Ingénierie Système Prouvable pour les Systèmes Informatiques Temps Réel Critiques

Gérard Le Lann
Directeur de Recherche
Conseiller Scientifique — Défense et Sûreté des Systèmes

“The future isn’t what it used to be.” – Arthur Clarke, 1969

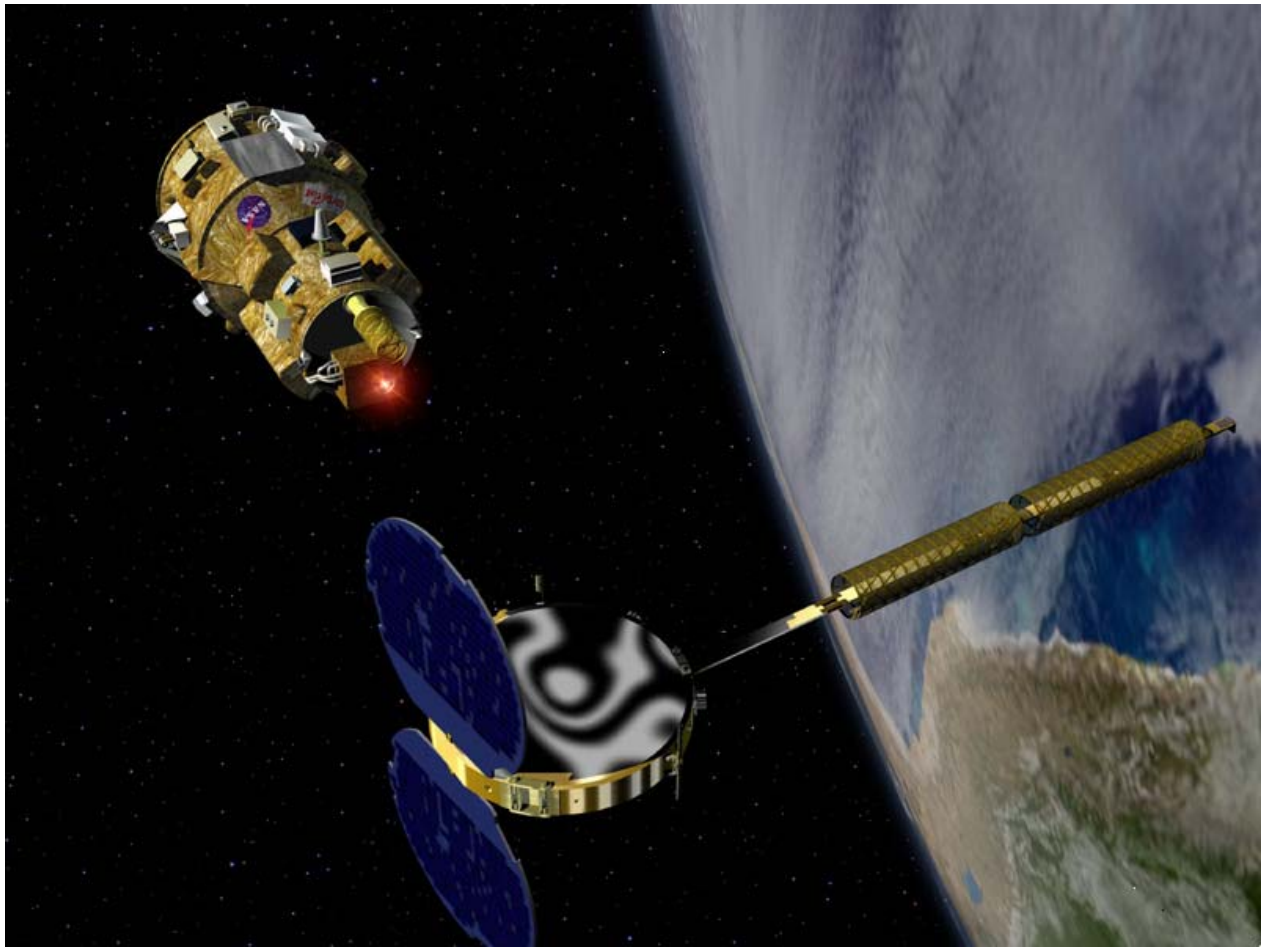
Utilité d’un tutoriel sur le sujet ? Vision globale !

Enseignement et recherche en ***systèmes*** en France ? Etat alarmant !
Quel thésard peut espérer se voir proposer une embauche à 140.000 $?
(le cas voici 6 ans, par HP Labs, pour un PhD en algorithmes distribués)

Deux domaines couverts par le tutoriel :
(1) Nature des problèmes posés par les systèmes informatiques (SI) temps réel critiques (TRC),
(2) Comment amener l’ingénierie système (IS) pour les SI au même niveau de maturité que les ingénieries « classiques » ➔ comment introduire les sciences exactes dans les méthodes et processus de l’IS pour les SI ?

Buts :
identifier les problèmes ouverts d’importance majeure ➔ « feuilles de route » pour la communauté scientifique/industrielle.

A la suite de ce tutoriel, vous (surtout les jeunes !) choisissez.



DART (NASA, 2005) : expérience de rendez-vous spatial totalement autonome

- non critique vis-à-vis de la sûreté humaine
- critique vis-à-vis de l'environnement
- critique vis-à-vis de la mission

Fut un échec (sonde et satellite sont entrés en collision)

If you need an accident to know there is a problem,

then you are part of the problem (Joe Barton).

Plan

I. Rappels et définitions

Objet de ce tutoriel : les systèmes ➔ les définitions valides sont celles établies au cours des 30 dernières années par les communautés ACM, IEEE, … (systèmes, réseaux, architectures, algorithmes, protocoles, temps réel, bases de données,…)

II. Quelques problèmes ouverts

Systèmes informatiques (SI) « traditionnels » et émergents (réseaux sans fil, mobilité, autonomie, anonymisation & authentification, …)

III. IS (pour les SI) à caractère scientifique

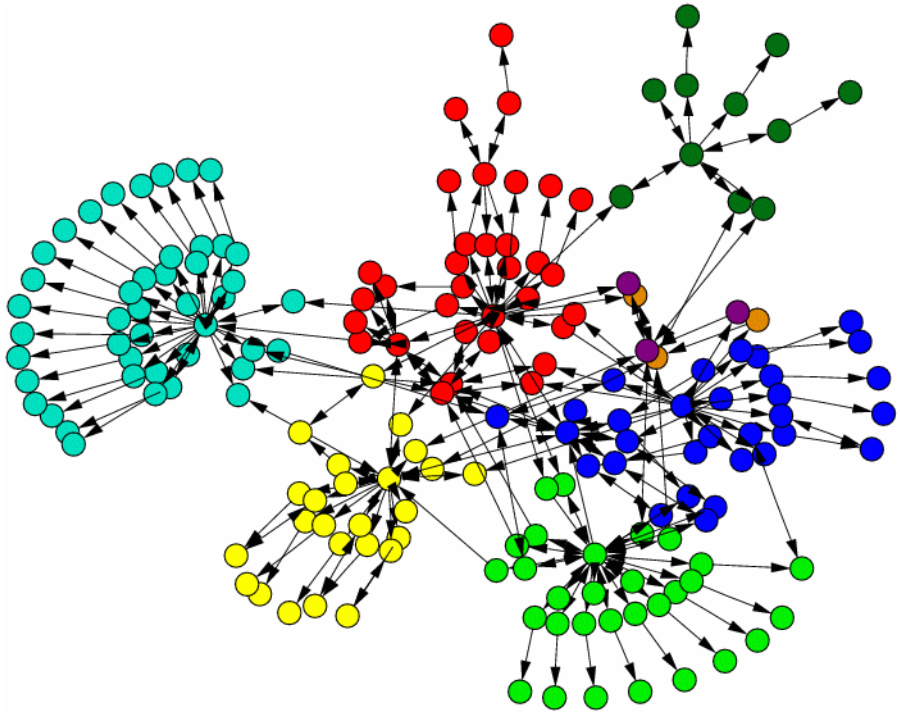
Comment permettre au monde réel (donneurs d'ordre, industrie) de s'approprier les résultats établis dans la communauté scientifique

➔ Ingénierie Système Prouvable

Système

Ensemble d'entités inter-agissantes, doté de propriétés comportementales entièrement déterminées par :

- les comportements individuels des entités qui le composent
- les règles d'interaction entre entités ➔ algorithmes & protocoles
- l'organisation topologique des entités ➔ architectures.



◈ Système

Aucune hypothèse a/s des réalisations possibles d'un SI (entités, algorithmes & protocoles, architectures) ➔ optronique, mécanique, logiciel, biochips, microélectronique, etc.

Corollaire :

métier « système » ≠ métier « logiciel »

« SI à logiciel prépondérant ». Message subliminal (pour certains) :
« seul le logiciel compte ».

L'homme est un système à eau prépondérante ➔ seule l'eau compte !

Les connaissances nécessaires sont différentes pour chacun de ces métiers.
Les mêmes termes ont souvent des définitions différentes dans ces deux métiers (distribué, asynchrone, …).

◈Temps réel (TR)

Discipline traitant de façon conjointe :

- des problèmes algorithmiques de type « décision en ligne » posés par le partage de ressources en présence de files d'attente (contention, etc.) ➔ théorie de l'ordonnancement, théorie des files d'attente, etc.
- des problèmes analytiques de type maximisation de fonctions de gain ➔ théorie des jeux, optimisation combinatoire, etc. ➔ analyses d'ordonnançabilité, conditions de faisabilité (CF).

Cas particuliers :

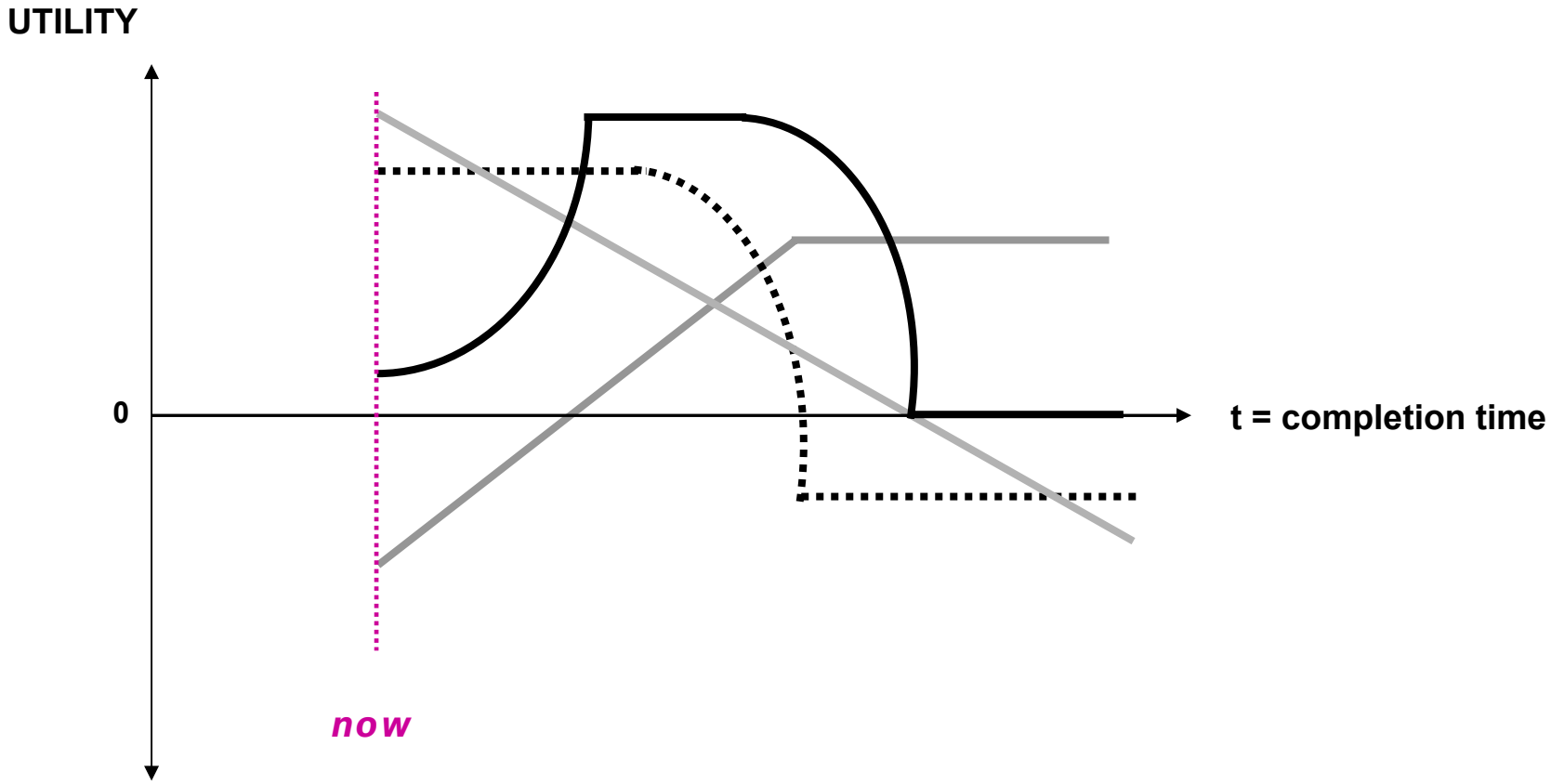
- si processus d'égale « valeur/importance », si absence de « surcharges »,
  alors temps réel ≡ tenue de contraintes temporelles strictes (p.e., échéances de terminaison au + tôt, au + tard) ➔ propriétés de ponctualité (« timeliness »).

On a le droit de dire qu'on a résolu un problème TR Ψ ssi :

- On donne un/des algorithme(s) d'ordonnancement Ω, entièrement dépendant(s) des modèles de « charges », données, processus, système, etc. spécifiés dans Ψ,
- On conduit une analyse d'ordonnançabilité donnant les CF pour le couple {Ψ, Ω}.

Four example time-utility functions

(excerpted from E.D. Jensen's WORDS'03 paper)



At time *now*, a scheduler must "find out" the ordering of execution for the 4 pending processes that maximizes the aggregated utility.

See work by D. Shasha & al. for constant time-utility functions.

Distributed access to a broadcast communication channel

Multiaccess algorithm = deterministic balanced tree search.

Permits to compute ξ^t_k = worst-case time for resolving any collision involving k message sources with t-leaf trees of branching degree m

$\Rightarrow$ Timeliness proofs for deterministic Ethernets

$$\xi_k^t = \begin{cases} \frac{m^{\left\lceil log_m\left(m\left\lfloor\frac{k}{2}\right\rfloor\right)\right\rceil}-1}{m-1} + m\left\lfloor\frac{k}{2}\right\rfloor\left\lfloor log_m\left(\frac{t}{m^{\left\lfloor\frac{k}{2}\right\rfloor}}\right)\right\rfloor - \left(k - m\left\lfloor\frac{k}{2}\right\rfloor\right) & \text{if } k \in \{2,\ldots,t\}, \\ 0 & \text{if } k = 1, \\ 1 & \text{if } k = 0. \end{cases}$$

$$t = m^n,\ m \in \mathbb{N}^* \setminus \{1\},\ n \in \mathbb{N}^*.$$

➔ worst-case time for transmitting k colliding messages is B(k) = s ξ^t_k + T(k), s = Ethernet channel slot time, T(k) = channel time for sending the k messages (back to back, collision free)

➔ worst-case feasibility conditions :: $\forall k$ B(k) < D(k)

where D(k) is channel-level transmission deadline of kth transmitted message

Distributed access to a broadcast communication channel

Ce résultat (protocole, étude analytique, prototypage) est un exemple de transfert réussi INRIA ➔ monde réel.

Les recherches qui ont conduit à la spécification des réseaux locaux de type « Ethernet déterministe » (années 80) ont été en partie financées par DCAN/Centre de Programmation de la Marine.
Le protocole variante d'Ethernet, basé sur les parcours d'arbres équilibrés, a été breveté par l'INRIA, à la demande de DCAN/CPM, brevet qui fut ensuite exploité par les industriels.

Exemples de retombées :

- **pour le domaine Défense ➔ le premier réseau local installé sur le porte-avions Charles-de-Gaulle (+ certaines frégates et sous-marins),**
- **pour le domaine civil ➔ réseaux locaux de sites de production manufacturière, du pas de tir d'Ariane (Kourou).**

Temps réel (TR) – CF

- With Time-Utility Functions:
  V(k) – k's highest utility – is given (k's TUF is known).
  Analytical lower bounds of achieved utilities – U(k) for process k – are established for every process.
  $0 < \alpha < 1$, T stands for any given time interval, K(T) = set of processes that may terminate within interval T.

$$\forall T \ \forall k \in K(T) :: \sum_T U(k) / \sum_T V(k) > \alpha$$

- With Strict Termination Deadlines:
  D(k) – k's deadline – is given.
  Analytical upper bounds on response times – B(k) for process k – are established for every process.

$$\forall k :: B(k) < D(k)$$

Temps réel (TR)

Preuves de TR ? Impliquent de spécifier les modèles d'arrivées événementielles (« charges »), pour chaque processus, notamment pour les événements « postés » par l'environnement, car les autres modèles dépendent entièrement de ces derniers. **Définitions de la communauté IEEE (RTSS, TC on RT, …) :**

périodique, sporadique, apériodique, …

Généralisations : arbitraire unimodal, arbitraire multimodal.

Recommandation: éviter de simplifier abusivement la réalité opérationnelle (confondre un modèle "commode" avec la réalité).

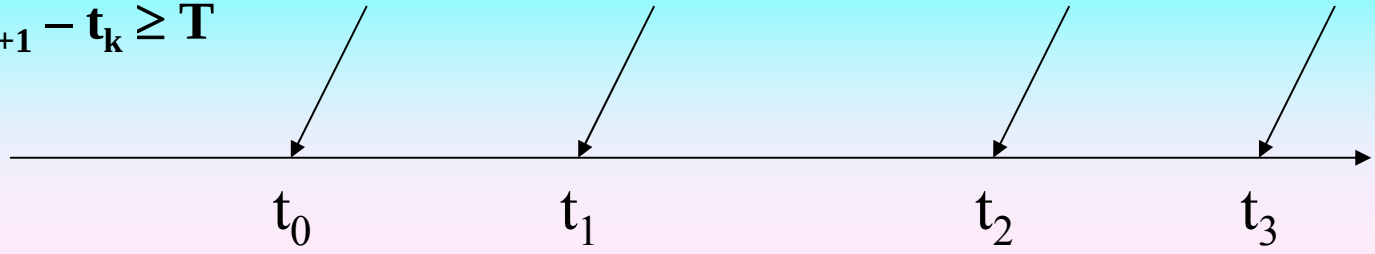
Ex 1 avec le modèle périodique :

tel phénomène est-il réellement périodique, ou bien ne l'observez-vous que de façon périodique ?

Ex 2 avec l'hypothèse « absence de surcharges » : hypothèse presque toujours invalide avec les systèmes dits complexes/autonomes (cf. NASA, etc.).

- **Sporadic (*sr*), sporadicity interval T given.**

$\forall k,\ t_{k+1} - t_k \geq T$

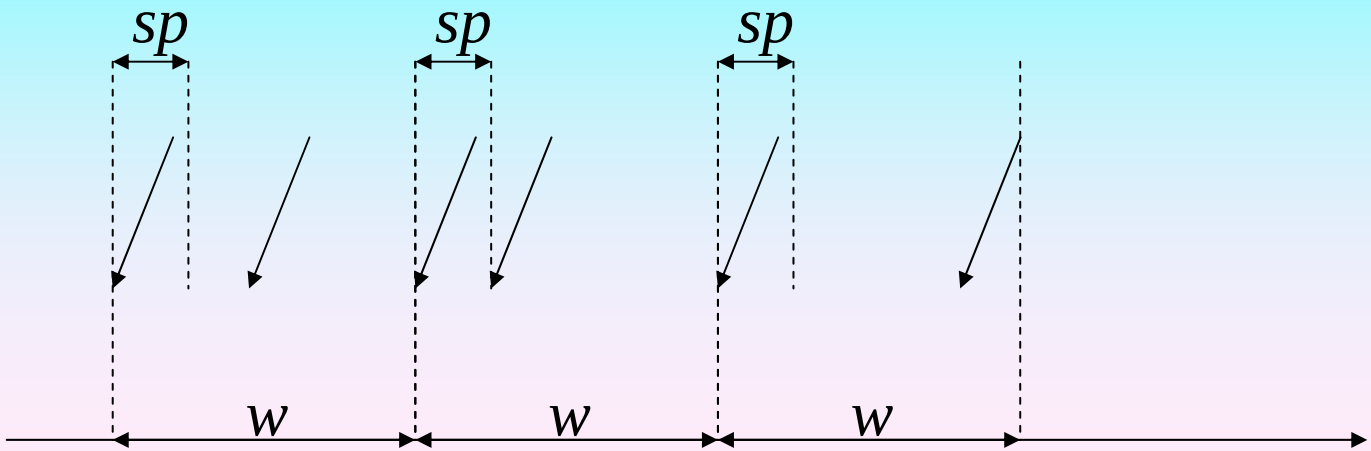


- **Unimodal arbitrary (*uarb*), defined as triple {*w*, *a* , *sp*}.**

  ***w* = sliding time window**

  ***a* = max number of arrivals within w**

  ***sp* = sporadicity interval within w**



◈ Temps réel (TR)

The arbitrary models dominate the other ones:

$$marb \supseteq uarb \supseteq sr \supseteq pr$$

$$marb \supseteq uarb \supseteq apr$$

pr :: *w* = period, *a* = 1, *sp* = nil.

sr :: *w* = sporadicity interval, *a* = 1, *sp* = nil.

apr :: *w* = ∞, *a* = 1, *sp* = nil.

➔ **Coverage(*assertion*) ≡ probability or likelihood of *assertion* not being violated in the universe under consideration.**

Illustration: *assertion* is "*model m holds true in deep space*".

I. Rappels et définitions

- **Temps réel (TR)**

  Time-Triggered model (TT) or Event-Triggered model (ET)?

- TT applicable only under restrictive assumptions
  - Group membership is advance knowledge (impossible with mobile nodes, …),
  - Ultra-reliable time base (acceptable lower bounds for precision and accuracy—hardly achievable in many MANETs),
  - Little variability w.r.t. computing/communicating times: sequential processes, fixed size messages, … (OK for simple fully predictable applications, such as, e.g., satellite launches, sensor sampling, …)

- ET applicable under all possible assumptions (unwise whenever TT is proved to yield higher efficiency, *under identical coverage*)

  Rapid deployment forces, systems of systems, VANETs, …

What about performance/efficiency?

Performance/efficiency issues: an illustration with shared multiaccess communication channels

- 10 message sources
- Every source generates a sequence of 10 messages, cyclically
- A sequence consists of 9 messages of duration 1 each,

and 1 message of duration 10, sent in any order

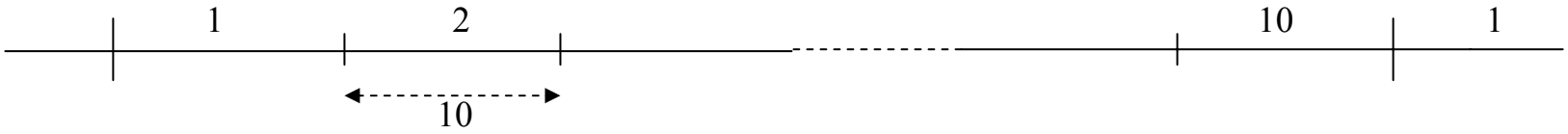
- Up to k messages may be submitted concurrently, $1 \leq k \leq 10$

TT multi-access scheme = conventional Static_Sync_TDMA

⇒ a pre-computed cyclic frame of 10 consecutive time slots, each of duration 10

w = protocol overhead per frame + 10 inter-slot “guard times”

Efficiency ratio (w ignored) = 0.19

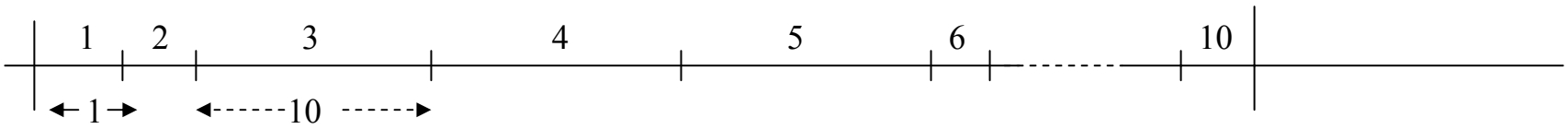


Deterministic Ethernets = conventional Async_TDMA

$\Rightarrow$ CSMA + deterministic contention resolution

w(k) = protocol overhead per 10 messages (see s ξ^t_k slide 10, where s is in the order of 40 μs with Ethernets)

Efficiency ratio (w(k) ignored) = 1



Not surprisingly, deterministic Ethernets win over TTP channels.

This simple example is an illustration of well-known results in **computer networking**.

Reason?

TT/Sync performance figures are max{max} functions, whereas ET/Async performance figures are min{max} functions

I. Rappels et définitions

◈ Temps réel (TR) – Corollaire des définitions

temps réel ≠ sûreté logique (« safety »)

On n'établit pas des expressions analytiques calculables de bornes supérieures de temps de séjour en files d'attente en faisant appel aux techniques relevant de la logique mathématique (sauf à considérer des modèles de processus, de données, d'architectures, de calcul/système extrêmement restrictifs).

◈ Criticité (C)

Existence d'une contrainte $\pi > 1 - 10^{-x}$, $\pi \equiv$ probabilité de fonctionnement correct.

Différents π pour différents sous-systèmes (propriétés/fonctions) dans un système donné (SIL 1 à 4, niveaux A à E en DO-178B). Exemple de la disponibilité :

[contrôle de trafic aérien : x = 7 ; informatique embarquée avionique civile : x = 9]

➔ **Tout modèle/hypothèse posé/e pour concevoir ou prouver (un système, un sous-système) doit avoir un taux de couverture > π.**

I. Rappels et définitions

◈ Temps Réel Critique (TRC)

Les propriétés TR (maximisation de gains, ponctualité) sont assurées avec une probabilité au moins égale à π.

Failles ou attaques possibles ne peuvent être ignorées (sauf cas improbables).

Failles ou attaques ➔ retards ➔ défaillances partielles ➔ retards ➔ défaillances partielles ➔ retards ➔ etc.

Donc ***TRC implique*** – au moins – sécurité et ***sûreté de fonctionnement (SdF)***.

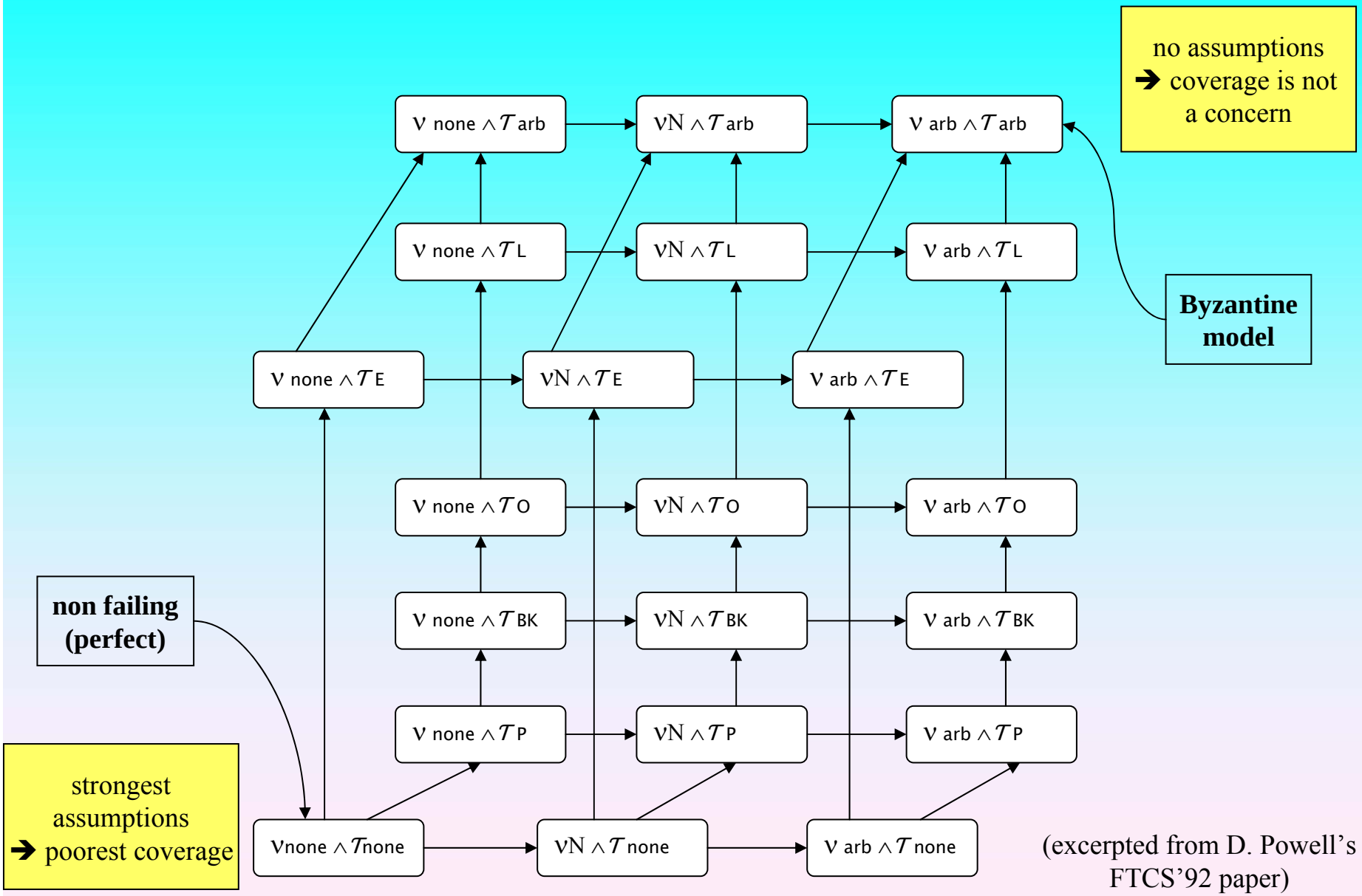
◈ Sûreté de fonctionnement (SdF)

Pas de défaillance de plusieurs entités dues à une cause unique ➔ redondance (à l'identique, diversifiée). Redondance informationnelle, spatiale, temporelle. Nécessairement exploitée/gérée sans recours à une entité unique/centralisatrice

➔ **SdF implique distribution**

Preuves de SdF ? Impliquent de spécifier les modèles de défaillances considérés (choix contraints par π).

A lattice of failure models



(excerpted from D. Powell's FTCS'92 paper)

SdF – propriétés usuelles

Fiabilité : Comportement conforme à spécification (malgré fautes et erreurs)

Disponibilité : Entité ou Service accessible sur demande

Atomicité : Tout ou rien

Cohérence : Invariants définis sur données partagées jamais violés

Isolation : Pas « d'effet de bord » (« orphan writes », …)

Durabilité : Les écritures/modifications sont rémanentes

SdF & C – propriétés usuelles

Sûreté-innocuité : non dangerosité. Ne pas confondre avec sûreté logique.

NB : Fiabilité n'implique pas sûreté-innocuité.

Sécurité : immunité aux comportements défaillants (accidentels) ou intrusifs (intentionnels) d'autrui.

NB : requis dans les systèmes à niveaux multiples de sécurité.

TRC implique distribution :

- **soit parce que TRC implique SdF,**
- **soit par nature.**

◈ Système distribué

Définition des communautés ACM (PODC, J. ACM, …) et IEEE (ICDCS, TC on DC, …) :

*** état global inconnu,**

*** concomitance des événements et des transitions d'état,**

*** les pas de calcul ou les transitions ne sont pas supposés atomiques**

Preuves de sûreté logique, de vivacité ? Impliquent de spécifier les modèles de calcul/système considérés (choix contraints par π).

Définitions des modèles *(communautés ACM (PODC, J. ACM, …) et IEEE (ICDCS, TC on DC, …)) :* ***connaissance préalable postulée concernant les délais (étapes de calcul, de communication).***

I. Rappels et définitions

- **Modèle synchrone pur (full synchrony)**

  Tous les délais sont supposés finis et bornés, bornes inf. et sup. connues.

- **Modèle asynchrone pur (full asynchrony)**

  Délais supposés finis, mais aucun délai n'est supposé borné.

- ***Modèles partiellement synchrones/asynchrones (partial synchrony, augmented asynchrony, …)***

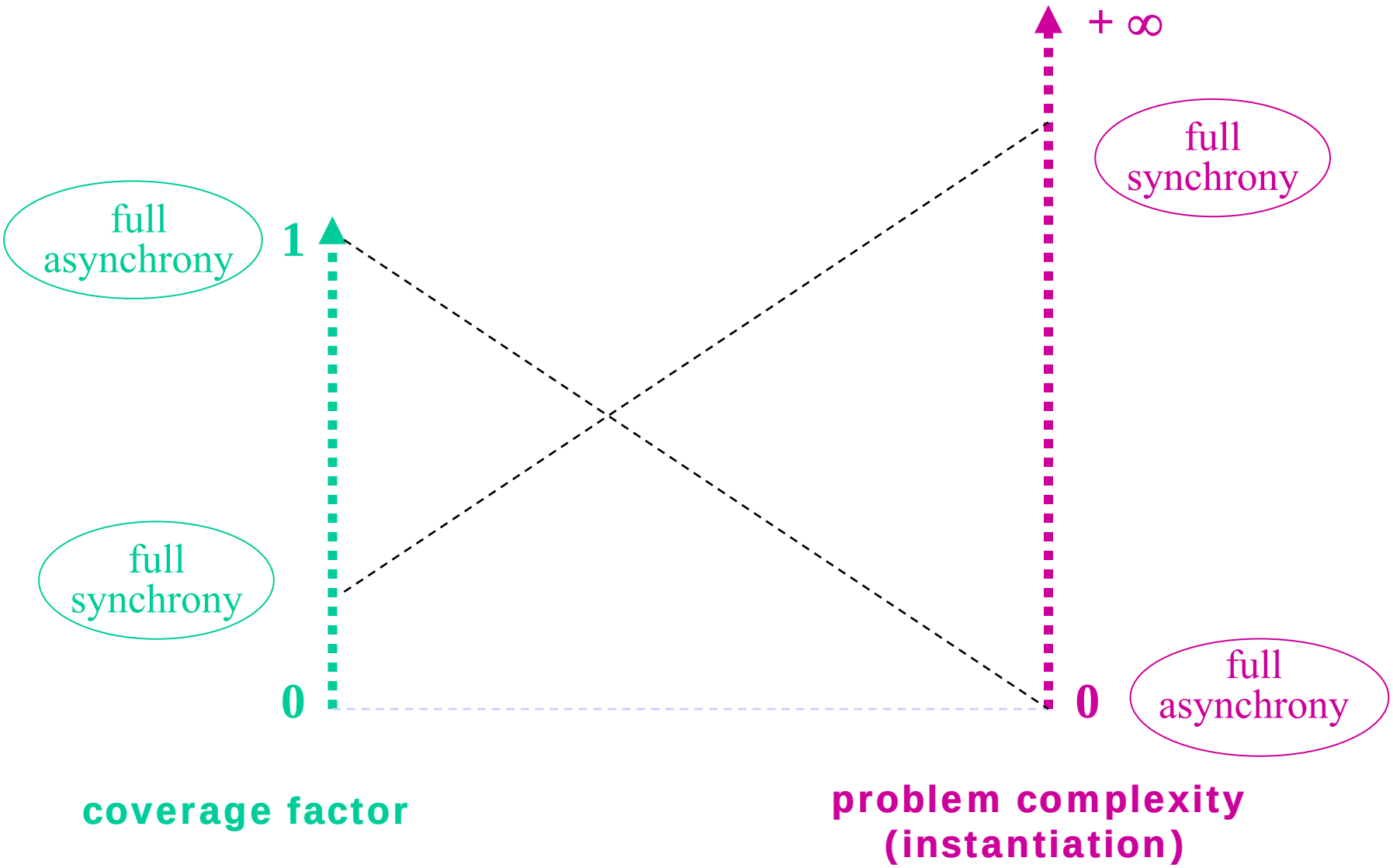
  Modèles intermédiaires, les plus pertinents/intéressants pour les vrais systèmes (ceux qui doivent être déployés et utilisés).

Ex. de définitions incompatibles :

« asynchrone » dans GALS, qui signifie « sporadique ».

Noter que du point de vue TR (analyse d'ordonnançabilité pires cas), il n'y a aucune différence entre « périodique » et « sporadique » étant donné les modèles de processus considérés en GALS ➔ pas de « meilleures » CF avec GALS qu'avec un modèle synchrone pur !

System models



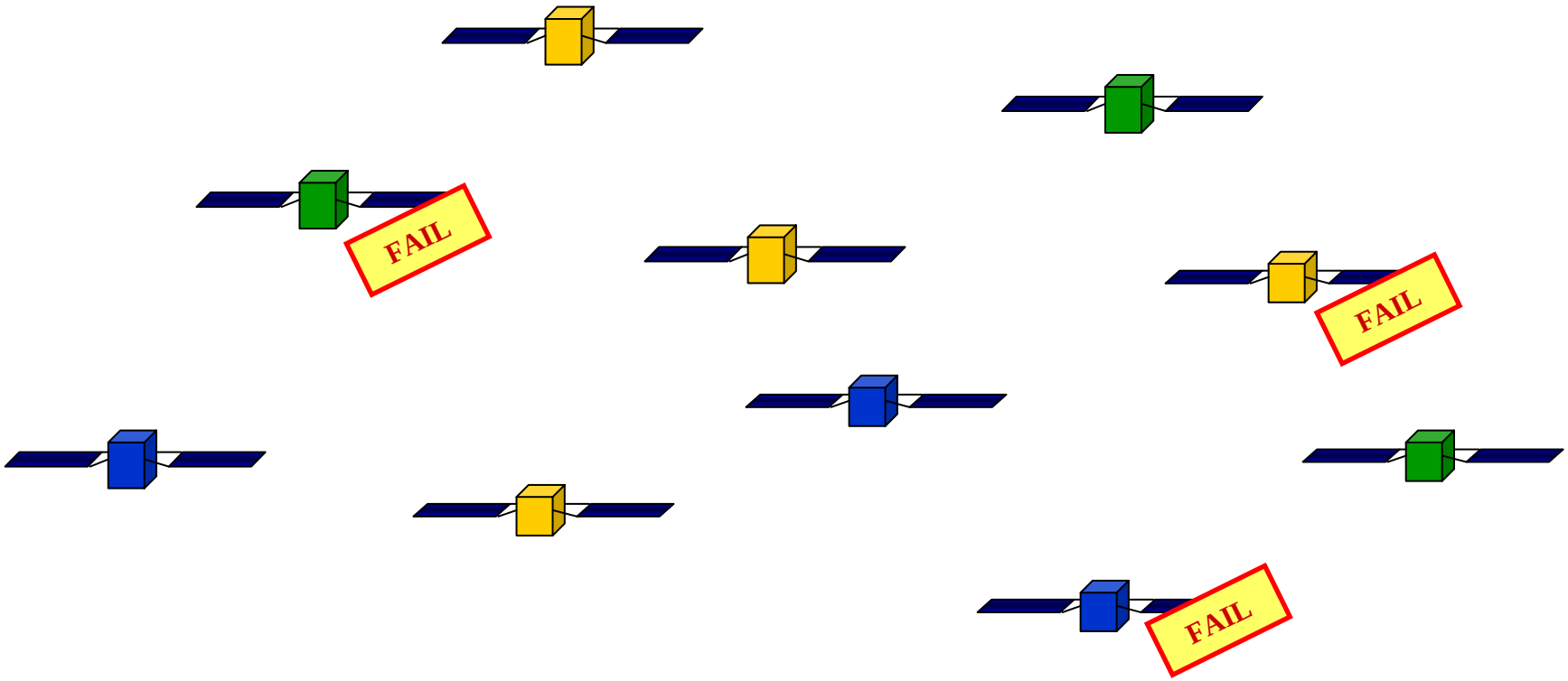
Nancy Lynch, MIT :
Distributed Algorithms (Morgan Kaufman pub.), 1996, 872 pages

"It is impossible or inefficient to implement the synchronous model in many types of distributed systems."

The synchronous model is a particular case of any other model:

full asynchrony $\supseteq$ *partial asynchrony/synchrony* $\supseteq$ *full synchrony*

Next: Examples of safety-, mission-, life-, critical systems or systems of systems which cannot be modeled after synchrony assumptions.



Autonomous near space/deep space exploration

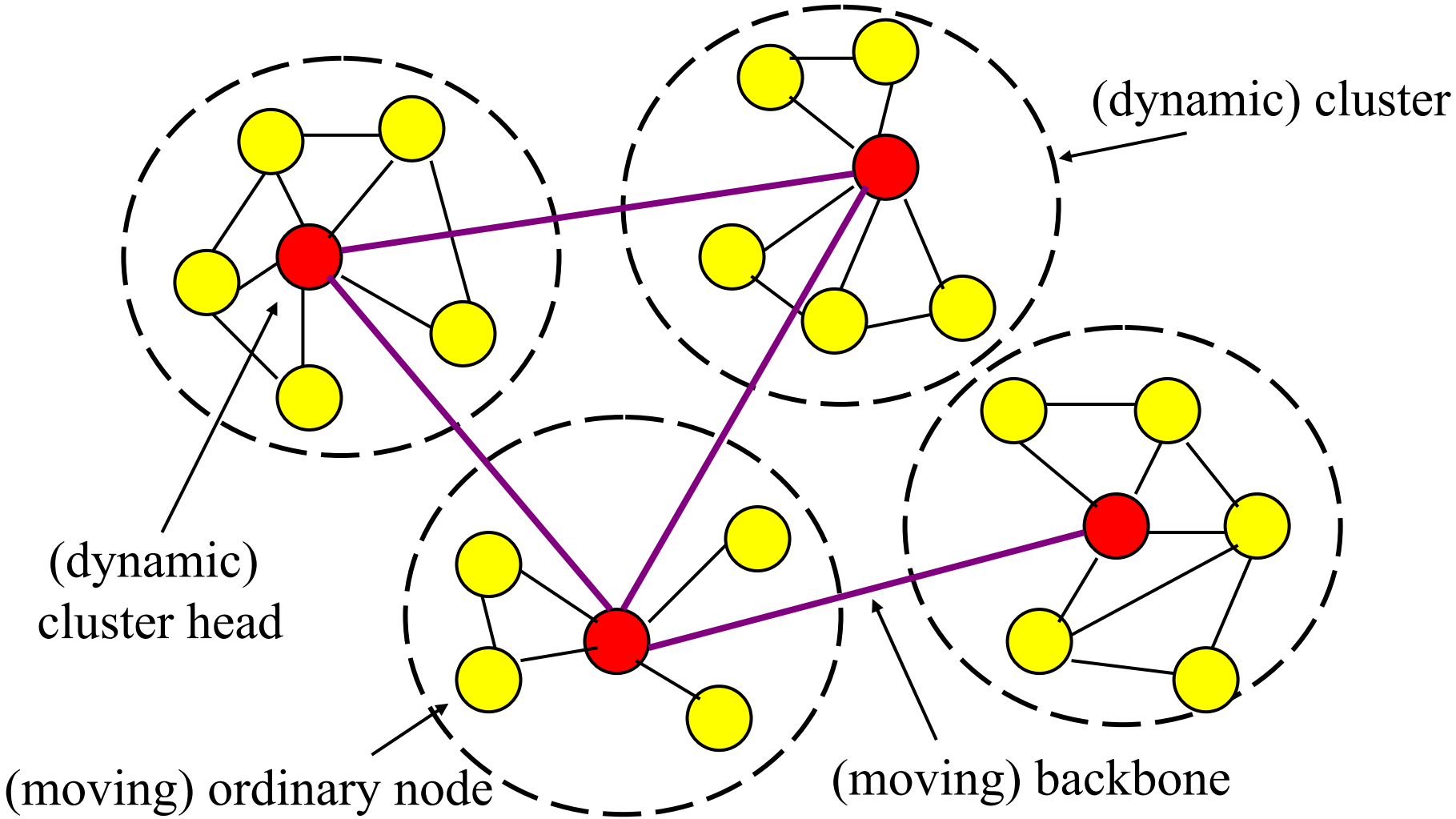
Avionique

1996 : « Dans 10 ans, tout le Rafale sera programmé en Esterel »…

On en est loin ➔ Evident : les langages synchrones ne permettent pas de « tout faire » (gérer l'événementiel, la distribution/modularité (propre à l'IMA), garantir la SdF, etc.).

MANETs: moving nodes, intermittent radio link disruptions, anonymity might be required, protection against masquerading, highly variable channel throughput, …

Coverage of any prediction on delay upper bounds???



Examples of partially synchronous/asynchronous models:

- **Pure asynchrony & ratio upper bound/lower bound Θ known for some delays**
  - ➔ the Θ–Model [Le Lann, Schmid, 2003]

Crash, omission, timing, process failures are accommodated.

Interesting feature with many systems: Θ is not violated when (postulated) upper bound is violated.

Of interest for safety-critical applications.

- **Pure Asynchrony & Unreliable Failure Detectors** [Chandra-Hadzilacos-Toueg, 1996]

An FD is a distributed oracle that provides hints about the operational status of processes. ➔ LS: list of suspected processes at every process. However, hints may be incorrect.

Crash process failures are accommodated.

FD defined after abstract properties

- Strong Completeness
  - Eventually, every process that crashes is permanently suspected by ***every*** correct process
- Weak Completeness
  - Eventually, every process that crashes is permanently suspected by ***some*** correct process

- Strong Accuracy
  - No process is suspected before it crashes
- Weak Accuracy
  - **Some** correct process is never suspected
- Eventual Strong Accuracy
  - There is a time after which correct processes are not suspected by any correct process
- Eventual Weak Accuracy
  - There is a time after which **some** correct process is never suspected by any correct process.

Failure Detectors Classes

Completeness	Accuracy			
	Strong	Weak	Eventual Strong	Eventual Weak
Strong	*Perfect* *P*	*Strong* *S*	*Eventually Perfect* ◊*P*	*Eventually Strong* ◊*S*
Weak	*Q*	*Weak* *W*	◊*Q*	Eventually Weak ◊W

◊S: Weakest FD for solving Consensus

Relations SdF/distribué/TR

Un problème « difficile » en SdF est celui de la détection des **vraies** défaillances en temps borné calculable, prédictible, si possible optimal.

Ainsi, que l'application considérée soit ou ne soit pas « temps réel »,
il est impératif d'exprimer des bornes supérieures de temps de réponse (valeurs des réveils qui servent à détecter des défaillances, latence pour obtenir un consensus, …) et donc de conduire des analyses d'ordonnançabilité …

► **La SdF en systèmes distribués pose des problèmes d'ordonnancement temps réel**

Plan

I. Rappels et définitions

Objet de ce tutoriel : les systèmes ➔ les définitions valides sont celles établies au cours des 30 dernières années par les communautés ACM, IEEE, … (systèmes, réseaux, architectures, algorithmes, protocoles, temps réel, bases de données,…)

II. Quelques problèmes ouverts

Systèmes informatiques (SI) « traditionnels » et émergents (réseaux sans fil, mobilité, autonomie, …)

III. IS (pour les SI) à caractère scientifique

Comment permettre au monde réel (donneurs d'ordre, industrie) de s'approprier les résultats établis dans la communauté scientifique

➔ Ingénierie Système Prouvable

II. Quelques problèmes ouverts

Exemples de problèmes génériques résolus en SI redondants/distribués :

* Accord approché (lectures de capteurs/horloges,…),
* Exclusion mutuelle non bloquante,
* Diffusion ordonnée,
* Consensus,
* Election de leader,
* Vues valides de groupes (group membership),
* Sérialisabilité en présence de données partagées rémanentes modifiables et d'exécutions non atomiques de processus,
* Terminaison atomique non bloquante.

Riche état de l'art : résultats de possibilité, optimalité, impossibilité, pour de nombreux éléments du produit cartésien {modèles de défaillances partielles x modèles de calcul/système} **pour chaque problème pris isolément**.

II. Quelques problèmes ouverts

- **Safety & Liveness**
  - Mutual exclusion: Bakery algorithm, Circulating Token, Logical Timestamps
  - Serializability: 2PL-TO, MVCC
- **Timeliness**
  - Fixed priorities: HPF, HPF/PI, HPF/PC
  - Deadline-driven: EDF, LLF, D-Over
  - TUF-driven: Trav Sman, EDF (quasi-concave TUFs), CM
- **Dependability**
  - Approximate agreement: Byzantine Generals
  - Consensus: Rotating Coordinator, Frequency Voting
  - Fragmentation & Dissemination (erasure codes)

« Vrais » systèmes : problèmes composites (p.e., SdF & sécurité & serialisabilité & temps réel). On ne sait pas grand-chose !

II. Quelques problèmes ouverts

- **Problèmes SI (redondants, distribués) TRC**

  Pour tous ces problèmes, même pris isolément, « surchargés » de contraintes temporelles/fonction de gain : état de l'art extrêmement limité !

Mine de problèmes ouverts.

Toute solution à venir d'un double intérêt majeur (théorique, pratique).

Croyance : modèles de calcul/système non synchrones et TR sont antagonistes.

Faux ! Il est bien évidemment possible de considérer un algorithme asynchrone (où ni temps ni durées n'apparaissent), de le « plonger » dans un système donné S, doté de capacités physiques connues, et de mener une analyse d'ordonnançabilité ➔ ponctualité (CF).

Si modélisation des capacités physiques de S est « violée » :

- **En synchrone, on perd toutes les propriétés,**
- **En asynchrone, on ne perd pas les propriétés de sûreté logique.**

Consensus

- A set of n processes in a distributed system want to reach agreement, e.g., about:
  - The value of a sensor reading,
  - Whether to accept/reject the results of a computation,
  - …/…
  - Abstractly, on a value in some set V.
- Each process starts with some initial value in V, and processes want to decide on a unique value in V.
- Properties:

  Validity: Any value decided by a process is a value proposed

  Agreement: No two processes decide differently

  Termination: Every correct process eventually decides

The twist: ***A (presumably small) number of processes (f) might be faulty, and might not participate correctly in the algorithm.***

Note: Consensus is harder than Mutual Exclusion

Un algorithme distribué asynchrone fondé sur ◊S qui résout Consensus ou Election de Leader

Code for process p_k $1 \leq k \leq n$

$est_k \leftarrow$ initial value v_k;
for r from 1 to f+1 **do**
if (r = k) **then** % p_k is the coordinator of round k %
multicast <est_k>
else wait until <est_r> is received from p_r or $p_r \in LS(p_k)$ % p_r suspected by p_k %
if <est_r> received from p_r **then** $est_k \leftarrow est_r$
endif
endif
endfor
decide est_k

Aucune variable liée au temps :
➔ aucune nécessité de « prouver » un système d'horloges
➔ aucun risque de violer des invariants de sûreté logique (transitions d'état possibles ssi les messages attendus sont reçus)
➔ aucun risque de violer des invariants de vivacité (attente infinie impossible grâce aux listes de suspects (LS))

Serializability (BHG, 1987) and Composability

A serializable history of interleaved actions is an history equivalent to some strictly sequential history.

The Serializability Theorem

In a (centralized, distributed) system where:

- ***Every process is locally correct,***
- ***Some concurrency control algorithm is used,***

every possible history is serializable

Composition/composability

This theorem is extremely important from both a theoretical and a practical viewpoint.

In particular, all existing distributed database systems, all existing on-line transactional systems, as well as corresponding CORBA standards, rest upon this result.

Transactions (processes) which share (read/write) persistent data, developed separately by engineers who don't know each other, can be freely and safely composed (no data inconsistency, no loss of transaction atomicity, …).

Serializability (BHG, 1987) and Composability

Composition/composability (cont'd)

This theorem is extremely important from both a theoretical and a practical viewpoint.

More fundamentally, this result solves the composability problem in distributed systems (dist. processes, dist. shared persistent updatable data, …), i.e. where processes do not know each other (and don't want/need to) and therefore do not exchange messages among themselves (contrary to "rendez-vous" or "multicast/broadcast" paradigms).

Such a paradigm is fundamental whenever one targets:

* *open systems,*
* *reusability of processes without having to validate/verify again (a new composition).*

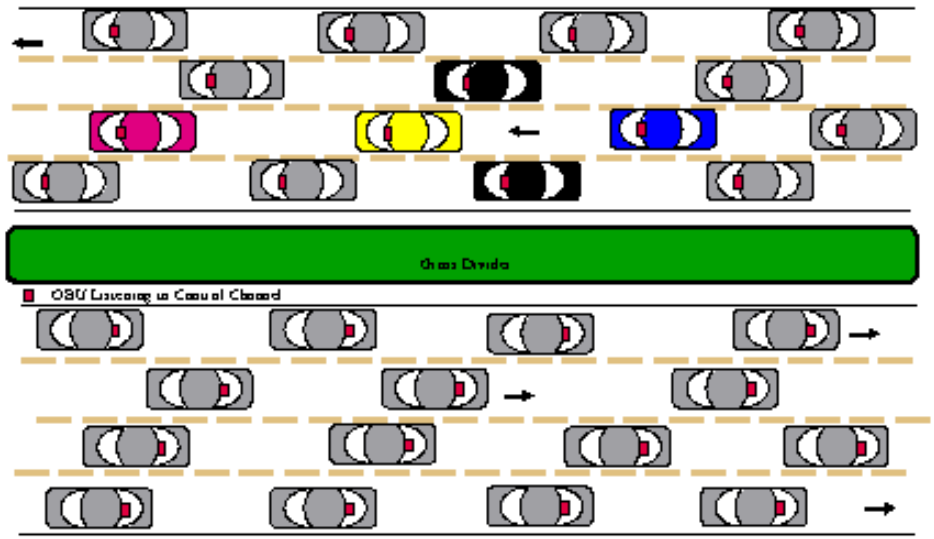
Concurrency control algorithms are early examples of « composability glue »:

- **Every process retains its local properties,**
- **Any possible collection of (locally correct) processes is endowed with proved global properties (logical safety, liveness), proofs established once forever for any given concurrency control algorithm.**

II. Quelques problèmes ouverts

Examples of VANET scenarios involving safety-critical real-time distributed decision making in highly mobile and changing groups

- **Forward Collision Warning:**
  - Host Vehicle (HV) utilizes messages from the immediate Forward Vehicle in the same lane to avoid forward collisions
- **Lane Change Assistance:**
  - Host Vehicle utilizes messages from the Adjacent Vehicle in a neighboring lane to assess unsafe lane changes
- **Electronic Emergency Brake Light:**
  - Host Vehicle utilizes messages to determine if one, or more, leading vehicles in the same lane are braking
- **Requirements:**
  - Wireless Platform
  - GPS device with ~1-1.5m resolution to properly associate vehicles with lanes



ACM VANET 2006

II. Quelques problèmes ouverts

Autonomous On-Ramp Merging at High Speed:
Consensus for deciding "where" to create a slot in rightmost lane for every car entering a highway?

II. Quelques problèmes ouverts

New frontiers in distributed real-time computing theory

- **Transient failure models**

  Lower bounds for permanent failure models to be revisited (too much pessimistic).

- **Mobile wireless networks**

  Much worse behaved than traditional wired networks:

  - No one knows who the participating processes are.
  - The set of participants may change.
  - Mobility models?
  - Unreliable communications.

- **Autonomy**

  How to merge AI, robotics (adaptive learning, mission re-planning, …) and CS?

- **Timeliness properties**

  How to compute worst-case, average-case, best-case, time bounds for system-wide decision-making processes with these new systems/models?

Recommended reading (examples):

- Journals

  Proc. ACM PODC Conf., Journal of the ACM
  Proc. DISC Symposia, Distributed Computing (Springer)
  IEEE/ACM Transactions on Networking
  Real-Time Systems Journal (Kluwer)
  Proc. IEEE Real-Time Systems Symposia

- Books

  Concurrency Control and Recovery in Database Systems, Addison-Wesley, 1987
  Distributed Algorithms, Morgan Kaufmann, 1996
  Deadline Scheduling for Real-Time Systems, Kluwer, 1998
  Fault-Tolerant Distributed Computing, Springer vol. 448, 1990

(where definitions and results can be found)

Plan

I. Rappels et définitions

Objet de ce tutoriel : les systèmes ➔ les définitions valides sont celles établies au cours des 30 dernières années par les communautés ACM, IEEE, … (systèmes, réseaux, architectures, algorithmes, protocoles, temps réel, bases de données,…)

II. Quelques problèmes ouverts

Systèmes informatiques (SI) « traditionnels » et émergents (réseaux sans fil, mobilité, autonomie, …)

III. IS (pour les SI) à caractère scientifique

Comment permettre au monde réel (donneurs d'ordre, industrie) de s'approprier les résultats établis dans la communauté scientifique

➔ Ingénierie Système Prouvable

III. IS (pour les SI) à caractère scientifique

Lacune majeure avec les méthodes actuelles, processus d'IS et les cycles de vie d'un projet/SI : pas d'obligations de valider/prouver a priori (avant réalisation). On vérifie/teste a posteriori, à l'opposé de ce qui se pratique dans les ingénieries matures et en sciences exactes. En génie civil, on valide un plan de pont avant de couler le béton et assembler les poutrelles...

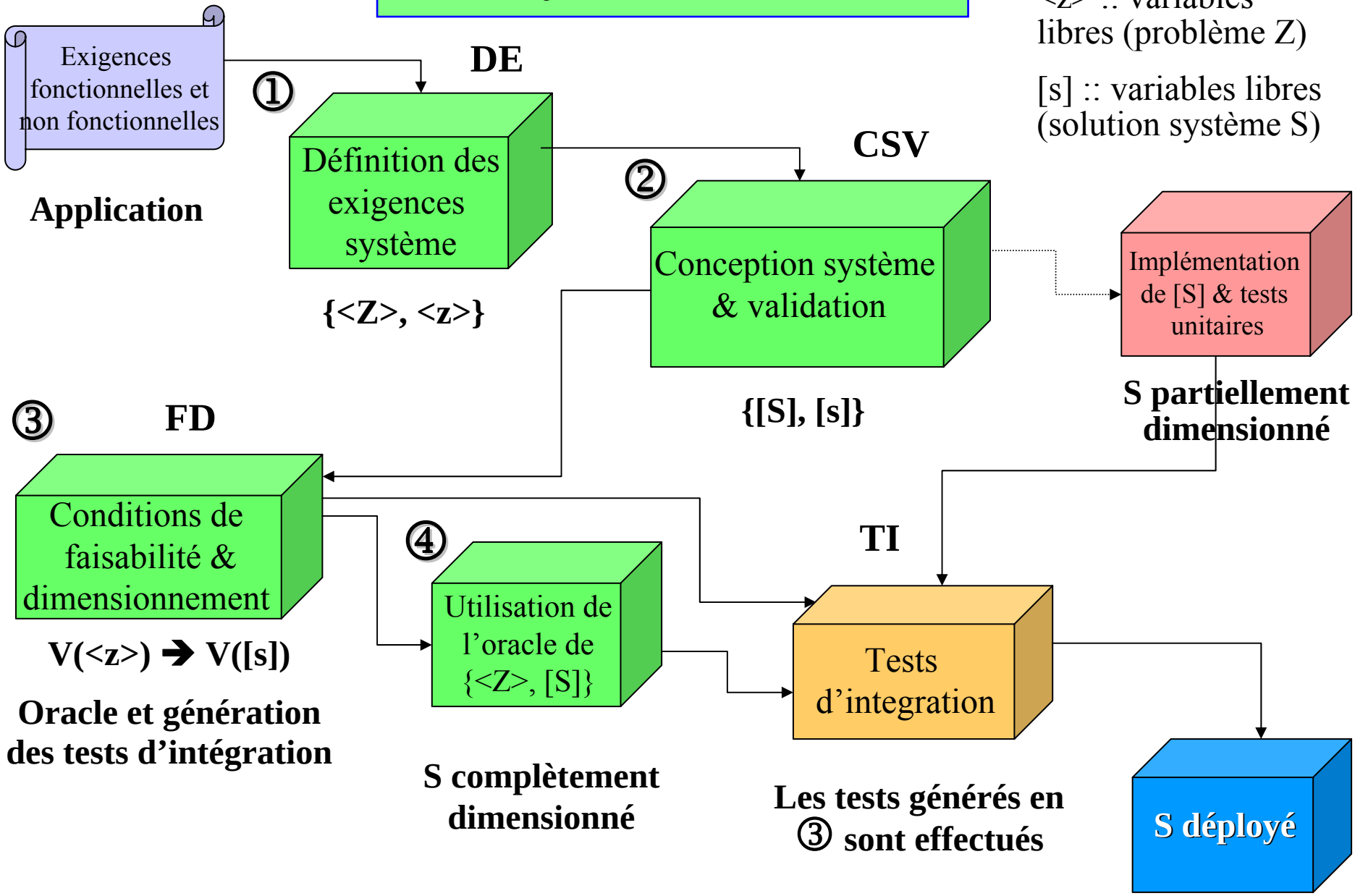
Le taux d'échecs (abandons de projets, défaillances opérationnelles, …) va croissant ainsi que les budgets afférents gaspillés, car la complexité croît, mais les méthodes d'IS restent « artisanales » (« bon sens », « expérience », etc.). Les coûts des systèmes « qui tombent en marche » restent exorbitants.

➔ Il est inévitable que l'IS pour les SI (a fortiori les SI TRC) repose un jour sur des fondations scientifiques ➔ ISP (Proof-Based System Engineering)

➔ Il est essentiel d'identifier les « maillons faibles » de la chaîne « cycle de vie » d'un SI (« feuilles de route » pour la communauté scientifique & industrielle).

Phases d'un cycle de vie en ISP ?

Cycle de vie en ISP



III. IS (pour les SI) à caractère scientifique

◈ **DE ≡ définition des exigences (requirements capture)**

Exigences fonctionnelles et non fonctionnelles du problème applicatif (langage naturel) ➔ spécification <Z> du problème informatique (composite) « enfoui » dans le problème applicatif.

◈ **CSV ≡ Conception système & validation (system design & validation)**

Problèmes mathématiques « enfouis » dans <Z> ➔ spécification [S] de la solution informatique (solution modulaire/composite) et preuves que [S] implique <Z>. Preuves : logique mathématique, …, disciplines analytiques, notamment celles utiles pour établir les conditions de faisabilité (CF) des propriétés (TR, SdF, SI, Sec, …) du couple {<Z>, [S]}. Les CF définissent l'ensemble T des tests d'intégration. Un outil (Oracle) instancie les CF.

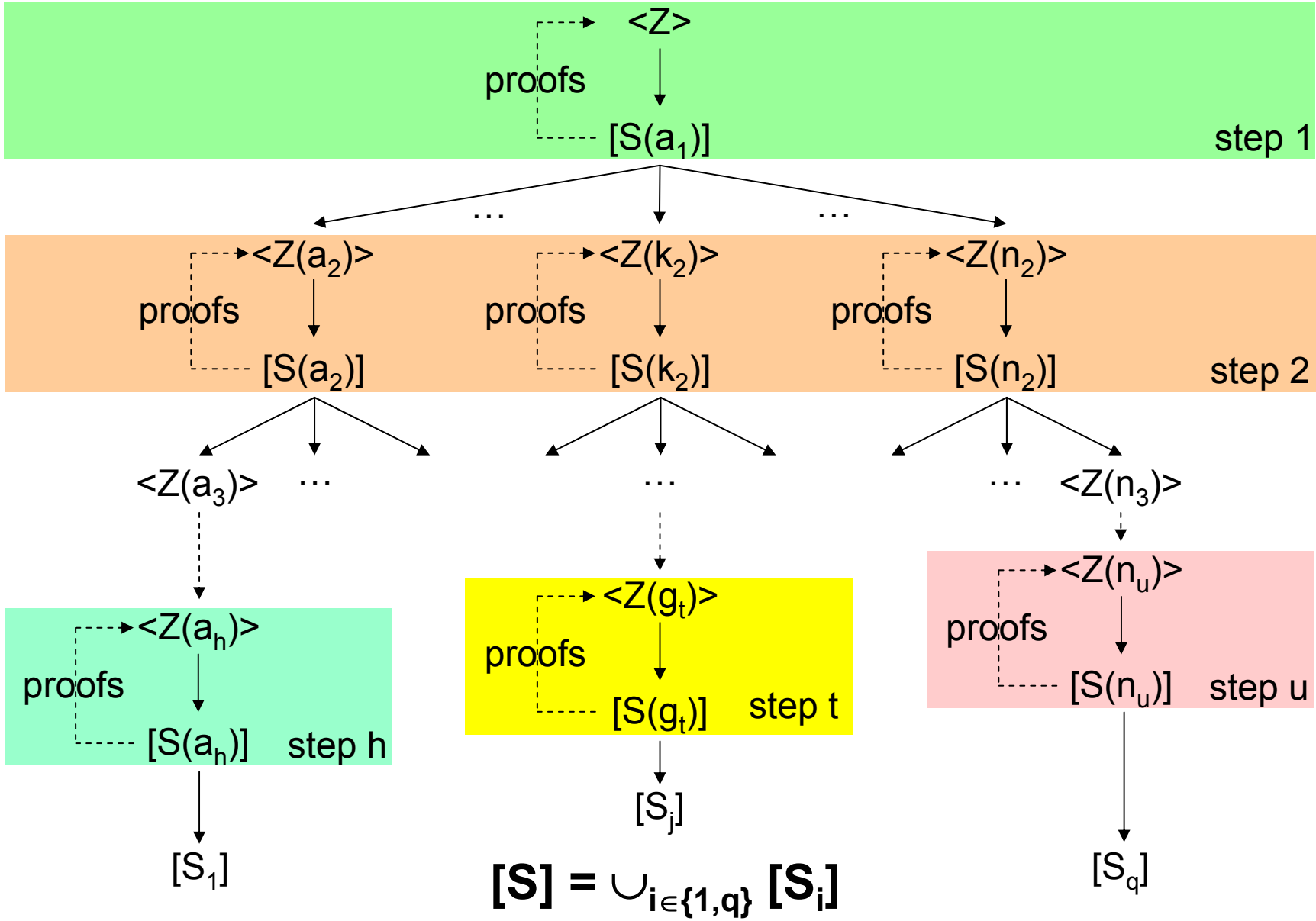
◈ **FD ≡ faisabilité & dimensionnement (feasibility & dimensioning)**

Quantification des variables libres de <Z> ➔ Oracle ➔ quantification (suffisante, nécessaire & suffisante) des variables libres de [S] – essentiel pour la réalisation de toute « release » de S (implantation finalisée (« customisée ») de [S]).

◈ **TI ≡ tests d'intégration (integration testing)**

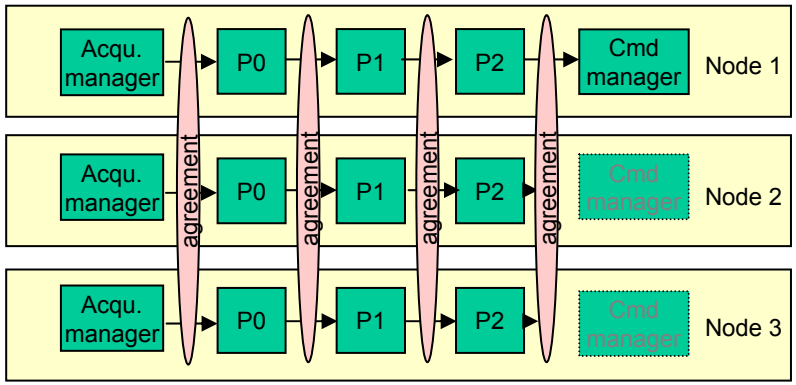
Exécution des tests de l'ensemble T (les preuves en phase CSV servent à « casser » la combinatoire des tests d'intégration).

The Design Tree (SDV phase)

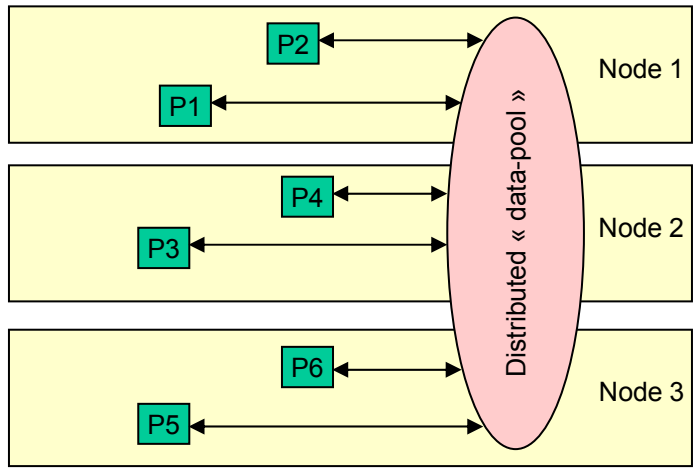


ISP/PBSE: exemple du projet A3M (contrat ESA/ESTEC – EADS Astrium)

- Distributed consistent processing under active redundancy
  - Some applications require active redundancy with error masking, in order to ensure a very high level of reliability & availability
  - Most solutions implemented today rely upon synchronous mechanisms (the programs and the computers must be tightly synchronized)



- Distributed & replicated data consistency, program serialization, program atomicity
  - Classical OBSW includes a standardized mechanism to share/exchange data between programs in a consistent manner (the data pool)
  - Existing data pool is centralized (single node)
  - A fault-tolerant data pool is a key to distribution



Drawings by courtesy of EADS Astrium

III. IS (pour les SI) à caractère scientifique

Major Past Projects & PBSE

- PHIDIAS, Air Traffic Control (proactive – RC, COTS analysis)
- A3M, Satellites (proactive – RC & SDV)
- Ariane 5/501 (post mortem analysis, endorsed by the IEEE Computer Society and the Safety Critical Forum)
- AGS, Satellites (proactive – RC & SDV)
- ORECA, Military Avionics (proactive – RC, SDV & FD)
- IPSN, Nuclear Power Plants (proactive – RC, COTS analysis)
- ASSERT IP/FP6, Aerospace (proactive – RC & SDV)*
- NCW Study (basic principles ➔ French MoD/DGA)

**INRIA has decided to withdraw at mid-course (along with other partners)*

III. IS (pour les SI) à caractère scientifique

Point dur : la phase DE

Problème applicatif ➔ DE ➔ problème informatique

Langage naturel (métier App) ➔ langage naturel restreint (métier informatique) ➔ formalisation

Fondée sur une classification des connaissances scientifiques. A ce jour, on connaît des formalismes pour certaines classes de modèles et de propriétés :

- tout problème informatique ayant une solution prouvée est un élément du produit cartésien défini sur ces classes (à enrichir),
- il existe un ordre partiel (à enrichir) dans chacune des classes ➔ définitions formelles de « complexité ». Exemples :
  - « problèmes P et R non comparables »,
  - « problèmes P et R comparables, P plus complexe que R » [rangs dans les treillis, nombre de propriétés exigées pour P supérieur à celui exigé pour R].

III. IS (pour les SI) à caractère scientifique

Objectifs de la communauté scientifique vis-à-vis de la phase DE :

(1) Transférer les connaissances scientifiques existantes
➢ extraction automatique/outillée du problème informatique composite « enfoui » dans un problème applicatif

(2) Développer les recherches en formalisation des modèles et propriétés pertinents % réalités opérationnelles et technologiques

A quoi sert-il de « faire de la preuve formelle » de solution correcte si le sous-problème considéré n'est pas le bon (vis-à-vis du problème initial Z) ?

D'où viennent les spécifications initiales du sous-problème ?

III. IS (pour les SI) à caractère scientifique

Point dur : la phase CSV

Problème informatique ➔ problèmes ∈ {théories connues}
Décomposition/analyse de <Z> bien conduite ➔ problèmes génériques (très souvent)
➔ Ré-utilisation des résultats connus (solutions & preuves), sauf si problème ouvert

Preuves en phase CSV ➔ réduction de la complexité
des tests d'intégration de plusieurs ordres de grandeur

III. IS (pour les SI) à caractère scientifique

Objectifs de la communauté scientifique vis-à-vis de la phase CSV :

(1) Transférer les connaissances scientifiques existantes

- **extraction automatique/outillée des problèmes de sciences exactes « enfouis » dans un problème informatique**
- **identification automatique/outillée des solutions connues, de leurs preuves, de leurs conditions de faisabilité**

(2) Développer les recherches sur :

- **les problèmes ouverts en algorithmique distribuée (TR, TRC)**
- **les méthodes formelles pour l'ingénierie des systèmes**
- **l'aide outillée à la conception et la validation des solutions (obligations de preuves)**

III. IS (pour les SI) à caractère scientifique

- Il existe des méthodes d'ingénierie logicielle, fondées sur des obligations de preuves et outillées, qui sont « ré-orientées » vers l'ingénierie système (B est le meilleur exemple actuel).

- Il existe des travaux scientifiques visant les méthodes conçues d'emblée pour l'ingénierie système, fondées sur des obligations de preuves (p.e., méthode TRDF de l'INRIA) ; on ne dispose pas encore d'outils.

➔ ***La situation en 2007 est celle qui prévalait voici ≈12 ans en ingénierie du logiciel***

Le point d'inflexion n'est plus très loin …