous computational resources (processors, clusters, etc.), interconnected by communications links of different characteristics.
- All the input data files are initially located at the root of the tree.

The studied problem

- Application: n identical, independent, tasks.
- Considered platform : a tree of heterogeneous computational resources (processors, clusters, etc.), interconnected by communications links of different characteristics.
- All the input data files are initially located at the root of the tree.
- The root processor decides which task it executes itself and which tasks it delegates to its sons. Each internal node do the same.

The studied problem

- Application: n identical, independent, tasks.
- Considered platform : a tree of heterogeneous computational resources (processors, clusters, etc.), interconnected by communications links of different characteristics.
- All the input data files are initially located at the root of the tree.
- The root processor decides which task it executes itself and which tasks it delegates to its sons. Each internal node do the same.
- A node sends work to one of its sons only when the son requests some. When there are simultaneous requests, a scheduling policy solves the conflicts.

Known results

- Hypothesis : 1) no interferences bewteen computations and communications or 2) communications and computations are mutually exclusive.
- Bandwidth centric solution: if there is enough available bandwidth, all the sons work; otherwise, the tasks are sent to the sons which have sufficiently fast communications, priority being given to the son with the fastest communications.

The interference model to be instantiated

- Normalized computation time:

$$1 - \sum_i \mathsf{IR}_s(i) \times \mathsf{TR}_s(i) - \sum_i \mathsf{IR}_r(i) \times \mathsf{TR}_r(i)$$

- To obtain a good approximation of the different interference rate is complicated and costly: it requires the measure of computational capabilities for different throughput of sends and receives.

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathsf{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathsf{MR}^n}$$

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathrm{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathrm{MR}^n}$$

- For each son i simultaneously measure:

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathsf{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathsf{MR}^n}$$

- For each son i simultaneously measure:
  1. the bandwidth of n when sending to the son i : $\mathsf{SR}(i)$;

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathsf{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathsf{MR}^n}$$

- For each son i simultaneously measure:
  1. the bandwidth of n when sending to the son i : $\mathsf{SR}(i)$;
  2. the computational power of node n: $C_{sr}^n(i)$;

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathsf{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathsf{MR}^n}$$

- For each son i simultaneously measure:
  1. the bandwidth of n when sending to the son i : $\mathsf{SR}(i)$;
  2. the computational power of node n: $C_{sr}^n(i)$;
  3. the bandwidth of n when receiving from its father : $\mathsf{RR}(i)$.

A more simply instantiated interference model (1)

For each node n,

- measure C^n : number of tasks executed by unit of time;
- measure the maximal bandwidth achieved when receiving MR^n messages and the corresponding computational power: C_r^n.

$$\mathsf{IR}_r = \frac{\frac{C^n - C_r^n}{C^n}}{\mathsf{MR}^n}$$

- For each son i simultaneously measure:
  1. the bandwidth of n when sending to the son i : $\mathsf{SR}(i)$;
  2. the computational power of node n: $C_{sr}^n(i)$;
  3. the bandwidth of n when receiving from its father : $\mathsf{RR}(i)$.

$$\mathsf{IR}_s(i) = \frac{\left(1 - \frac{\mathsf{RR}(i)}{\mathsf{MR}^n}\frac{C^n - C_r^n}{C^n} - \frac{C_{sr}^n(i)}{C^n}\right)}{\mathsf{SR}(i)}$$

A more simply instantiated interference model (2)

- One measure for node n and one for each of its sons, instead of a potentially exponential number of measures with a set of different bandwidth values.
- The global precision is of a lesser quality, but a better local precision (inside the convex hull of the measured points).

Using the interference model

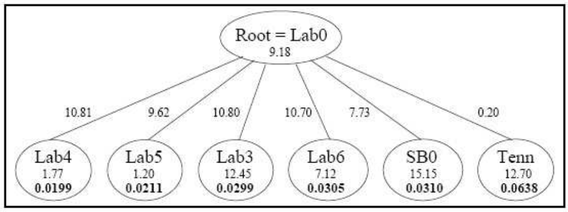
- Multi-port sends: the throughput of each node is maximized when priority is given to the sons with the smaller interference rates.
  One should not give a task to a son for which $\mathsf{IR}_s^n(i) Z C^n \geq 1$, where Z is the size of a task (to send a task to a son is more expensive in computational time than what it saves).
- One-port sends: the throughput of each node is maximized when priority is given to the sons with the highest value for: $B_r^i(1 - \mathsf{IR}_s^n Z C^n)$.

Experimental verification

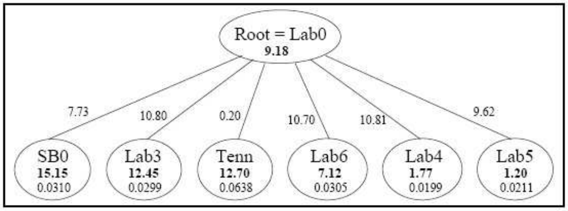
Scheduling policy:

- IA:single : *interference aware, single port*, ordering by increasing IR_s values.
- IA:multi : *interference aware, multiple ports*, ordering by increasing IR_s values.
- FCFS : first come, first serve.
- CompRate : by decreasing computational power.
- BWC : by decreasing bandwidth.
- RootComputes : the root computes everything.

Impact of the policies on the order (1)

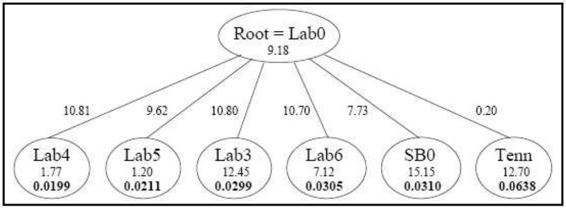


Order : interference rates.

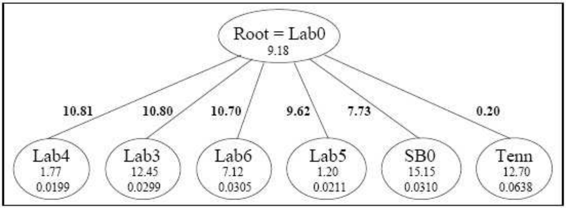


Order : computational power.

Impact of the policies on the order (2)

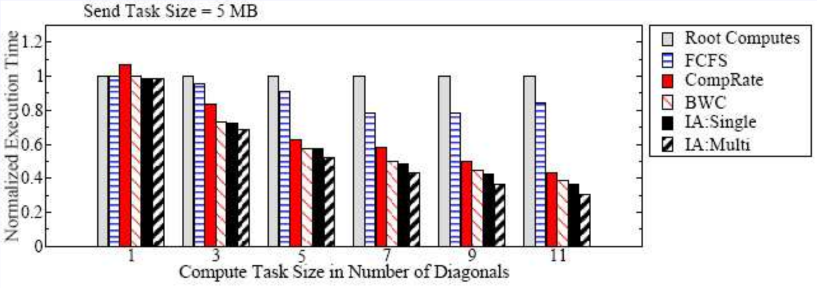


Order : interference rates.



Order : bandwidth.

Results


Send Task Size = 5 MB
Normalized Execution Time
1.2
1
0.8
0.6
0.4
0.2
0
1
3
5
7
9
11
Compute Task Size in Number of Diagonals
Root Computes
FCFS
CompRate
BWC
IA:Single
IA:Multi

Outline

The problem

How can we model the network in order to predict the time needed for messages to be sent ?

Network latencies (1)

- Classical model of communication times: $\alpha + \dfrac{x}{\beta}$ where
  - x is the message size;
  - α is the latency;
  - β is the bandwidth.

Network latencies (1)

- Classical model of communication times: $\alpha + \dfrac{x}{\beta}$ where
  - x is the message size;
  - α is the latency;
  - β is the bandwidth.
- Numerous works neglect α.
  Justification: to ease or enable problem solving.

Network latencies (1)

- Classical model of communication times: $\alpha + \dfrac{x}{\beta}$ where
  - x is the message size;
  - α is the latency;
  - β is the bandwidth.
- Numerous works neglect α.
  Justification: to ease or enable problem solving.
- Problem: the solution obtained may be unrealistic has the sending of an infinitely small message is not penalized.
  (Ex.: distributing an infinite number of infinitely small rounds.)

Network latencies (2)

Are latencies actually negligible ?

An example: TeraGrid between SDSC and NCSA (USA)

- Latency: $\approx$ 100ms
- Bandwidth: 40 Gb/s.
- Sending a 1 gigabyte message: more than a third of the time needed to send a message is due to the latency.

Conclusion ?

Computation latencies

- Programs may have a non negligible initialization cost.
- Launching a process may be long: on Globus Toolkit 2.0, launching a task doing no work (*no-op job*) may take 25 seconds (authentication, resources acquisition, process creation, etc.).

It may be necessary to take into account computation latencies even for programs whose execution time is linear in the size of input data.

Classical model of networks

Classical model

- Processors are interconnected through point to point connections (no routers, no switch, etc.).
- Multi port model: a processor can simultaneously send messages to several other processors.
- One port model: a processor can, at one time, send at most one message to one other processor.
- Only one message can travel along a given link at a given time.
- A complete graph can model a switch, but not an Ethernet network.

First problem of the classical model of network

Different logical communication links can share physical communication links.

Simultaneous communications from A to B and from C to D. Logically: no interferences; In practice: one needs to know the network topology to be able to predict what is going to happen.

Bad predictions of contentions.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.

Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.
Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links. Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.
- At time $t+5$ arrives a task needing 1 minute of communications and 19 minutes of computations.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.
Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.
- At time $t+5$ arrives a task needing 1 minute of communications and 19 minutes of computations.
- Objective: average stretch (ratio between the time the task spent in the system and the time it would have spent if there had been no other task in the system)

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.

Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.
- At time $t+5$ arrives a task needing 1 minute of communications and 19 minutes of computations.
- Objective: average stretch (ratio between the time the task spent in the system and the time it would have spent if there had been no other task in the system)
- FIFO policy: $\frac{1}{2}(\frac{20}{20} + \frac{25}{20}) = 1.125$.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.
Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.
- At time $t+5$ arrives a task needing 1 minute of communications and 19 minutes of computations.
- Objective: average stretch (ratio between the time the task spent in the system and the time it would have spent if there had been no other task in the system)
- FIFO policy: $\frac{1}{2}(\frac{20}{20} + \frac{25}{20}) = 1.125$.
- Second task first: $\frac{1}{2}(\frac{26}{20} + \frac{20}{20}) = 1.15$.

Second problem of the classical model of network

It may be an advantage to share the use of the communication links.

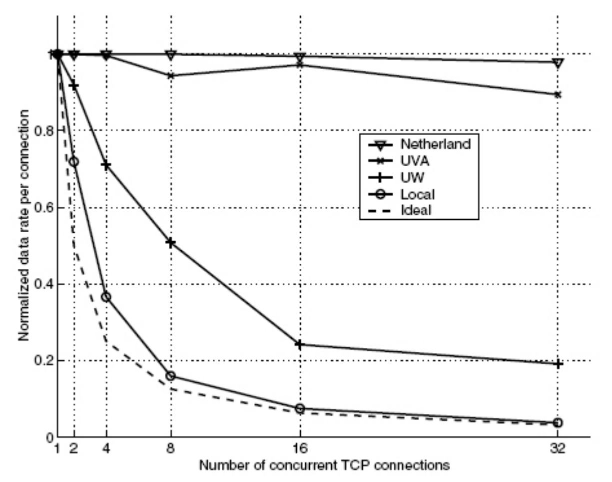
Example (a caricature)

- A server receives tasks and must spread them on some processors of same power, using a single communication link.
- At time t arrives a task needing 10 minutes of communications and 10 minutes of computations.
- At time $t+5$ arrives a task needing 1 minute of communications and 19 minutes of computations.
- Objective: average stretch (ratio between the time the task spent in the system and the time it would have spent if there had been no other task in the system)
- FIFO policy: $\frac{1}{2}(\frac{20}{20} + \frac{25}{20}) = 1.125$.
- Second task first: $\frac{1}{2}(\frac{26}{20} + \frac{20}{20}) = 1.15$.
- Fair bandwidth sharing: $\frac{1}{2}(\frac{21}{20} + \frac{21}{20}) = 1.05$.

Bandwidth sharing: classical model

- We have supposed that when x communications share a same communication link of bandwidth B, each was receiving a bandwidth of $\frac{B}{x}$.
- This is the classical model.
- This is (usually) true on a local network (LAN).

Bandwidth sharing: in practice (1)



Bandwidth sharing: in practice (2)

- The classical model is invalid for long distance networks (WAN).
- With very long distances, all connections receive the same bandwidth than the first opened connection !

  Through a *backbone* travel a very very large number of communications: one more or one less communication does not significantly change the bandwidth allocated to each of the communications.

  Whatever the bandwidth allocated on a backbone, the amount a sender can effectively use is limited by its TCP congestion window.
- Generalizing the classical model: $\dfrac{B}{\alpha + x\beta}$.

Network model

- Machines are not linked by backbones: the communication uses a certain number of local links before reaching the backbone(s) used for the long distance communication.

Network model

- Machines are not linked by backbones: the communication uses a certain number of local links before reaching the backbone(s) used for the long distance communication.
- The bandwidth of a communication using several links is defined by the "slowest" link.

Network model

- Machines are not linked by backbones: the communication uses a certain number of local links before reaching the backbone(s) used for the long distance communication.
- The bandwidth of a communication using several links is defined by the “slowest” link.
- The limiting factor may be the machine network card or the local network.

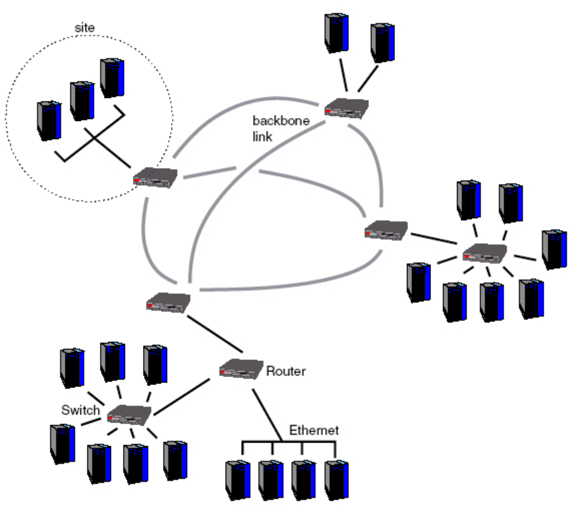
Network model

- Machines are not linked by backbones: the communication uses a certain number of local links before reaching the backbone(s) used for the long distance communication.
- The bandwidth of a communication using several links is defined by the "slowest" link.
- The limiting factor may be the machine network card or the local network.
- It is necessary to consider the local topology if several machines from a same site may communicate simultaneously.

Network model

- Machines are not linked by backbones: the communication uses a certain number of local links before reaching the backbone(s) used for the long distance communication.
- The bandwidth of a communication using several links is defined by the "slowest" link.
- The limiting factor may be the machine network card or the local network.
- It is necessary to consider the local topology if several machines from a same site may communicate simultaneously.
- TCP behavior: on a congested link, the different communications receive bandwidths which are inversely proportional to their round-trip-times.

Network model



Outline

The problem

- A graph $G = (S, A)$.
  The edge $a \in A$ has a maximal bandwidth $b(a)$.

- A set $\mathcal{R}$ of paths in the graph G.
  Our problem is to allocate a bandwidth λ_r to each path $r \in \mathcal{R}$

- Respecting the available bandwidths:

$$\forall a \in A, \sum_{r \in \mathcal{R}, a \in r} \lambda_r \leq b(a),$$

where $a \in r$ means that the path r uses the edge a.

The problem

- A graph $G = (S, A)$.
  The edge $a \in A$ has a maximal bandwidth $b(a)$.
- A set $\mathcal{R}$ of paths in the graph G.
  Our problem is to allocate a bandwidth λ_r to each path $r \in \mathcal{R}$
- Respecting the available bandwidths:

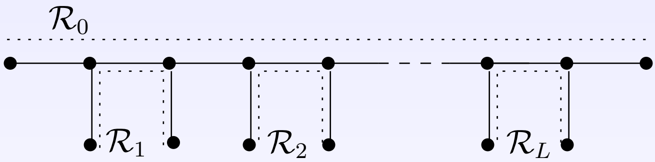
$$\forall a \in A, \sum_{r \in \mathcal{R}, a \in r} \lambda_r \leq b(a),$$

where $a \in r$ means that the path r uses the edge a.

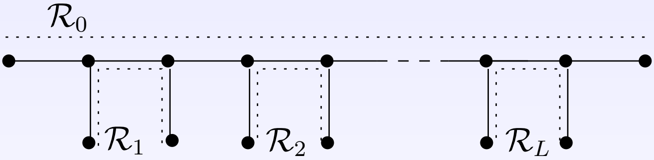
How should we allocate the bandwidths ?

Maximizing the total throughput

Maximizing the total throughput



Maximizing the total throughput



In an optimal solution: $\lambda_1 = \lambda_2 = ... = \lambda_L = 1 - \lambda_0$

The total throughput is equal to: $L - (L-1)\lambda_0$ and is maximal when the route $\mathcal{R}_0$ is allocated a null bandwidth !

Max-min fairness: definitions and properties

- Definition: the smallest allocated bandwidth is maximal.

Max-min fairness: definitions and properties

- Definition: the smallest allocated bandwidth is maximal.
- If no edge used by the path $\mathcal{R}_i$ is saturated

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

one can strictly increase the bandwidth allocated to $\mathcal{R}_i$.

Max-min fairness: definitions and properties

- Definition: the smallest allocated bandwidth is maximal.
- If no edge used by the path $\mathcal{R}_i$ is saturated

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

one can strictly increase the bandwidth allocated to $\mathcal{R}_i$.

Thus, there exists a path r such that $\lambda_r = \lambda_{\min}$ and such that there exists at least one edge $a \in r$ which is saturated.

Max-min fairness: definitions and properties

- Definition: the smallest allocated bandwidth is maximal.
- If no edge used by the path $\mathcal{R}_i$ is saturated

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

  one can strictly increase the bandwidth allocated to $\mathcal{R}_i$.

  Thus, there exists a path r such that $\lambda_r = \lambda_{\min}$ and such that there exists at least one edge $a \in r$ which is saturated.
- Let e be a saturated edge of r. If there exists a path r', $e \in r'$, of bandwidth $\lambda_{r'} > \lambda_r$, then one can give some of r''s bandwidth to r while having $\lambda'_{r'} > \lambda'_r > \lambda_r$.

Max-min fairness: definitions and properties

- Definition: the smallest allocated bandwidth is maximal.
- If no edge used by the path $\mathcal{R}_i$ is saturated

$$\forall a \in \mathcal{R}_i, \sum_{r \in \mathcal{R}, a \in r} \lambda_r < b(a)$$

  one can strictly increase the bandwidth allocated to $\mathcal{R}_i$.

  Thus, there exists a path r such that $\lambda_r = \lambda_{\min}$ and such that there exists at least one edge $a \in r$ which is saturated.
- Let e be a saturated edge of r. If there exists a path r', $e \in r'$, of bandwidth $\lambda_{r'} > \lambda_r$, then one can give some of r''s bandwidth to r while having $\lambda'_{r'} > \lambda'_r > \lambda_r$.

  Therefore, there exists a path r such that $\lambda_r = \lambda_{\min}$ and such that there exists an edge $a \in r$ satisfying: $\forall r', a \in r' \Rightarrow \lambda_{r'} = \lambda_{\min}$.

Max-min fairness: allocation

- Minimal allocated bandwidth:

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

Max-min fairness: allocation

- Minimal allocated bandwidth:

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

- Is the solution unique ?

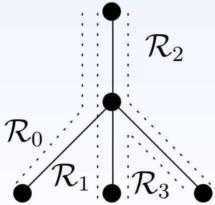
Max-min fairness: allocation

- Minimal allocated bandwidth:

$$\lambda_{\min} = \min_{a \in A} \frac{b(a)}{|\{r \in \mathcal{R} \mid a \in r\}|}.$$

- Is the solution unique ?

  Obviously not:

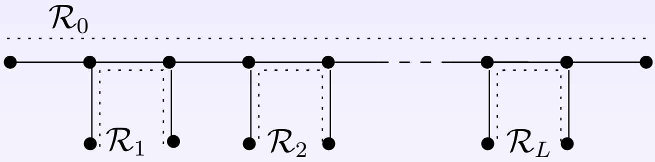


  If all edges are supposed to have a bandwidth of 1, $\lambda_0 = \lambda_1 = \lambda_2 = \frac{1}{3}$ but λ_3 can take any value in the interval $[\frac{1}{3}; \frac{2}{3}]$.

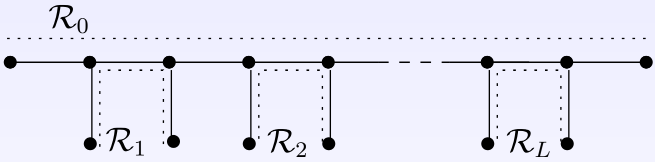
Max-min fairness: the algorithm

Algorithm: we recursively apply the max-min fairness principle: we determine the edges which define the minimal allocated bandwidth, we allocate the corresponding bandwidth to all routes using these edges and one call recursively the algorithm on the remaining paths using the updated available bandwidths.

Back to the example



Back to the example



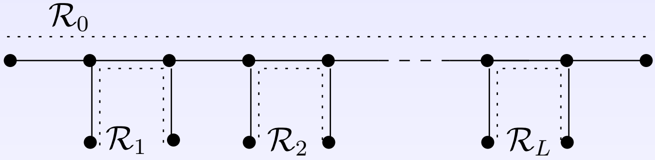
In an optimal solution: $\lambda_1 = \lambda_2 = ... = \lambda_L = \lambda_0 = \frac{1}{2}$

The total throughput is equal to: $\frac{L+1}{2}$ (which is far below the optimal value of L).

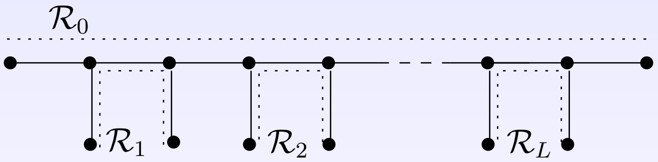
Proportional fairness

- Definition: one want to maximize $\sum_{\mathcal{R}} \log \lambda_{\mathcal{R}}$.
- Closer to TCP behavior.

Back on the example



Back on the example



In an optimal solution: $\lambda_1 = \lambda_2 = ... = \lambda_L = 1 - \lambda_0$.
One want to maximize: $L \log(1 - \lambda_0) + \log(\lambda_0)$.
One thus want to maximize: $(1 - \lambda_0)^L \lambda_0$.
The maximum is reached when $\lambda_0 = \frac{1}{L+1}$.

The total throughput is then equal to: $L - \frac{L-1}{L+1}$ (which is far closer to the optimum L).
The proportional fairness penalizes the long distance route for the benefit of the total throughput.

Conclusion

- A complicated reality.
- Need to have a model which enables to predict the behavior of applications.
- Need to take into account the topology and, more generally, the behavior of networks.
- The solution needs to be adapted to the targeted platforms and applications.