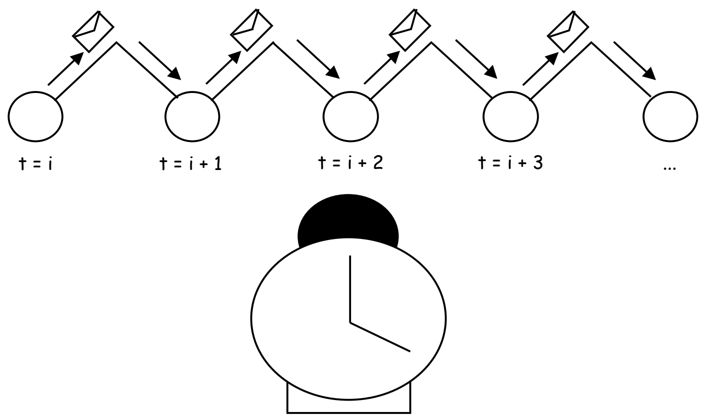
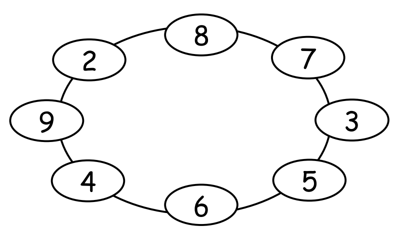
Systèmes Synchrones



Élection dans les Anneaux Synchrones: Surmonter la Limite Inférieure avec "Speeding"

Idée Générale: Les messages voyagent à différentes vitesses

- Il n'est pas nécessaire de connaître n
- Version unidirectionnelle
- synchrone



Nous supposons que le processus débute simultanément, mais ce n'est pas nécessaire

Deux façons d'éliminer les identificateurs:

- Comme dans AS FAR, les grands identificateurs sont arrêtés par les plus petits identificateurs

-Les petits identificateurs voyagent **plus vite** pour rattraper les identificateurs plus grands et les éliminer.

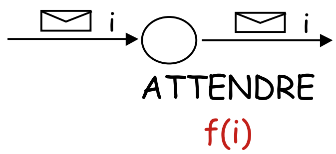
Chaque message voyage à une vitesse qui dépend de la valeur de l'identificateur qu'il contien.

L'identificateur **i** voyage à une vitesse **f(i)**

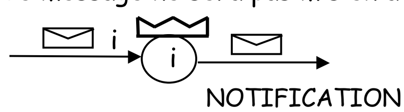
On suppose que la vitesse est unitaire et identique pour chaque message. Comment peut-on la changer ?

En introduisant les **DÉLAIS appropriés**

Lorsqu'un noeud reçoit un message contenant *i*, il attend f(i) unités de temps.

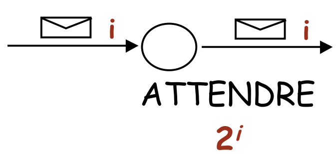


Lorsqu'un noeud reçoit son propre identificateur, il devient le leader, et envoie un message de notification aux autres noeuds de l'anneau. Ce message ne sera pas mis en attente.

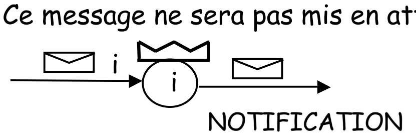


E.g., $f(i) = 2^i$

Lorsqu'un noeud reçoit un message contenant *i*, il attend 2^i unités de temps.



Lorsqu'un noeud reçoit son propre identificateur, il devient le leader, et envoie un message de notification aux autres noeuds de l'anneau. Ce message ne sera pas mis en attente.



En $2^i n + n$ unités de temps, le plus petit identificateur i traverse l'anneau

Soit $i+1$ le deuxième plus petit identificateur (avec un temps d'attente de 2^{i+1}).
Combien de liens a-t-il le temps de traverser (au plus) pendant que le plus petit identificateur effectue un tour complet ?

$(2^i n+n) / 2^{i+1}$ = au plus $n/2$ liens

Complexité

Le plus petit identificateur i fait le tour de l'anneau le plus rapidement

Messages:	n	$O(n)$
Temps:	$2^i n + n$ unités	$O(2^i n)$

Le deuxième plus petit identificateur: (pire de cas: $i + 1$)

$(2^i n + n) / 2^{i+1} < (n / 2)$ liens

A traversé $(2^i n + n) / 2^{i+1} < (n / 2)$ liens en $2^i n + n$ unités de temps

Le troisième plus petit identificateur: (pire de cas: $i + 2$)

$(2^i n + n) / 2^{i+2} < (n / 4)$ liens

A traversé $(2^i n + n) / 2^{i+2} < (n / 4)$ liens en $2^i n + n$ unités de temps

Complexité

Le j^e identificateur

$2^i n+n) / 2^{i+j} < (n/2^j)$ liens

A traversé $(2^i n+n) / 2^{i+j} < (n/2^j)$ liens en $2^i n + n$ unités de temps

Nombre Total de Messages

$$\sum_{j=1}^{n-1} n/2^j = n \sum 1/2^j = O(n)$$

Temps d'exécution Total

$$\boxed{O(2^i n)}$$

BITS	TEMPS
$O(n \log Id)$	$O(2^i n)$

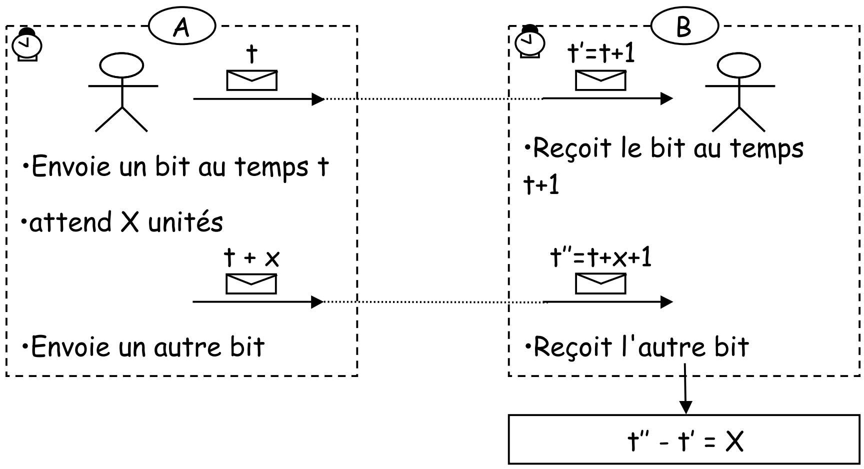
i : plus petit identificateur
Id : plus grand identificateur

Surmonter les coûts de transmission: communication à 2 bits

Toute information peut être transmise en utilisant 2 BITS

Essayez de transmettre la valeur 1.384.752.600 avec 2 bits

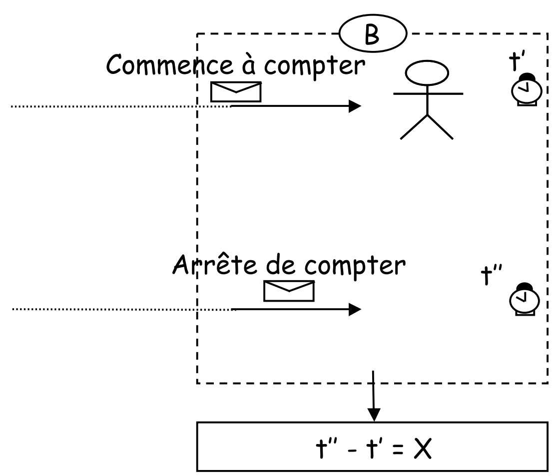
A veut envoyer la valeur X à B.



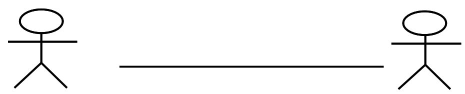
BITS: 2
TEMPS: X

Le silence a une valeur informative

Le protocole

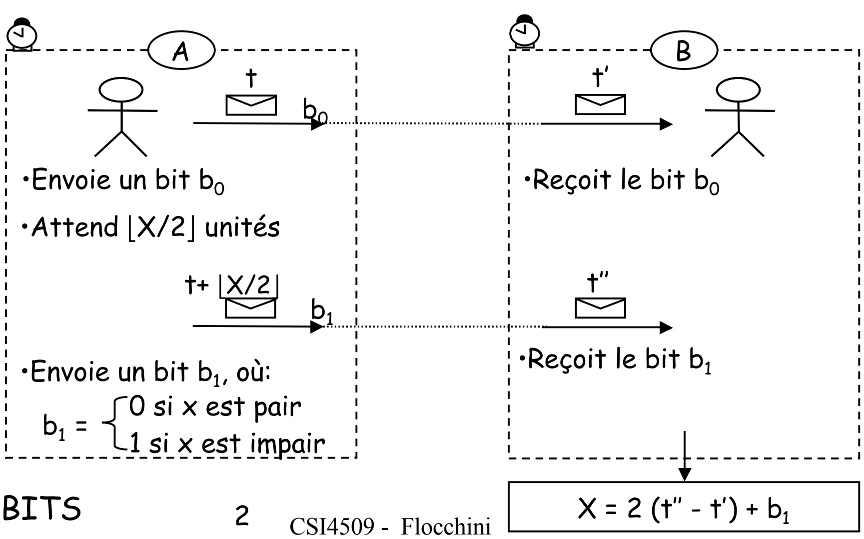


Communication entre 2 entités



Communication à 2 bits

A veut envoyer la valeur X à B.

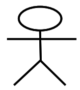


BITS 2

TEMPS $\lfloor X/2 \rfloor$

A veut envoyer la valeur X à B.

A



t

b_1

- Envoie b_1 au temps t, où:

$$b_1 = \begin{cases} 0 \text{ si } x \text{ est pair} \\ 1 \text{ si } x \text{ est impair} \end{cases}$$

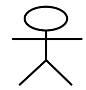
- Attend $y = \lfloor x/4 \rfloor$

b_2

- Envoie b_2, où:

$$b_2 = \begin{cases} 0 \text{ si } \lfloor x/2 \rfloor \text{ est pair} \\ 1 \text{ si } \lfloor x/2 \rfloor \text{ est impair} \end{cases}$$

B



t_1

b_1

- Reçoit b_1 au temps t_1

t_2

b_2

- Reçoit b_2 au temps t_2

$$X = 2 \cdot (2 \cdot (t_2 - t_1) + b_2) + b_1$$

BITS 2

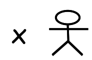
TEMPS $\lfloor X / 4 \rfloor$

BIT	TEMPS
2	X
2	$\lfloor X/2 \rfloor$
2	$\lfloor X/4 \rfloor$
3	?
4	?
...	...

Communication à 3 bit

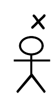
X désire transmettre $I \in \mathbf{Z}^+$ à son voisin Y. X peut:

•Envoyer un bit (b_0) pour avertir le voisin d'un début de transmission



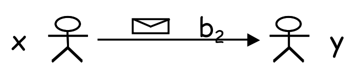
Attendre $q_0 = \lceil\sqrt{I}\rceil$ unités de temps

• Envoyer une autre *bit* (b_1) pour l'avertir que la première partie de la transmission est complété



• Attendre $q_1 = \lceil\sqrt{I}\rceil^2 - I$ unités de temps

• Envoyer un autre bit (b_2) pour l'avertir que la transmission est complété.



Le receveur doit résoudre l'équation suivante pour décoder l'information:

$$I = q_0^2 - q_1$$

BITS	TEMPS
3	$O(\lceil\sqrt{I}\rceil)$

Example:

I = 23 $\lceil\sqrt{I}\,\rceil$ $\lceil\sqrt{I}\,\rceil^2 - I$

$q_0 = \lceil\sqrt{23}\,\rceil = 5$ $q_1 = 25\text{-}23 = 2$

t = 0 Envoie b_0 reçoit b_0 $t_0 = 1$

Attends 5

Envoie b_1 reçoit b_1 $t_1 = 6$

Attends 2

Envoie b_2 reçoit b_2 $t_2 = 8$

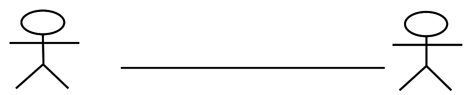
$$\boxed{(t_1 - t_0)^2 - (t_2 - t_0)}$$

25 - 2 = 23

Communication à k bits

BITS	TEMPS
k	$O(\lceil k\, I^{1/k} \rceil)$

Les voisins peuvent communiquer entre eux grâce à cette échange de bits

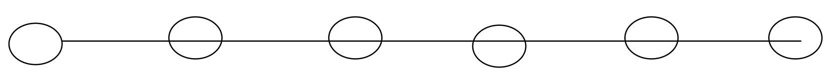


Technique PIPELINE

Pour communiquer à distance

Example: Technique précédente Avec Intermédiaires

message: "30"



Bit de départ

0

Attendre 30

Bit de terminaison

30 → 31

Attendre 30

61 → 62

Attendre 30

92 → 93

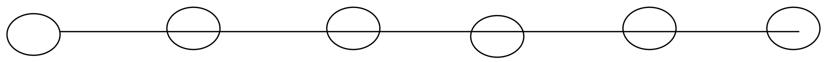
...

Attendre 30

123 → 124

Avec PIPELINE

message: "30"



Bit de départ

0 → 1 → 2 → 3 → 4 → 5

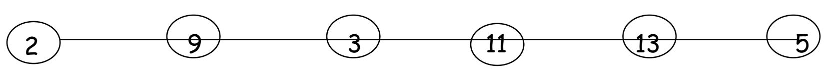
Attendre 30

Bit de terminaison

30 → 31 → 32 → 33 → 34 → 35

PIPELINE

Example: communiquer le maximum



Bit de départ

0 → 1 → 2 → 3 → 4 → 5

Attente (2) (9)

2 $\xrightarrow{+1}$ 3 $\xrightarrow{+1}$ 11 $\xrightarrow{+1}$ 12 $\xrightarrow{+1}$ 15 $\xrightarrow{+1}$ 18

. . . .

Attends sa propre valeur 2

Attends 7= 9-2

Aucune attente

Attends 2= 11-9

Max = 13 =18-5

Convertisseurs asynchrone-à-synchrone

En utilisant des communicateurs, tout algorithme asynchrone A peut être transformé en algorithme synchrone S.

A	S
m_A messages	$2m_A$ bits
t_A temps causal	$(2 + v_A)t_A$ unités de temps

Où v_A est l'identificateur le plus grand

En utilisant la communication à k bits, nous aurons:

BITS	TEMPS
km_A	$kt_A v^{1/k}_A$

Example - Transformation de STAGES

$$m_A = 2n \log n$$
$$t_A = n$$

Let $v_A = i$. Nous avons:

$$bsynch = 4n \log n = O(n \log n)$$
$$tsynch = (2 + i)n = O(ni)$$

note: La performance de l'algorithme transformé est supérieure à celle de "Speed" (la complexité en temps passe d'exponentiel à linéaire!)

	SPEED	Transformation
bits	$O(n \log Id)$	$O(n \log Id)$
temps	$O(n \cdot 2^{i}_{min})$	$O(n \cdot i_{min})$

Recherche du Minimum et Élection

Le coût d'une élection peut être réduit par l'utilisation de certaines techniques: waiting et guessing

Waiting

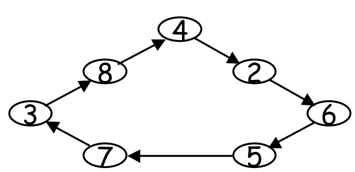
Idée: Les entités attendent un certain évènement avant de commencer

anneau synchrone

- Connaissance de n
- Identificateurs entiers
- Dans cette exemple, l'anneau est unidirectionnel mais l'algorithme s'applique aussi dans le cas d'un anneau bidirectionnel

Waiting

1. Chaque entité éveillée attend un certain temps
2. Si elle détecte aucune activité, elle devient le leader et averti les autres



L'entité ayant l'identificateur *i* doit attendre un certain temps *f*(*i*,*n*).

(Supposons, pour le moment, que toutes les entités commencent simultanément)

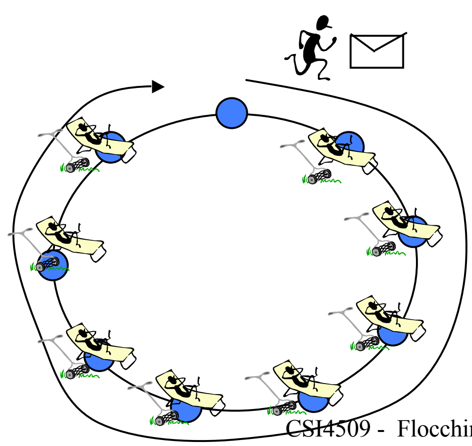
Soit x, le minimum et d(x, y) la distance entre x et y (x < y).

La fonction *f*(.,.) doit être définie de sorte que $\forall y$:

$$f(x,n) + d(x, y) < f(y,n)$$

La fonction *f*(.,.) doit être tel que, si x est le minimum, $\forall y$:

$$f(x,n) + d(x, y) < f(y,n)$$



Ceci doit être vrai dans le pire cas, c'est-à-dire lorsque

$y = x + 1$ and $d(x, x + 1) = n - 1$.

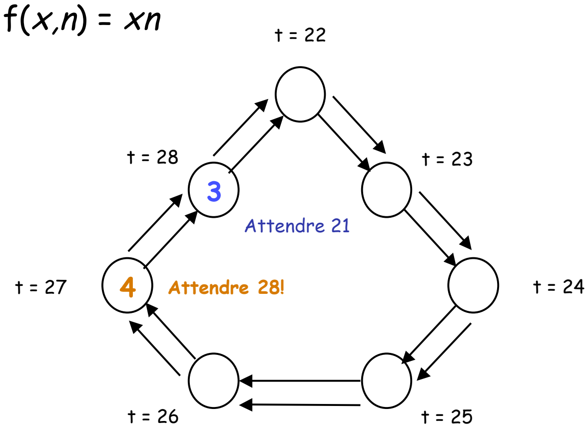
$$\begin{cases} f(0) = 0 \\ f(x+1,n) - f(x,n) > n-1 \end{cases}$$

$\boxed{\text{Une solution est } f(x,n) = xn}$

$(x+1)n - xn = xn-n-xm = n > n-1$

Example:

$f(x,n) = xn$



n = 7

x = 3

nx = 21

x + 1 = 4

n(x+1) = 28

Complexité

Bits: Seulement la plus petite identité envoie des messages — n bits

$Id_{min} \cdot n + n$

Temps:

BITS	TIME
$O(n)$	$O(Id_{min} \cdot n)$

Comparaison avec "Speed"

BITS	TIME
$O(n \log I_{max})$	$O(2^{Id_{min}}\, n)$

Sans Commencer simultanément

1) WAKE UP

Lorsqu'une entité débute spontanément, elle réveil ses voisins avant de commencer le processus d'attente.

Lorsqu'une entité inactive reçoit le message de réveil, elle le transfert à son voisin et débute le processus d'attente

t(x): temps auquel x se réveil

Pour x min et tout y

$|t(y) - t(x)| \leq d(x,y)$

donc, si $t(x)>t(y)$:

Pour tout y

$t(x) - t(y) \leq n-1$

C'est-à-dire

Pour tout y

$t(x) - t(y) < n$

Nous avons $id(x) < id(y)$

x doit finir d'attendre avant tout y et son message doit atteindre y lorsqu'il est encore en attente donc nous voulons:

$$\boxed{t(x) + f(x,n) + d(x, y) < t(y) + f(y, n)}$$

Pire cas
lorsque $t(x)>t(y)$
(et $y = x+1$)

$$t(x) + f(x,n) + d(x, y) - t(y) < f(y, n)$$

Puisque $t(y) - t(x) < n$ et $d(x, y) < n$

$$\begin{aligned} & t(x) + f(x,n) + d(x, y) - t(y) \\ &= f(x,n)+t(x)-t(y) +d(x,y) \\ &< f(x,n) +2n \end{aligned}$$

$$t(x) + f(x,n) + d(x, y) - t(y) < f(x,n) +2n$$

Nous voulons donc $f(x,n) +2n < f(y,n)$

pour le pire cas $y = x+1$

$$\begin{cases} f(0) \\ f(x,n) + 2n < f(x+1,n) \end{cases}$$

Une solution serait: $f(x,n) = 2\,n\,x$

Observations

n doit être connu

Connaître la limite supérieure de n serait suffisant

Si la valeur de n ou sa limite supérieure n'est pas connue, utiliser la fonction $f(x,n) = x^2$ fonctionnerait si:

Les identificateurs ne sont pas plus grand que n

Si une valeur n' plus petite que n est connue, la fonction $f(x,n') = x\,n'$ fonctionnera seulement si ......

Attente Universelle

1) Des WAKE-UP (messages d'amorçage) sont envoyés
2) Dès qu'une entité devient ACTIVE, elle commence à attendre f(x) unités de temps
3) Si, pendant l'attente, elle ne détecte aucune activité, x décice qu'elle est le minimum et envoie le message STOP dans l'anneau
4) Si une entité reçoit STOP lorsqu'elle attend, elle détermine qu'elle n'est pas le minimum et passe le message à son voisin

$$t(x) + f(x) + d_G(x, y) < t(y) + f(y)$$

distance entre x et y dans G

Puisque $|t(y) - t(x)| \leq d_G(x,y)$

$$\begin{cases} f(0) \\ f(x) + 2\, d_G(x, y) < f(x+1) \end{cases}$$

Ex: $f(x) = 2 \times (d_G(x, y) + 1)$

Puisque $n-1 \geq d_G$, $f(x) = 2 \times n$ serait valide

Applications de Waiting

Calculer AND et OR

Réduire le coût en temps de “Speed”

Guessing

Utilisé pour calculer le résultat d'une fonction des valeurs d'entrée sans transmettre les valeurs en question

Processus de recherche

1. Essayer d'estimer le résultat
2. Verifier l'estimation
3. Si elle est exacte, ok
4. Autrement, retourner à l'étape 1

CSI4509 - Flocchini

Example: Trouver la valeur minimale dans un anneau de grandeur connue

- Les identificateurs ne sont pas nécessairement distincts
- *n* est connu
- Les entités commencent en même temps

Si l'estimation des entités suit une séquence prédéfinie :

$$g_1, g_2, \dots g_k$$

Chaque estimation g_i est vérifié collectivement

Fonction de Vérification

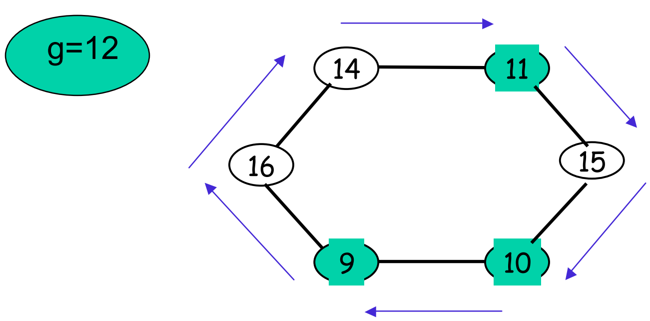
DÉCIDE(*g*)

Chaque entité compare sa valeur avec g

Si son identificateur ≤ *g*, l'entité envoie un message.

Autrement (identificateur > *g*) l'entité transfert les messages arrivant à son voisin

Example



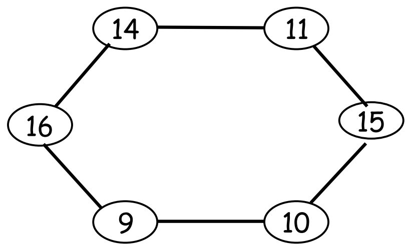
DÉCIDE(*g*)

Chaque entité compare sa valeur avec ***g***

Si valeur ≤ ***g*** envoie un message
autrement transfert les messages reçus

Example

g=7



DÉCIDE(*g*)

Chaque entité compare sa valeur avec ***g***

Si valeur ≤ ***g*** envoie un message
autrement transfert les messages reçus

Toutes les valeurs > g → SILENCE

Au moins une valeur ≤ g: → MESSAGES

Chaque entité est au courant en moins de n unités de temps

DÉCIDE(*g*)

Chaque entité compare sa valeur avec ***g***

Si valeur ≤ ***g*** envoie un message
autrement transfert les messages reçus

CSI4509 - Flocchini

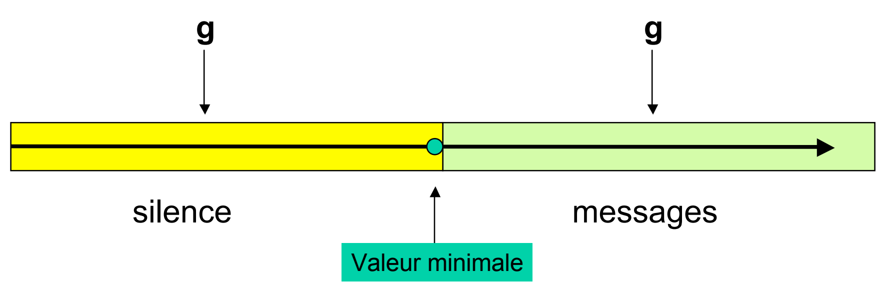
Après n unités de temps

i) Rien n'est arrivé

Tous les identificateurs sont plus grand que g

ii) Un message est reçu

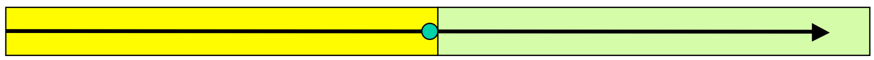
Il y a au moins un identificateur plus petit ou égal à g



Devinez: Notre estimation est **g**

Sous estimation

Surestimation



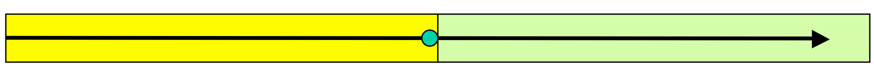
silence

messages

Devinez: Notre estimation est **g**

Sous estimation

surestimation



0 bits

n bits

n unités de temps

n unités de temps

START
Spontaneously
set alarm = c(x)+n
If id(x) ≤ g then
decision = high
send(high) to right
become(DECIDED)
Else
become(UNDECIDED)

Nous disons que c'est une surestimation (mais ça pourrait être la valeur exacte)

Je ne le sais pas

UNDECIDED
receiving(high)
decision=high
send(high)
become(DECIDED)
When c(x)=alarm
decision=low
become(DECIDED)

À la fin, tous le monde a décidé
La décision peut être "low" ou "high"

Sequence d'estimations:

$g_1, g_2, \dots g_k$

DECIDE(g_i)
choisit(g_{i+1})
DECIDE(g_{i+1})
....

DECIDE(g)

TEMPS BITS

n 0 (sous-estimation) n (surestimation)

Devinez: Pourquoi pas **g** ?

Chaque question coûte | n unités de temps

n unités de temps	
0 bits	sous-estimation
n bits	surestimation

BUT: $\Theta(n)$ BITS → O(1) surestimation

Nous voulons une stratégie qui **MINIMISE le nombre de surestimations**

Question: Que puis-je faire avec une seule surestimation ????

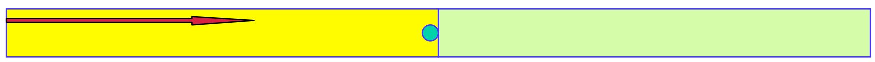
Que puis-je faire avec deux surestimations ????

UNE Surestimation Permise

Supposition: Le nombre d'estimations est entre 1 et M

1 ***Recherche Linéaire:***

M



Essayer: 1,2,3,4

Jusqu'à l'obtension d'une surestimation

Nous avons trouvé la valeur

TEMPS
$O(id_{min})$

BITS
$O(n)$

Pire cas $O(M)$

UNE Surestimation Permise

Q # d'estimations (pire cas)
K # surestimation

Recherche Linéaire:

1

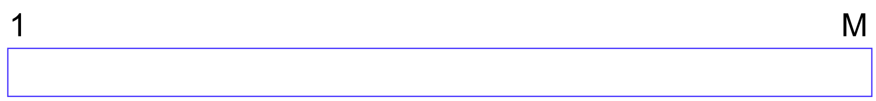
M

Recherche séquentielle

K=1 $\longrightarrow$ Q= M-1

DEUX Surestimations Permises

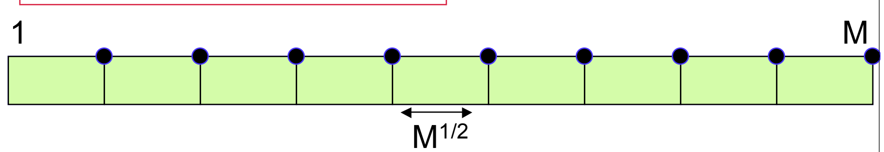
Q # d'estimations (pire cas)
K # de surestimations



K=2

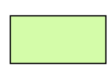
DEUX Surestimations Permises

Q # d'estimations (pire cas)
K # de surestimations

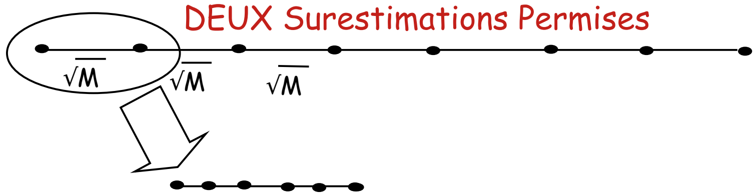


K=2 → **Q** $= 2 M^{1/2} - 2$

Recherche séquentielle sur 

K=1 Recherche séquentielle sur 

DEUX Surestimations Permises



2 $\sqrt{M}$ sous-estimations

2 surestimations

Pire Cas: dans le dernier interval

TEMPS	BITS
$O(M^{1/2})$	$O(2\,n)$

En général, la complexité est:

TEMPS	BITS
$O(M\ id^{1/k})$	$O(k\,n)$

K est constant

Est-ce la stratégie optimale ?