République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Abess Laghrour Khenchela

Institut des Sciences et Technologies

Département d'Informatique et Mathématiques

Mémoire

En vue de l'obtention du diplôme de Master 2 en Informatique

Spécialité : Sécurité et Technologies Web

Thème:

TOLERANCE AUX FAUTES DANS UN CONTEXTE DE NUAGE (CLOUD) POUR LES SYSTEMES EMBARQUEES

Présenté par :

- Zerzour Abderrahim

- Boumedjane Abdelwahab

Rapporteur:

-Dr : MEHALAINE Ridha

Année Universitaire 2020-2021

REMERCIEMENTS

Tous d'abord je remercie mon grand dieu pour m'avoir donnés la santé, le courage et la volonté pour achever mon travail

J'adresse le grand remerciement à mon encadreur Dr : MEHALAINE RIDHA qui a proposé le thème de ce mémoire, pour ses conseils et ses dirigés du début à la fin de ce travail, Ainsi qu'à tous les membres du jury

Nos remerciements les plus sincères aussi pour tous les enseignants de mon département informatique de l'université KHENCHELA pour leur patience et leurs efforts à la cour de ma formation

Nous remercions en particulier notre ami Walid Boukhors qui a toujours été là pour nous. Son soutien inconditionnel et ses encouragements ont été d'une grande aide

Je voudrais exprimer mes vifs remerciements à l'ensemble de nos responsables de l'université Oum el Bouaghi qui nous a facilité nos obligations professionnels

Nous voulons remercie nos collègues avec lesquels nous avons passé des moments inoubliables au cours des années théorique et pendant la réalisation de ce travail, ils ont manifesté leur soutien et ont toujours encouragé à allons au-devant

Enfin, nous remercions, également, toute personne qui a participé de près ou de loin à la réalisation de ce travail

A nos familles

Résumé :

Les systèmes embarqués temps réel sont des systèmes dont les exigences constituent un défi pour les développeurs. En plus d'offrir les fonctionnalités requises, ces systèmes doivent impérativement respecter les contraintes temps réel qui leur sont imposées.

Notre étude vise à développer une technologie qui améliore la tolérance aux pannes spécifiquement dans la phase de transmission des messages à travers le Bus (CAN 2.0 a/b), et cette technologie vise à améliorer le temps d'envoi et de réception, et à réduire le temps d'attente et le retard, et c'est à ce moment que les nœuds essaient d'envoyer et de recevoir des données en même temps sur le même canal.

Notre conception proposée tente d'assurer le bon fonctionnement des nœuds car elle garantit que le système ne tombe pas en panne, et augmente son succès et son efficacité lors de la transmission et de la réception pendant un certain moment, sans retard même s'il est encouru par un coût élevé.

Mots clé :

Systèmes embarqués, Temps réel, Bus CAN (2.0 a / b), Cloud Computing (nuage), Tolérance aux Fautes, Erreur, Nœuds, Trame.

Abstract:

Real-time embedded systems are systems whose requirements pose a challenge for developers. In addition to offering the required functionalities, these systems must comply with the real-time constraints imposed on them.

Our work is oriented to develop a technique improves the fault tolerance precisely at the phase of transmission of messages in a CAN bus (2.0 a / b), this technique aims to improve the sending time, reducing the time of transmission. 'Wait and delay time when nodes attempt to send and receive data simultaneously on the same channel.

Our proposed design tries to guarantee the proper functioning of the nodes (shareholders or detectors), deprives the probability of system failure, increases the number of messages (frames) sent and received in a given moment, deprived of latency even if it will increase a may cost.

key words :

Real-time embedded systems, CAN bus (2.0 a / b), Cloud Computing, Fault tolerance, Error nodes, Frame

الملخـــــص:

الأنظمة المدمجة في الوقت الحقيقيهي أنظمة تشكل متطلباتها تحديًا للمطورين ، بالإضافة إلى توفير الوظائف المطلوبة يجب أن تمتثل هذه الأنظمة لقيود الوقت الفعلي المفروضة عليها.

تهدف دراستنا إلى تطوير تقنية تعمل على تحسين التسامح مع الخطأ على وجه التحديد في مرحلة إرسال الرسائل من خلال الناقل(CAN 2.0 a / b) ، وتهدف هذه التقنية إلى تحسين وقت الإرسال و الاستقبال، وتقليل من وقت الانتظار والتأخير، وهذا عندما تحاول العقد إرسال واستقبال البيانات في نفس الوقت على نفس القناة.

يحاول تصميمنا المقترح ضمان الأداء السليم للعقد حيث يضمن عدم فشل النظام، ويزيد من نسبة نجاحه وفعاليته أثناء الارسال والاستقبال خلال لحظة معينة، مع عدم التأخير حتى لو انجر عليه ارتفاع في التكلفة.

الكلمـات المفتـــاحية:

الأنظمة المدمجة ,أنظمة الوقت الحقيقي, الحوسبة السحابية ،التسامح مع الأعطال العقد، الحزم، الناقل(CAN (2.0 a / b

Table des matières

Table des figures

Liste des tableaux

Liste des fonctions

Introduction générale :

L'augmentation constante des besoins en terme de puissance de calcul informatique a toujours été un dé auquel la communauté informatique a été (et est toujours) confrontée. Même si les évolutions technologiques de ces dernières années ont permis de mettre en place des machines de plus en plus puissantes, ces dernières restent très insuffisantes par rapport aux besoins en termes de calcul. Partant de cette constatation, les solutions se sont orientées vers une approche qui consiste à regrouper le plus grand nombre de machines (ou systèmes) distribuées pour fournir une puissance de calcul très élevée.

De ce fait les systèmes embarque en vue le jour, par le besoin de micro système spécialiser dont des taches précise et en temps réel, Une application temps-réel effectue des fonctions de contrôle et de pilotage. Chaque fonctionnalité est assumée par une tâche temps-réel. Certaines fonctions sont critiques et ne peuvent pas être retardées, c-à-d que les tâches correspondantes doivent impérativement avoir terminé leur traitement dans un délai donné, à partir de leur date de réveil. Dans le cas contraire, si les résultats produits sont incorrect alors la tâche est considérée être invalide. Les systèmes temps réel embarqués sont des systèmes de traitement dont la validité est conditionnée non seulement parles dates de délivrance des résultats mais surtout par es corrections de ces derniers. C'est pour ça que les techniques de tolérance aux faute atterré les développeurs, elle est définie par la capacité du système de se conformer à ses spécification sen début de présence des fautes dans n'importe lequel de ses composants (logiciels ou matériels).

Donc ce mémoire est consacré à l'étude de problèmes de communication entre les Systèmes embarques temps réel, l'utilisation de La redondance des composants matériels ou logiciels est la technique la plus efficace utilisée pour assurer la tolérance aux fautes dans le cas de collision de la transmission des trames, et le Claude Computing comme un moyen d'interaction avec le monde extérieur et concevoir une simulation de ces méthodes

Chapitre 1-

Systèmes Embarqués et Systèmes temps réel

1-Introduction :

Le système embarqué est un système informatique autonome destiné à accomplir une tâche spécifique. Son fonctionnement n'implique aucune intervention humaine. Ce qui lui offre un très vaste champ d'utilisation. Ce type de dispositif est aussi bien utilisé dans le domaine de l'aéronautique que dans la fabrication de jouets.

Le terme « système embarqué » peut être associé à la partie matérielle qu'à la partie logicielle. Dans tous les cas, ces utilitaires se présentent sous la forme d'un objet de petite taille. Il embarque un processeur ainsi qu'un système d'exploitation qui lui est propre. Ces dispositifs ont été conçus pour supporter des conditions extrêmes. C'est pour cette raison qu'ils intègrent différents appareils destinés à assurer la sécurité des particuliers.

Ces équipements ne requièrent aucun entretien et peuvent fonctionner pendant plusieurs années. Il faudra simplement prévoir une source d'énergie fiable. On peut également mettre en place un module permettant de vérifier son bon fonctionnement à distance. C'est souvent le cas dans le domaine industriel. Les responsables effectuent alors un diagnostic mensuel, hebdomadaire ou journalier pour écarter les risques de dysfonctionnements.

Une technologie indispensable dans plusieurs domaines les systèmes embarqués sont utilisés dans d'innombrables domaines. L'exemple le plus concret est sans aucun doute les supercalculateurs des voitures. Cette unité de calcul gère de nombreux composants du véhicule. Il va enclencher certains éléments en fonction des événements. C'est de cette façon que les airbags seront activés si l'on est face à une collision. De même, le freinage assisté permettra d'éviter une éventuelle perte de contrôle.

2- Systèmes embarqués :

2-1-Historique :

[FAB 12] Fin du XIX^{ème} (19$^{ème)}$ siècle : démonstration du potentiel des technologies électroniques pour les transmissions sans fil.

- Seconde guerre mondiale
  - ✓ . Application au calcul (décryptage)
  - ✓ . Premiers ordinateurs (ENIAC)
- 1947-1954
  - ✓ . Invention puis commercialisation des transistors à semi-conducteurs

- ✓ . Les « transistors » remplacent les postes radio à tubes

- Années 60 et 70: premiers circuits intégrés, LSI
- 1960-2010
  - ✓ Réduction des tailles de transistors (≃ 100000)
  - ✓ Évolution anticipée (loi de Moore)
  - ✓ Industrie au cœur de la croissance économique des 50 dernières années
  - ✓ Augmentation exponentielle des performances
- Puissance (Joy) : MIPS = 2année−1984
- Densité (Moore) : Transistors par puce = 2année−1964
- Densité des supports magnétiques «Maximal Areal Density» (Frank): MAD = 10 année−1971 10
- "Stagnation" des performances pour la rapidité d'accès aux disques → RAID (multiplications des disques) et caches.

Aujourd'hui

- Circuits mixtes analogique/digital (télécommunications)
- Microsystèmes (capteurs, actionneurs intégrés)
- Nanotechnologies (horizon 5 à 10 ans)
- Taille de gravure: 90 nm
- Coûts de R&D de plus en plus élevés
- La demande (ordinateurs, téléphones mobiles, etc.) stagne
- Peu de produits vraiment nouveaux (VHS => DVD, caméscopes => numériques, téléphones fixes => mobiles)
- Crise financière et économique des TIC

2-2-Définitions :

En reprenant la définition proposée par Tammy Noergaard [TAM 05], un système embarqué est un système informatique appliqué, qui se distingue des autres types de systèmes informatiques tels que les ordinateurs personnels (PC) ou les supercalculateurs. Cependant, vous constaterez que la définition de «système embarqué» est fluide et difficile à cerner, car elle évolue constamment avec les progrès technologiques et les diminutions spectaculaires du coût de mise en œuvre de divers composants matériels et logiciels. Ces dernières années, le domaine a dépassé la plupart de ses descriptions traditionnelles. Étant donné que le lecteur rencontrera probablement certaines de ces descriptions et définitions, il est important de comprendre le raisonnement qui les sous-tend et

pourquoi elles peuvent ou non être exactes aujourd'hui, et être en mesure de les discuter en connaissance de cause. Voici quelques-unes des descriptions les plus courantes d'un système embarqué:

1-Les systèmes embarqués sont plus limités en termes de fonctionnalités matérielles et / ou logicielles qu'un ordinateur personnel (PC). Cela est vrai pour un sous-ensemble important de la famille de systèmes informatiques embarqués. En termes de limitations matérielles, cela peut entraîner des limitations des performances de traitement, de la consommation d'énergie, de la mémoire, des fonctionnalités matérielles, etc. Dans les logiciels, cela signifie généralement des limitations par rapport à un PC - moins d'applications, des applications réduites, pas de système d'exploitation (OS) ou un système d'exploitation limité, ou moins de code au niveau de l'abstraction. Cependant, cette définition n'est que partiellement vraie aujourd'hui, car les cartes et les logiciels que l'on trouve généralement dans les PC d'hier et d'aujourd'hui ont été reconditionnés dans des conceptions de systèmes embarqués plus complexes.

2-Un système embarqué est conçu pour exécuter une fonction dédiée. La plupart des périphériques embarqués sont principalement conçus pour une fonction spécifique. Cependant, nous voyons maintenant des appareils tels que les hybrides d'assistant de données personnelles (PDA) / téléphone portable, qui sont des systèmes embarqués conçus pour pouvoir effectuer une variété de fonctions principales. En outre, les derniers téléviseurs numériques incluent des applications interactives qui exécutent une grande variété de fonctions générales sans rapport avec la fonction «TV» mais tout aussi importantes, telles que le courrier électronique, la navigation Web et les jeux.

3-Un système embarqué est un système informatique avec des exigences de qualité et de fiabilité plus élevées que les autres types de systèmes informatiques. Certaines familles d'appareils embarqués ont un seuil d'exigence de qualité et de fiabilité très élevé. Par exemple, si le contrôleur du moteur d'une voiture tombe en panne sur une autoroute très fréquentée ou si un dispositif médical critique tombe en panne pendant une intervention chirurgicale, des problèmes très graves en résultent. Cependant, il existe également des appareils intégrés, tels que des téléviseurs, des jeux et des téléphones portables, dans lesquels un dysfonctionnement est un inconvénient, mais pas généralement une situation potentiellement mortelle.

4-Certains périphériques appelés systèmes embarqués, tels que les PDA ou les tablettes Web, ne sont pas vraiment des systèmes embarqués. Il y a une discussion sur la question de savoir si les systèmes informatiques qui répondent à certaines définitions de systèmes embarqués traditionnels, mais pas à toutes, sont en fait des systèmes embarqués ou autre. Certains estiment que la désignation de ces conceptions plus complexes, telles que les PDA, comme des systèmes

embarqués est motivée par des professionnels du marketing et des ventes non techniques, plutôt que par des ingénieurs. En réalité, les ingénieurs embarqués sont divisés quant à savoir si ces conceptions sont ou non des systèmes embarqués, même si actuellement ces systèmes sont souvent discutés comme tels entre ces mêmes concepteurs.

La question de savoir si les définitions intégrées traditionnelles doivent ou non continuer d'évoluer ou si un nouveau domaine de systèmes informatiques doit être désigné pour inclure ces systèmes plus complexes sera en fin de compte déterminé par d'autres acteurs de l'industrie. Pour le moment, puisqu'il n'y a pas de nouveau domaine de systèmes informatiques pris en charge par l'industrie, désigné pour des conceptions qui se situent entre le système embarqué traditionnel et les systèmes PC à usage général.

5-Un système embarqué est un système informatique à petite échelle qui fait partie d'une machine ou d'un système électrique / mécanique plus grand. il est souvent conçu pour effectuer certaines tâches dédiées et souvent un système en temps réel. il est appelé intégré car le système informatique est intégré dans un périphérique matériel. Les systèmes intégrés sont importants, car ils sont de plus en plus utilisés dans de nombreux appareils ménagers quotidiens, tels que les montres numériques, les appareils photo, les fours à micro-ondes, les machines à laver, les chaudières, les réfrigérateurs, les téléviseurs intelligents et les voitures. Les systèmes embarqués doivent également souvent être de petite taille, peu coûteux et avoir une faible consommation d'énergie. [PER18]

2-3-Caractéristiques d'un système embarqué : [LOU 17]

2-3-1-Fonctionnement en temps réel :

-Réactivité : des opérations de calcul doivent être faites en réponse à un événement extérieur.

- La validité d'un résultat (et sa pertinence) dépend du moment où il est délivré.

- Rater une échéance va causer une erreur de fonctionnement.

2-3-2-Faible encombrement, faible poids :

-Electronique "Pocket PC", applications portables où l'on doit minimiser la consommation électrique.

- Difficulté de réaliser le packaging afin de faire cohabiter sur une faible surface électronique analogique, électronique numérique, sans interférences.

2-3-3 -Faible consommation :

- Batterie de 8 heures et plus (PC portable : 2 heures).

2-3-4-Environnement :

- Température, vibrations, chocs, variations d'alimentation, corrosion, eau, feu, radiations.
- Prise en compte des évolutions des caractéristiques des composants en fonction de la température, des radiations...

2-3-5 -Fonctionnement critique pour la sécurité des personnes. Sûreté :

- Le système doit toujours fonctionner correctement.
-Sûreté à faible coût avec une redondance minimale.
- Sûreté de fonctionnement du logiciel.
-Choix entre un design tout électronique ou électromécanique.
-Système opérationnel même quand un composant électronique lâche.

2-3-6 -Prix de revient :

Beaucoup de systèmes embarqués sont fabriqués en grande série et doivent avoir des prix de revient extrêmement faibles, ce qui induit **:**

-Une faible capacité mémoire.

-Un petit processeur (4 bits). Petit mais en grand nombre !

2-3-7 -La consommation est un point critique pour les systèmes avec autonomie :

- Une consommation excessive augmente le prix de revient du système embarqué car il faut alors des batteries de forte capacité.

2-3-8 -Faible coût :

-Optimisation du prix de revient : une exigence incontournable.

L'informatique embarquée à des impératifs différents de l'informatique personnelle (Les micro-ordinateurs). Ce sont principalement :

2-3-9-La criticité :

Les systèmes embarqués sont souvent critiques, et les systèmes critiques sont presque toujours embarqués. En effet, comme un tel système agit sur un environnement physique, les actions qu'il effectue sont irrémédiables. Le degré de criticité est fonction des conséquences des déviations par rapport à un comportement nominal, conséquences qui peuvent concerner la sûreté des personnes et des biens, la sécurité, l'accomplissement des missions, la rentabilité économique.

2-3-10- La réactivité :

Ces systèmes doivent interagir avec leur environnement à une vitesse qui est imposée par ce dernier. Ceci induit donc des impératifs de temps de réponses. C'est pour cette raison que l'informatique embarquée est souvent basée sur un système temps réel.

2-3-11-L'autonomie :

Les systèmes embarqués doivent en général être autonomes, c'est-à-dire remplir leur mission pendant de longues périodes sans intervention humaine. Cette autonomie est nécessaire lorsque l'intervention humaine est impossible, mais aussi lorsque la réaction humaine est trop lente ou insuffisamment fiable.

2-3-12-La robustesse, sécurité et fiabilité :

L'environnement est souvent hostile, pour des raisons physiques (chocs, variations de température, impact d'ions lourds dans les systèmes spatiaux,...) ou humaines (malveillance). C'est pour cela que la sécurité au sens de la résistance aux malveillances et la fiabilité au sens continuité de service sont souvent rattachées à la problématique des systèmes embarqués.

2-4-Conception des systèmes embarqués : [MEH 15]

La conception c'est faire Présenter des méthodes de vérification pour s'assurer que le logiciel d'un système embarqué travaille correctement avec le matériel et que le matériel a été correctement conçu pour exécuter le logiciel , la conception se devise en 4 phases :

a-La phase d'analyse: elle consiste à identifier les caractéristiques essentielles de toutes les solutions possibles pour engendrer le système désiré.

b-La phase de conception: c'est la phase qui ajoute des éléments à l'analyse afin de définir une solution particulière qui optimise certains critères.

c-La phase de implantation : ou la phase qui, automatiquement ou manuellement, fournit le code source de la partie logicielle du système, pour aboutir à l'exécutable ; par extension du terme, elle peut représenter aussi la phase qui fournit les composantes matérielles spécifiques, définies dans la phase de design.

d-La phase de test : une phase pendant laquelle il est vérifié que l'implantation est équivalente à la conception, et qui valide que l'implantation respecte tous les critères de correction identifiés dans la phase d'analyse.

3-Systèmes temps réel :

3-1-Définition :[SAM 14]

•Un système temps réel est un système (application ou ensemble d'applications) informatique dont le fonctionnement est assujetti à l'évolution dynamique d'un procédé extérieur qui lui est connecté et dont il doit contrôler le comportement.

•La correction d'un système temps réel dépend non seulement de la justesse des calculs mais aussi du temps auquel les résultats sont produits [Stankovic 1988] (contraintes temporelles).

•Un système temps réel n'est pas un système « qui va vite / rapide» mais un système qui satisfait des contraintes temporelles (les contraintes de temps dépendent de l'application et de l'environnement alors que la rapidité dépend de la technologie utilisée, celle du processeur par exemple).

3-2-Classification des systèmes temps réel :[IUL21]

3-2-1- Temps réel strict ou dur (hard real-time) :

La réponse du système dans les délais est vitale. L'absence de réponse est catastrophique (plus qu'une réponse incorrecte).

Exemples : contrôle aérien, contrôle d'une centrale nucléaire…

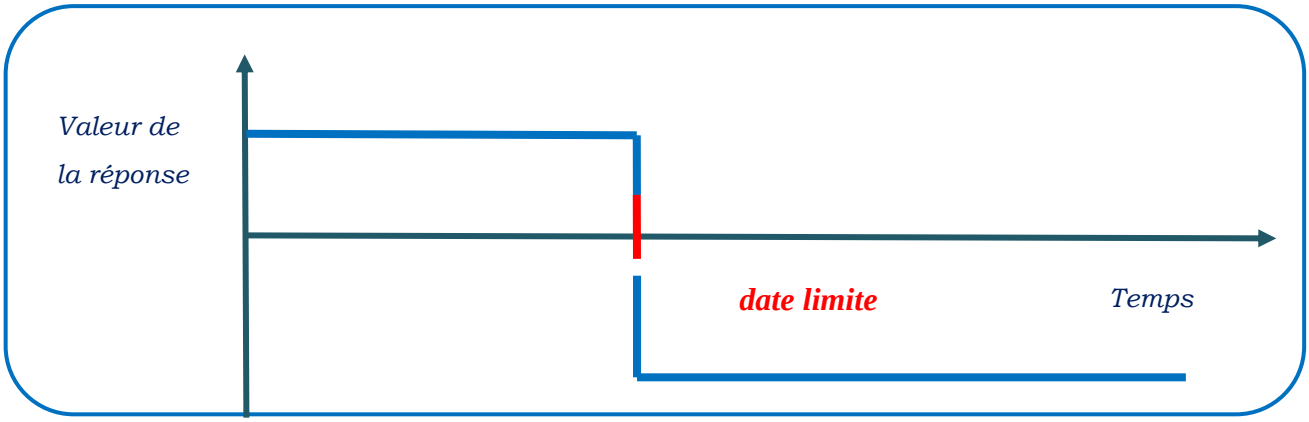


Figure 1.1 : date limite (deadline) en temps réel dur[IUL 21]

3-2-2- Temps-réel ferme (firm real-time) :

La réponse du système dans les délais est essentielle. Le résultat ne sert plus à rien une fois le deadline passé. (définition alternative : la pénalité de non-réponse est dans le même ordre de magnitude que la valeur de la réponse).

Exemples : transactions en bourse… (c'est subjectif : le temps réel ferme de l'un peut être le temps réel dur de l'autre).

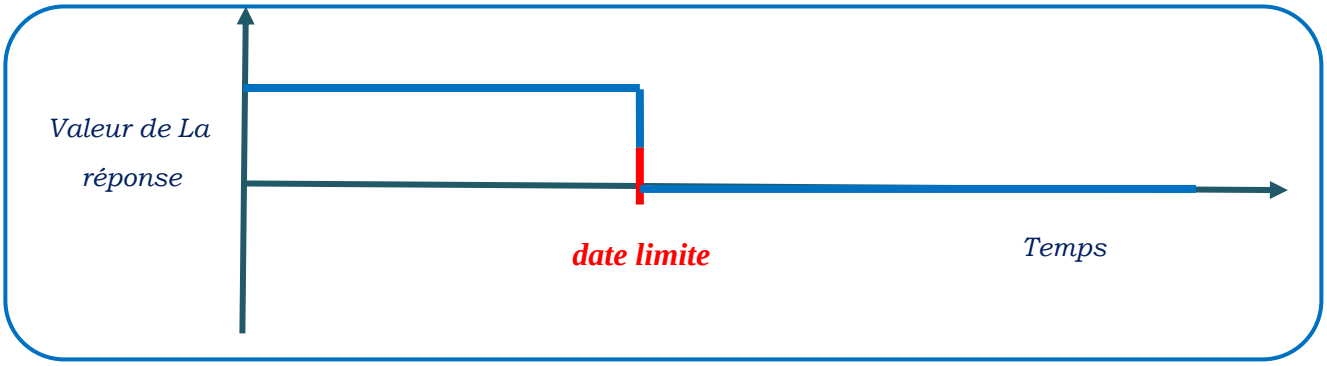


Figure 1.2 :date limite (deadline) en temps réel ferme[IUL 21]

3-2-3- Temps réel souple/mou (soft real-time) :

La réponse du système après les délais réduit progressivement son intérêt. Les pénalités ne sont pas catastrophiques.

Exemples : VoD, logiciel embarqué de téléphone, iPod, etc.

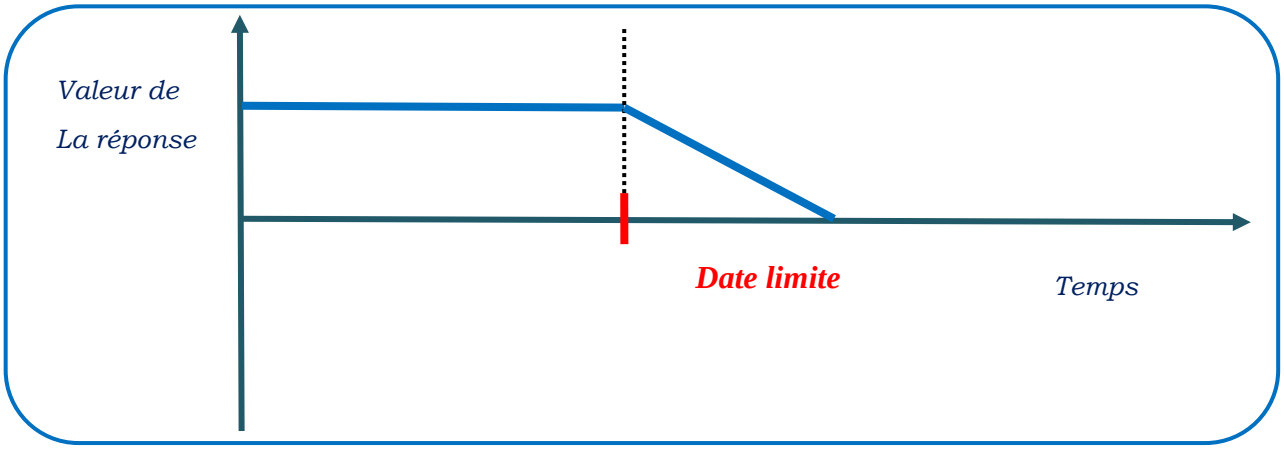


Figure 1.3 :date limite (deadline) en temps réel mou[IUL 21]

3-3-Exemples de grandeur des contraintes temporelle :

-La milliseconde pour les systèmes de radar

-La seconde pour les systèmes de visualisation humaine

-Quelques heures pour le contrôle de production impliquant des réactions chimiques

-24 heures pour les prévisions météo.

-Plusieurs mois ou années pour les systèmes de navigation de sonde spatiale.

3-4-Généralités sur l'ordonnancement temps réel embarqué :[Mar 12]

L'ordonnancement est le terme informatique désignant le mécanisme permettant de réaliser la sélection, parmi plusieurs composants de l'algorithme (tâches), de celui qui va obtenir l'utilisation d'un composant de l'architecture pour s'exécuter, de manière à optimiser un ou plusieurs critères. L'ordonnancement temps réel a une particularité qui nécessite de respecter des contraintes de temps réel. Le processus qui réalise une telle sélection, est donc qui définit comme ordre d'exécution entre les composants de l'algorithme, est appelé algorithme d'ordonnancement.

(Algorithme de distribution et d'ordonnancement temps réel) "C'est un algorithme d'ordonnancement temps réel qui doit réaliser, en plus d'une opération désélection d'un composant

logiciel, une opération de sélection du composant matériel qui peut exécuter le composant logiciel sélectionné. Ce type d'algorithme est nécessaire dès que l'architecture est distribuée".

3-5-Nature des tâches :

On distingue deux genres de tâches selon le type des intervalles de temps, intervalles réguliers donc tâches périodiques, intervalles aléatoires donc tâches non périodiques.

3-5-1-Tâches périodiques :

3-5-1-1-Modèle classique de tâches préemptives périodiques :

Un modèle canonique de tâches temps réel préemptives périodiques a été proposé par Liu &Layland. Ce modèle regroupe les principaux paramètres temporels qui permettent de caractériser une tâche temps réel et son ordonnancement.

Ces paramètres sont scindés en deux groupes : paramètres statiques et paramètres dynamiques.

Afin d'alléger la lecture, les tâches préemptives périodiques (non strictes) sont notées tâches PP dans la suite du manuscrit.

Les paramètres de base d'une tâche $\tau i(r^0_i, C_i, D_i, T_i)$ sont :

– r^0_i (first release ou offset) : première date de réveil ou d'activation, qui représente le moment de déclenchement de la première requête d'exécution,

– C_i (computing time) : durée d'exécution maximale d'une tâche, souvent considéré comme le pire temps d'exécution WCET (Worst Case Execution Time), qui est une borne supérieure du temps d'exécution d'une tâche sur un processeur donné.

– D_i (relative deadline) : échéance relative, date au plus tard ou délai critique, ce paramètre représente le délai, relativement à la date d'activation, avant laquelle l'exécution de la tâche doit être terminée.

– T_i (period) : période d'exécution d'une tâche périodique ou écart minimum entre deux activations successives d'une tâche apériodique.

Notez que pour la k^{eme} instance τ^k_i, la date d'activation (release time, request time, ready time) est donnée par $r^k_i = r^0_i + k \cdot T_i$.

Lorsque la période d'une tâche est égale à son échéance, on dit que la tâche est à échéance sur requête.

D'autres paramètres sont dérivés des paramètres de base :

– $U_i = C_i/T_i$(utilization factor) : facteur d'utilisation d'une tâche ; on a $U_i \leq 1$.

– $CH_i = C_i/D_i$(density) : densité d'une tâche ; on a $CH_i \leq 1$.

Les paramètres dynamiques servent à suivre le comportement de l'exécution d'une tâche :

– s^k_i(start time) : date de début d'exécution de la k^{me} instance τ^k_i.

– e^k_i(end time) : date de la fin d'exécution de la k^{me} instance τ^k_i ,

– d^k_i(absolute deadline) : échéance absolue de la k^{me} instance τ^k_i donnée par $d^k_i = r^k_i + D_i = r^0_i + k \cdot T_i + D_i$.

– R^k_i(response time) : temps de réponse de la k^{me} instance τ^k_i donné par e^k_i

– r^k_i . On a $C_i \leq R^k_i \leq D_i$,

– L_i (laxity) : laxité nominale d'une tâche qui représente le retard maximum pour son début d'exécution ski

Lorsque cette tâche est exécutée seule.

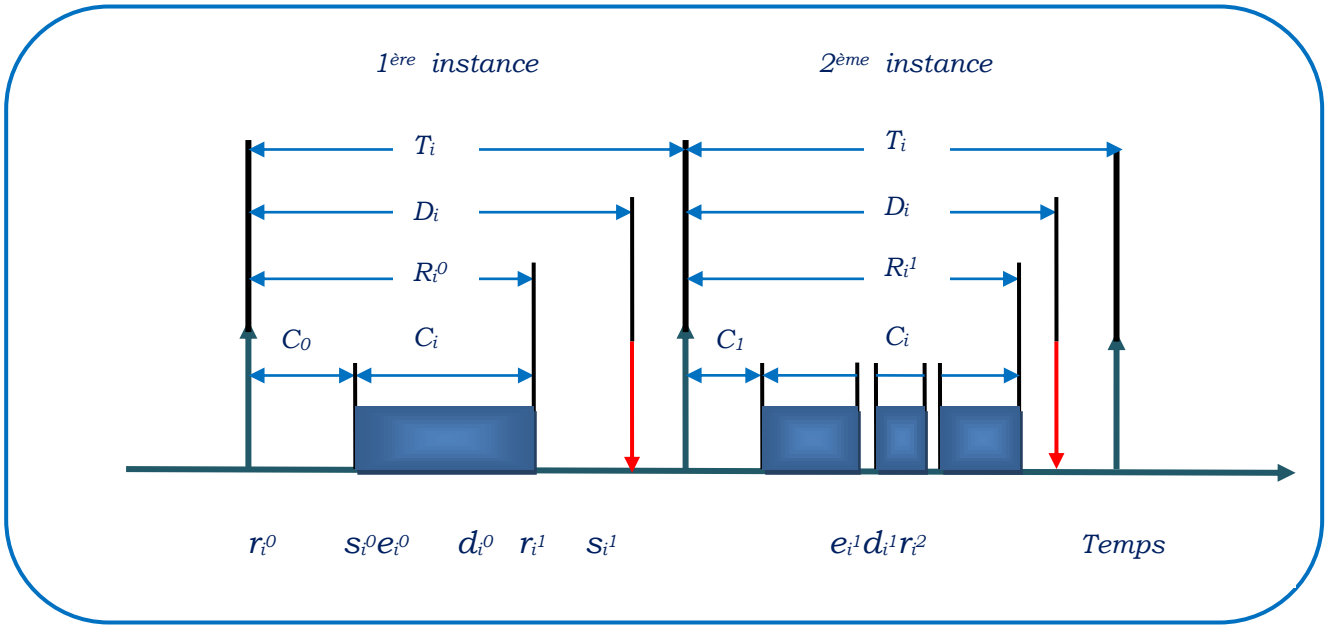


Figure 1.4 : modèle classique d'une tâche préemptive périodique. [Mar 12]

3-5-1-2-Modèle de tâches non préemptives périodiques strictes :

Dans les applications de contrôle/commande temps réel critiques, les tâches, réalisant les entrées/sorties ne doivent pas avoir de gigue. Donc pour éviter la gigue sur les entrées issues des capteurs les tâches les gérant doivent être périodiques strictes, et pour éviter la gigue sur les sorties fournies aux actionneurs les tâches les gérant doivent être non préemptives.

Le modèle de tâches à non préemptives périodiques strictes est le modèle naturel utilisé pour faire de l'ordonnancement sans préemption qui a existé bien avant celui préemptif dit de Liu &Layland décrit précédemment. Dans ce dernier la différence entre la date d'activation et la date de

début d'exécution peut varier alors qu'elle est constante dans le cas de tâches non préemptives périodiques strictes.

Ce qui revient à dire qu'il n'y a pas de gigue . De plus il a le gros avantage d'être déterministe car il ne dépend pas de l'approximation du coût de l'OS définie par Liu &Layland. nécessaire pour gérer la préemption. Cet avantage est utile dans le cas des applications temps réel critiques citées plus haut.

Afin d'alléger la lecture, les tâches non préemptives périodiques strictes sont notées tâches NPPS dans la suite du manuscrit. Ce modèle a les caractéristiques suivantes :

– la différence entre la date d'activation et la date de début d'exécution d'une instance $\boldsymbol{\tau}^{k}_{i}$ est égale à une constante : $s^{k}_{i} - r^{k}_{i} = d$.

– le temps de réponse est constant : $R_i = C_i + d$.

– la différence entre deux dates de début d'exécution successives est égale à la période :

$\boldsymbol{s_i^{k+1} - s^k_i = T_i.}$

– l'échéance est égale à la période : $T_i = D_i$.

La figure 05 représente un modèle de tâches NPPS dans le cas où d = 0. Dans la suite du manuscrit et sans perte de généralité nous ferons cette hypothèse.

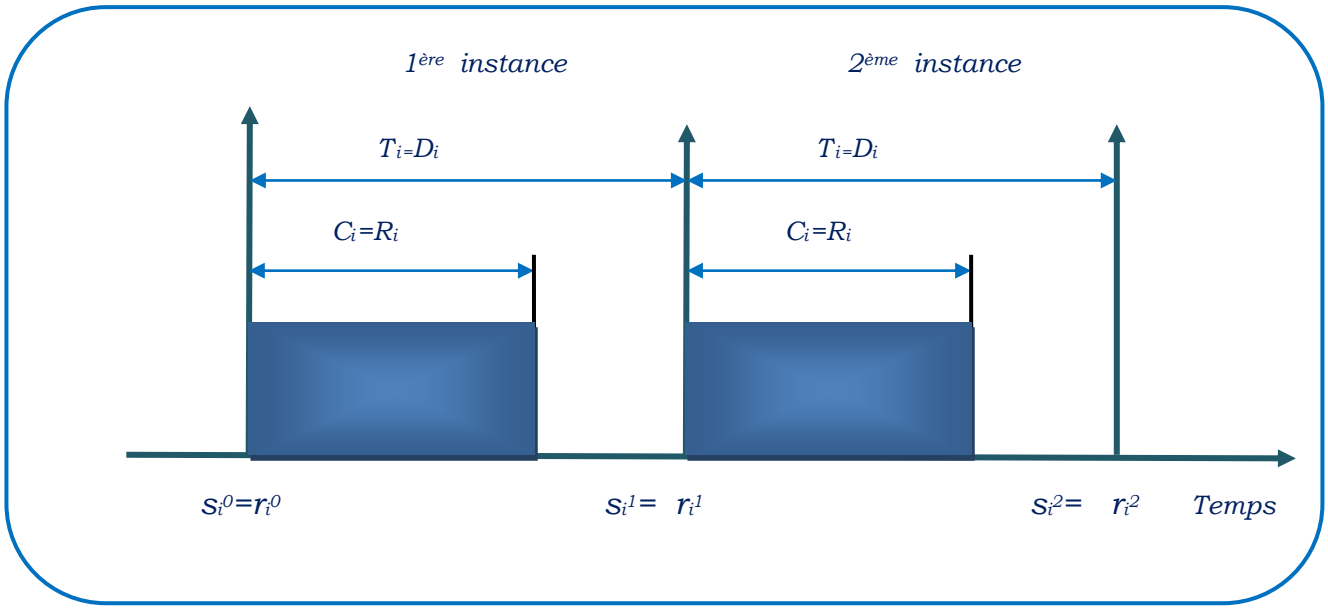


Figure 1.5 modèle classique d'une tâche non préemptive périodique stricte [Mar 12]

3-5-2-Tâches non périodiques :[Oma 09]

Il existe deux types de tâches non périodiques :

3-5-2-1-Apériodique :

Les dates d'activations sont aléatoires et ne peuvent être anticipées, Son exécution est le produit d'événements internes ou externes qui peuvent se déclencher à tout instant.

Dans différents procédés, pour ordonnancer l'exécution des tâches apériodiques d'un système temps réel, sont exposés.

3-5-2-2- Sporadique :

C'est un cas particulier des tâches apériodiques où une durée de temps minimale sépare deux activations successives, Pour les prendre en compte, ces tâches sont souvent considérées comme tâches périodiques pour appliquer les résultats existant des tâches périodiques.

3-5-3-Tâches concrètes/non-concrètes :

Si un scénario d'activation particulier est imposé aux tâches on dit qu'elles sont concrètes. Par ailleurs les tâches sont non-concrètes si les dates de première activation ne sont pas connues à priori.

3-5-4-Approches synchrones/asynchrones :

Dans le cas synchrone un scénario d'activation est imposé et reproduit à l'identique pendant toute la vie du système, tandis que dans le cas asynchrone une hypothèse sur les activations des tâches est prise en compte.

3-5-5-Dépendance et précédence :

En général les automaticiens et traiteurs de signal spécifient fonctionnellement les systèmes temps réel embarqué avec des blocs diagrammes décrivant des fonctions qui sont soit indépendantes, soit dépendantes dans le sens où une fonction produit des données pour une autre fonction qui les consomme.

Une dépendance impose une précédence entre la fonction qui produit et celle qui consomme, les fonctions deviennent des tâches dès qu'on les a caractérisées temporellement..

Une contrainte de précédence entre deux tâches impose un ordre entre ces tâches Il faut noter que cette contrainte de précédence est la plupart du temps (donc pas toujours) due au fait que les deux fonctions sont dépendantes. Le terme .précédence. désigne un ordre entre deux tâches sans qu'il y ait transfert de données alors que .indépendance . Signifie qu'un ordre n'est pas imposé entre deux tâches. Par ailleurs deux tâches sont dites .dépendantes. si l'une a besoin du résultat

d'exécution de l'autre pour pouvoir être exécutée à son tour, ce qui impose, bien évidemment, une

Précédence entre ces deux tâches. On appelle la tâche qui produit les données .la tâche productrice (Prédécesseur) et la tâche qui consomme les données .la tâche consommatrice. (Successeur). La dépendance entre les tâches peut être modélisée par un graphe orienté où les nœuds représentent les tâches et les arcs les relations de dépendance entre les tâches. On distingue deux types de dépendances :

1. sans perte de données : les données produites par l'exécution des instances de la tâche productrice

sont toutes consommées par les instances de la tâche consommatrice,

2. avec perte de données : la tâche consommatrice peut perdre des données qui sont écrasées par d'autres données produites par l'exécution d'autres instances de la tâche productrice.

3-5-6-Latence :

Imposer une contrainte de latence entre les deux tâches t_a et t_b équivalente à L (t_a,t_b) signifie imposer que la durée de temps qui s'écoule entre la date de début d'exécution de la tâche *ta* et la date de _n d'exécution de la tâche t_a ne doit pas dépasser L (t_a,t_b). L'intérêt de borner le temps de calcul entre ces deux tâches est de garantir que le temps de réponse - le temps total que met une tâche pour s'exécuter entièrement - de la deuxième tâche ne dépassera jamais une certaine valeur critique à partir de laquelle les performances du système ne seraient plus satisfaisantes ou à partir de laquelle le système serait instable.

3-5-7-Makespan :

Dans les systèmes temps réels chaque sortie peut être fonction de plusieurs entrées. Le temps total d'exécution, que l'on appellera dans la suite « makespan». Car c'est le terme utilisé dans la littérature, reflète le temps qui s'écoule entre la date de début d'exécution de la première tâche exécutée et la date de fin de la dernière tâche exécutée. L'objectif est de développer des algorithmes qui, en plus du respect des autres contraintes temporelles, minimisent le makespan.

Prenons l'exemple d'une application de chronomètre électronique, l'environnement est constitué par des boutons de mise en marche, d'arrêt, de remise à zéro et d'un afficheur.

Le système informatique qui contrôle cet environnement est un système temps réel. En effet, l'action .arrêter le comptage du chronomètre. Déclenchée par l'événement .stop. (pression sur un bouton) doit être accomplie en un temps inférieur à la précision recherchée sur la mesure sous peine d'aboutir à un résultat erroné. Ici un temps de réaction trop long conduit à un mauvais

fonctionnement de l'application. Un autre exemple, le cas du véhicule détecteur d'obstacles. La détection tardive d'un obstacle peut conduire à une collision susceptible d'engendrer des dégâts.

3-6-Les algorithmes d'ordonnancement :

3-6-1-L'algorithmes d'ordonnancement sur une architecture monoprocesseur :

Cette sous-section [ROH04] expose brièvement les principes des différents algorithmes d'ordonnancement, en fournissant aussi des exemples de leur fonctionnement dans le cas monoprocesseur, et pour des tâches indépendantes et préemptibles.

Dans la littérature on distingue quatre catégories principales d'algorithmes d'ordonnancement:

• statiques pilotés par table.

• statiques préemptifs basés sur des priorités.

• dynamiques basés sur une planification de l'exécution.

• dynamiques basés sur la notion du meilleur effort (best effort).

Les algorithmes dynamiques sont les plus flexibles car ils permettent la prise en compte d'événements « non prévus ». Parmi ceux-ci les principaux sont :

• l'ordonnancement par priorités fixes : ƒ

-premier arrivé, premier servi (First-Come, First-Served – FCFS) où toutes les tâches ont la même priorité.ƒ

-le tour de rôle ou tourniquet (Round Robin – RR).ƒ

-monotone par fréquences (Rate-Monotonic – RM). ƒ

-monotone par échéances (« Invers Deadline » - ID, ou « Deadline Monotonic » -DM).

• l'ordonnancement par priorités dynamiques : ƒ

-Earliest Deadline First – EDF. ƒ

-Least Slack Scheduling – LLS, ou Least Laxity First – LLF, ou Shortest Slack Time – SST.

Dans un cadre plus général, les mêmes principes d'ordonnancement s'appliquent aux tâches sur une machine monoprocesseur comme sur une plate-forme multiprocesseur. Dans ce dernier cas quelques hypothèses spécifiques à l'existence de plusieurs processeurs sont prises en compte :

• l'exécution d'une tâche n'est pas associée à un processeur déterminé.

• à tout instant une tâche ne peut s'exécuter que sur un seul processeur.

Pour certaines situations spécifiques aux applications, la première hypothèse peut ne pas être valable ; cette thèse ne traite pas cet aspect.

3-6-1-1-Rate-monotonic (RM) :[ROH04]

Principe de fonctionnement : l'algorithme « *rate-monotonic* » est un algorithme statique qui attribue la plus haute priorité à la tâche ayant la fréquence la plus élevée (la plus petite période).

La Figure 06 indique l'ordonnancement « *rate-monotonic* » de trois tâches périodiques à échéances sur requêtes : T1(R1=0, C1=3, P1=20), T2(R2=0, C2=2, P2=5) et T3(R3=0, C3=2, P3=10).

- La tâche la plus prioritaire est la tâche T2 (P2=5),
- la tâche la moins prioritaire est la tâche T1(P1=20).
- La séquence est décrite sur l'intervalle [0,20] = [0,ppcm(Pi)],
- les trois tâches respectent leurs contraintes temporelles.

A observer la répétitivité de la séquence durant tout intervalle ayant la forme ***[k∗ppcm(Pi),(k+1)∗ppcm(Pi)], k≥0, k∈N, où ppcm(Pi)]*** représente le plus petit multiple commun des périodes Pi.

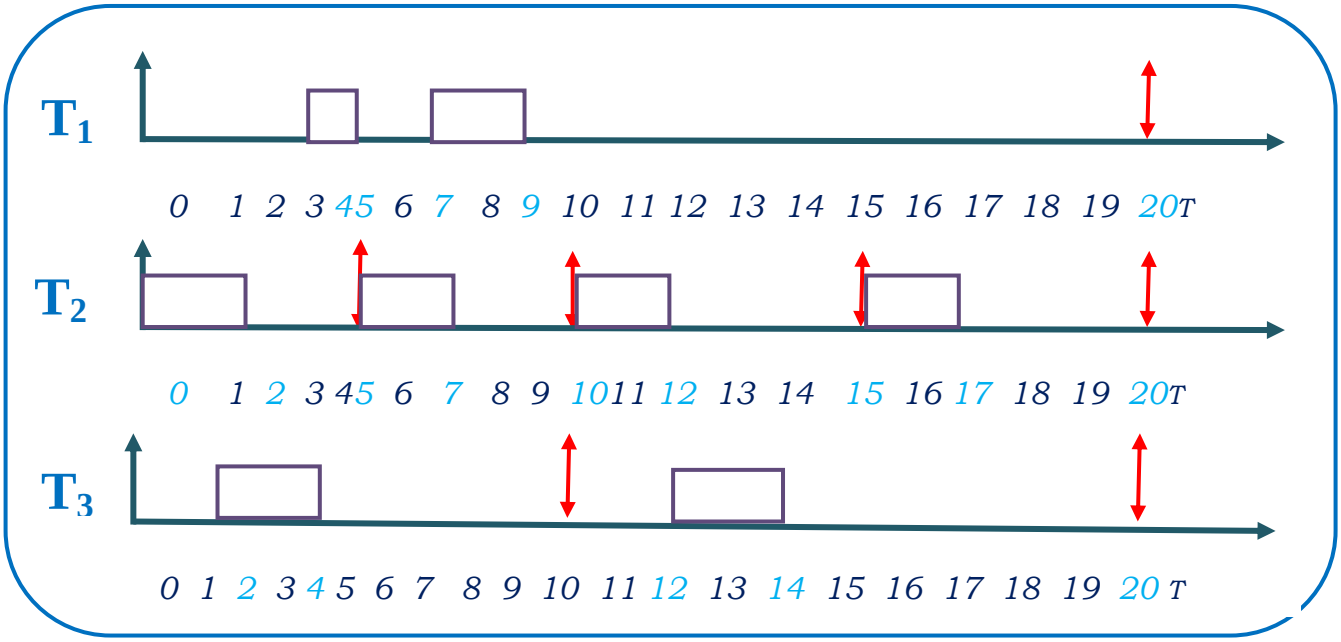


Figure 1.6 : rate-monotonic[ROH04]

Notons que l'utilisation de la périodicité comme critère d'ordonnancement limite l'applicabilité de cet algorithme aux seules tâches périodiques à échéance sur requête. L'utilisation de cet algorithme pour d'autres types de tâches ne donne aucune garantie pour le respect des échéances.

3-6-1-2-Inverse deadline (ID) ou "deadline monotonic" (DM) :

Principe de fonctionnement [ROH04]: l'algorithme *« inverse deadline »* est un algorithme statique où les priorités des tâches sont exprimées en fonction de leurs délais, la tâche la plus prioritaire étant celle qui a le plus petit délai.

(La Figure 07) indique l'ordonnancement « inverse deadline » des trois tâches périodiques suivantes:

.T_1(R_1=0, C_1=3, D_1=7, P_1=20).

.T_2(R_2=0, C_2=2, D_2=4, P_2=5) .

.T_3(R_3=0, C_3=2, D_3=9, P_3=10).

La tâche la plus prioritaire est la tâche T_2 (D_2=4), et la tâche la moins prioritaire est la tâche T_3(D_3=9). La séquence est décrite sur l'intervalle [0,20] = **[0,ppcm(P_i)]**, et les trois tâches respectent leurs contraintes temporelles.

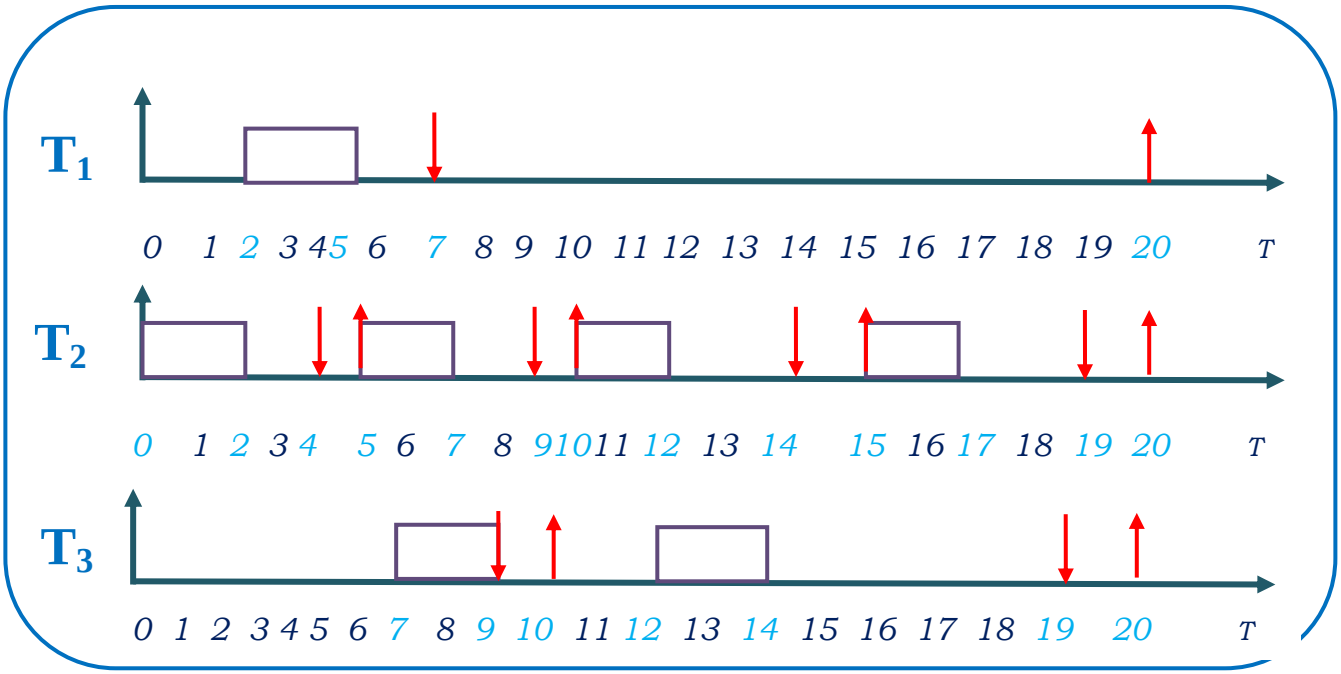


Figure 1.7 : Inverse Deadline[ROH04]

Notons que, par rapport à l'algorithme Rate-monotonic, étant basé sur la notion d'échéance, cet algorithme s'applique aussi bien pour d'autres modèles de tâches que ceux des tâches à échéances sur requêtes.

3-6-1-3-Least laxity first (LLF) :

La laxité d'une tâche au moment *t* représente son retard maximum par rapport à son échéance pour (re)prendre son exécution, quand la tâche s'exécute seule [ROH04] :

$$L(t)=D(t)-C(t)$$

Principe de fonctionnement : L'algorithme *"least laxity first"* attribue, à l'instant *t*, la plus haute priorité à la tâche ayant la plus petite laxité. Le calcul des laxités des tâches peut se faire :

-aux dates de réveil des tâches – cas où la séquence produite est équivalente à celle produite par EDF.

-à chaque instant – cas qui entraîne plus de changements de contexte.

La Figure (08) indique un ordonnancement donné par LLF des trois tâches périodiques :

$T_1(R_1=0, C_1=3, D_1=7, P_1=20)$,

$T_2(R_2=0, C_2=2, D_2=4, P_2=5)$ et

$T_3(R_3=0, C_3=1, D_3=8, P_3=10)$.

A l'instant t=0 la tâche la plus prioritaire est la tâche T_2 (L_2=2), et la tâche la moins prioritaire est la tâche T_3(L_3=7).

Le calcul des laxités est fait aux dates de réveil des tâches et, si deux tâches ont la même laxité au même instant, celle qui s'exécute en première est celle d'indice plus petit (comme le montre le cas de l'instant t=5 où les tâches T_2 et T_3 ont la même laxité, 2).

La séquence est décrite sur l'intervalle [0,20] = **[0,ppcm(P_i)]**, et les trois tâches respectent leurs contraintes temporelles.

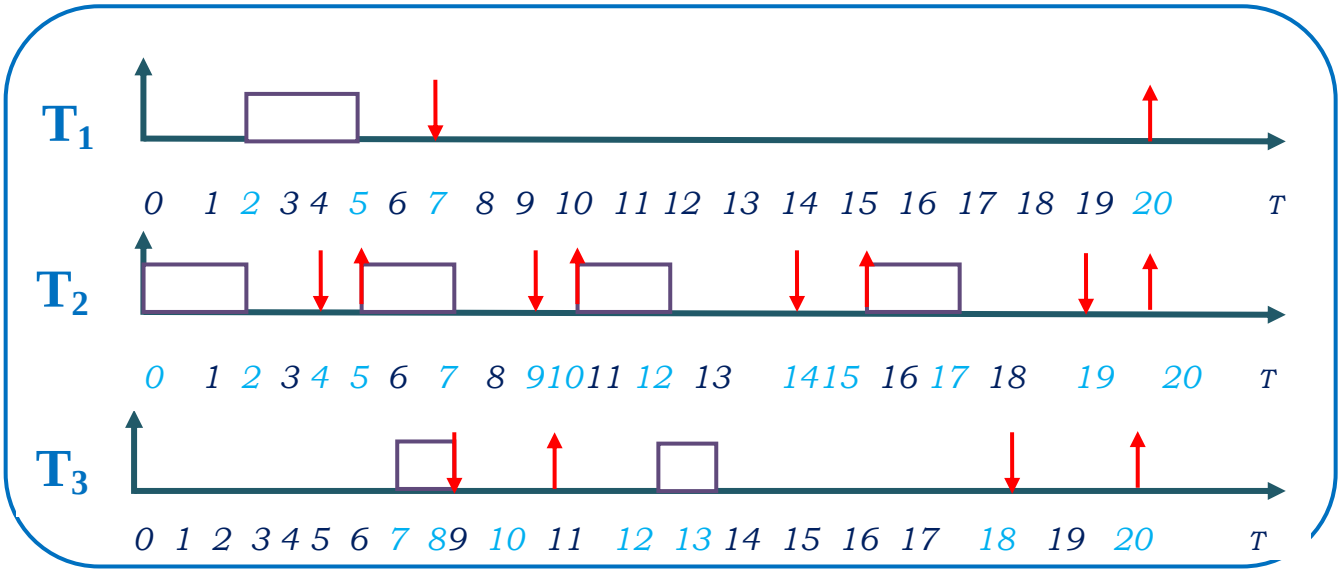


Figure 1.8: least laxity first [ROH04]

3-6-1-4-Earliest deadline first (EDF) :

Principe de fonctionnement [ROH04] **:** L'algorithme "Earliest Deadline First" attribue, à l'instant t, la plus haute priorité à la tâche ayant la plus proche échéance.

La Figure (07) indique un ordonnancement donné par EDF des trois tâches périodiques :

$T_1(R_1=0, C_1=3, D_1=7, P_1=20)$.

$T_2(R_2=0, C_2=2, D_2=4, P_2=5)$.

$T_3(R_3=0, C_3=1, D_3=8, P_3=10)$.

A l'instant t=0 la tâche la plus prioritaire est la tâche T_2 (D_2=4), et la tâche la moins prioritaire est la tâche T_3(D_3=8). Les priorités des tâches, et donc l'ordre de leur exécution, évoluent les unes par rapport aux autres en fonction de leur urgence. La séquence est décrite sur l'intervalle [0,20] = **[0,ppcm(P_i)]**, et les trois tâches respectent leurs contraintes temporelles.

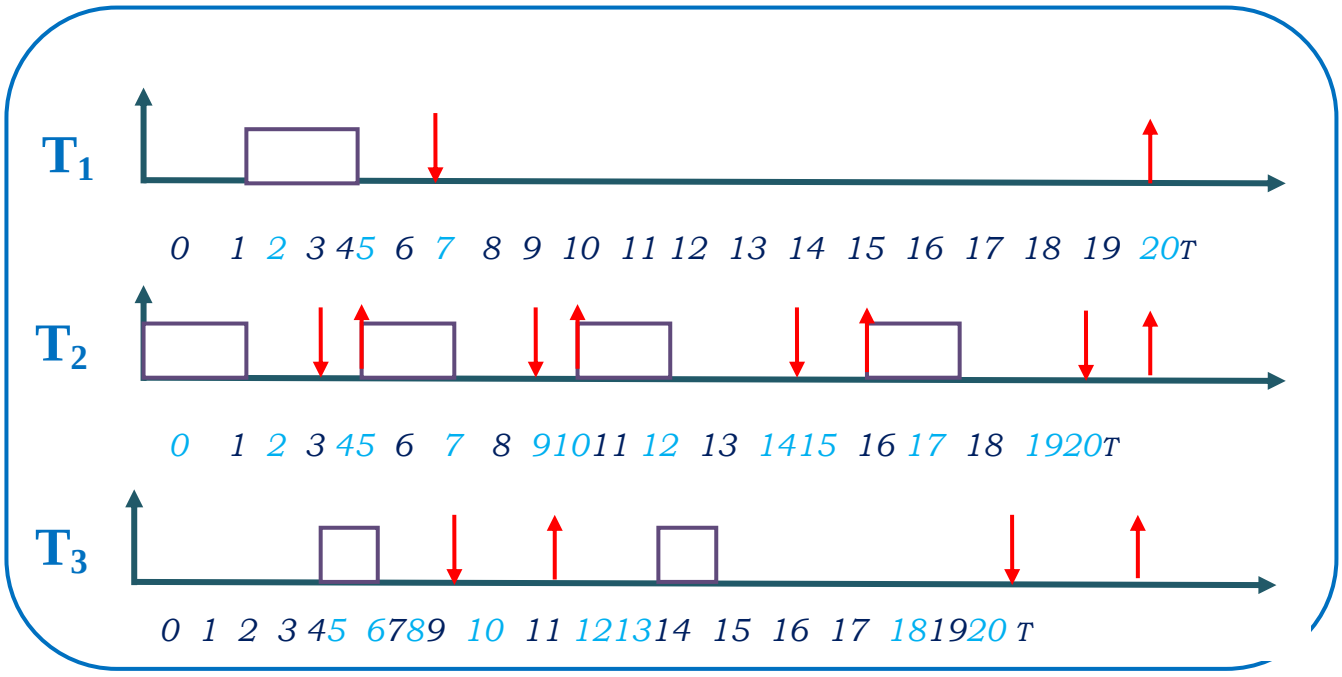


Figure 1.9 : earliest deadline first[ROH04]

EDF est la notation utilisée pour l'algorithme EDF où, parmi les tâches ayant la même échéance, celle qui arrive en première sera élue.

3-7-Conclusion :

Les systèmes embarqués se réfèrent à des systèmes informatiques qui sont créés pour exécuter un nombre défini de tâches ou fonctions. Ils sont noyés dans le sens où le système informatique est incorporé dans un appareil à côté matériel nécessaire.

Conception et efficacité le noyau central de traitement dans les systèmes embarqués est généralement moins complexe, ce qui rend plus facile à entretenir. La fonction limitée nécessaire des systèmes embarqués leur permet d'être conçus pour exécuter plus efficacement leurs fonctions.

Chapitre 2-

LE CLOUD COMPUTING

1-Introduction

Le Cloud Computing (nuage) est un terme général employé pour désigner la livraison de ressources et de services à la demande par internet**.** Il désigne le stockage et l'accès aux données par l'intermédiaire d'internet plutôt que via le disque dur d'un ordinateur. Il s'oppose ainsi à la notion de stockage local, consistant à entreposer des données ou à lancer des programmes depuis le disque dur. La notion de Cloud ne doit pas non plus être confondue avec celle du Network Attached Storage (NAS), utilisée par beaucoup d'entreprises via un serveur en résidence. Ces réseaux locaux n'entrent pas dans la définition du Cloud. cependant, certains NAS permettent d'accéder aux données à distance depuis Internet.

De manière générale, on parle de Cloud Computing lorsqu'il est possible d'accéder à des données ou à des programmes depuis internet**,** ou tout du moins lorsque ces données sont synchronisées avec d'autres informations sur internet. Il suffit donc pour y accéder de bénéficier d'une connexion internet.

2-Description du Cloud computing (informatique en nuage) [GRE 09]

1-Le « Cloud computing » est un néologisme utilisé pour décrire l'association d'Internet (cloud= le nuage) et l'utilisation de l'informatique (computing), C'est une manière d'utiliser l'informatique dans laquelle tout est dynamiquement couplé et évolutif et dans laquelle les ressources sont fournies sous la forme de services au travers d'internet. Les utilisateurs n'ont ainsi besoin d'aucune connaissance ni expérience en rapport avec la technologie derrière les services proposés.

Le Cloud Computing est un nuage de services et de données. Plus précisément, c'est un paradigme, et à ce titre, il est difficile de lui donner une définition exacte et de dire avec certitude s'il s'agit ou non de Cloud.

Il faut donc être vigilant, car de nombreux fournisseurs de services utilisent le mot« Cloud » à des fins marketings. sur Internet, il n'y a pas de définition exacte du Cloud Computing et donc pas de certification pour dire si nous avons à faire à un« vrai Cloud ».

Le cloud Computing correspond au développement et à l'utilisation des applications accessibles uniquement via Internet. les utilisateurs dépendent ainsi uniquement d'Internet pour utiliser leurs logiciels, ils ont la possibilité d'accéder à des services sans installer quoique ce soit d'autre qu'un simple navigateur Internet. Aujourd'hui, le cloud computing est exploité par la quasi-totalité des grandes entreprises car il fournit une analyse sophistiquée des données de la manière la plus rapide possible.

2-Le Cloud Computing :[AMA 21] est la mise à disposition de ressources informatiques à la demande via Internet, avec une tarification en fonction de votre utilisation. Au lieu d'acheter, de posséder et de gérer des serveurs et des centres de données physiques, vous pouvez accéder à votre guise aux services technologiques, tels que la puissance de calcul, le stockage et les bases de données, d'un fournisseur cloud tel qu'Amazon Web Services (AWS).

3-Historique : [HOL 21]

Pour localiser les débuts du Cloud, nous devons retourner à la préhistoire d'Internet. Bien que ce concept soit né en 1990, certains pionniers de l'informatique avaient anticipé l'avancée des réseaux informatiques dans les années 1960. *J.C.R. Liclkider, qui a participé au développement d'ARPANET, – réseau informatique créé sur demande du département de la Défense des États-Unis– et John McCarthy, père du terme Intelligence Artificielle,* ont été deux des principaux visionnaires de ce qu'allait devenir le Cloud.

Licklider a dessiné les premières idées d'un réseau informatique mondial en 1962, dans des discussions sur le concept du « Réseau informatique intergalactique ». Ces idées contenaient une grande partie du substrat de ce que nous connaissons maintenant comme Internet. L'Américain a, dans divers documents, décrit certaines applications sur les réseaux et a prédit l'utilisation de réseaux pour soutenir les communautés d'intérêts communs indépendamment de l'emplacement de leurs utilisateurs. Pour sa part, McCarthy a déclaré : « L'informatique sera un jour organisée comme un service public, comme l'électricité ou l'eau. »

Il s'agissait de l'idée de base, mais l'application du Cloud tel que nous le connaissons aujourd'hui a commencé dans les années 80, lorsque certaines tâches ont commencé à prendre corps dans un réseau informatique, plutôt que sur un seul ordinateur. De cette façon, la tâche est répartie entre plusieurs machines, exigeant moins du système pour fournir le service aux utilisateurs.

Ainsi, le terme a commencé à être utilisé pour d'abord désigner les réseaux téléphoniques, comme un moyen de se référer à quelque chose dont l'utilisateur n'a pas besoin de s'inquiéter, puis il est passé dans les infrastructures informatiques.

Ici, le mot « Cloud » a été utilisé pour la première fois au sein de l'environnement académique en 1997 par le professeur Ramnath *Chellappa,* qui l'a défini comme « un nouveau paradigme informatique ». Puis, en 1999, la société Salesforce.com a été la première à introduire le terme « logiciel en tant que service », en fournissant des applications pour les entreprises par l'intermédiaire d'un site Web.

4-La virtualisation :[WAR 11]

La virtualisation a été la première pierre vers l'ère du Cloud Computing. En effet, cette notion permet une gestion optimisée des ressources matérielles dans le but de pouvoir y exécuter plusieurs systèmes « virtuels » sur une seule ressource physique et fournir une couche supplémentaire d'abstraction du matériel. Les premiers travaux peuvent être attribués à IBM, qui dans les années 60, travaillait déjà sur les mécanismes de virtualisation en développant dans les centres de recherche de Cambridge et de Grenoble, CMS (Conversation Monitor System), le tout premier hyperviseur.

C'est donc depuis presque 50 ans que l'idée d'une informatique à la demande est présente dans les esprits même si les technologies n'étaient jusqu'alors pas au rendez-vous pour pouvoir concrétiser cette idée. Avec les différents progrès technologiques réalisés durant ces 50 dernières années, tant sur le plan matériel, logiciel et conceptuel, aux avancées des mécanismes de sécurité, à l'élaboration de réseaux complexes mais standardisés comme Internet, et à l'expérience dans l'édition et la gestion de logiciels, services, infrastructures et stockage de données, nous sommes maintenant prêts à entrer dans l'ère du Cloud Computing, telle que rêvait par John McCarthy en 1961.

5-Composants du nuage: [ORA 21]

a. Applications virtualisées : Les applications virtualisées rendent compatible les applications de l'utilisateur avec les hardwares, les systèmes d'exploitations, le réseau et le stockage pour permettra la flexibilité du déploiement.

b. Infrastructure virtualisée: L'infrastructure virtualisée fournit l'abstraction nécessaire pour s'assurer qu'une application ou un service ne soit pas directement attachée à l'infrastructure matérielle (serveurs, stockage ou réseaux). Ceci permet au service de se déplacer dynamiquement à travers les ressources virtualisées d'infrastructure.

c. Gestion de sécurité et d'identité: Le système de gestion de sécurité fournie les commandes nécessaire pour assurer les informations sensibles (les protéger) être pendre à l'exigence de conformité.

d. Développement: Les infrastructures de développement facilitent non seulement l'orchestration de service mais permettent également aux processus d'être développés. C'est les outils de développement comme le compilateur, SDK (*Software Developpment Kit*) et l'environnement de développement.

e. Gestion d'entreprise : La couche de gestion d'entreprise manipule le cycle de vie des ressources virtualisées et fournit les éléments additionnels d'infrastructure commune pour la gestion

de taux de disponibilité, utilisation dosée, gestion de politique, gestion de permis, et recouvrement des pertes.

6-L'informatique en tant que service :

On distingue trois sous-ensembles de services (Figure 10) au sein de l'informatique dans le nuage : le logiciel en tant que service (SaaS), la plateforme en tant que service(PaaS) et l'infrastructure en tant que service (IaaS). Chacun de ces types de service correspond à un niveau d'abstraction logiciel précis par rapport aux ressources informatiques matérielles accessibles via Internet, et donc hébergées au sein du nuage du point de vue de l'utilisateur

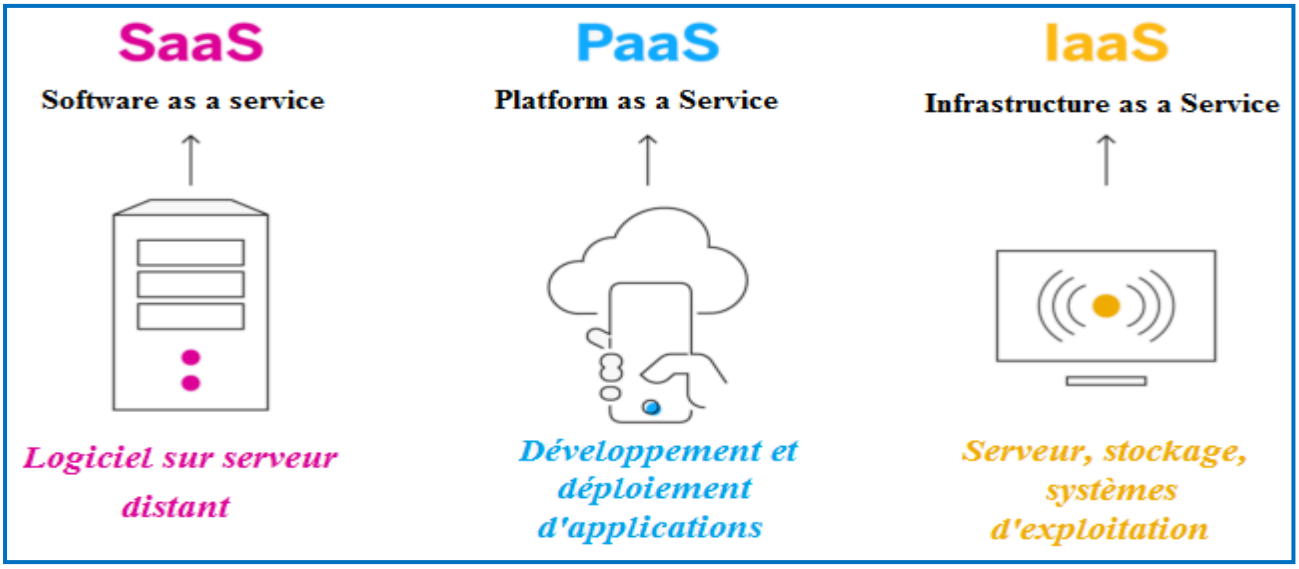


Figure 2.1 : les (SaaS-PaaS-IaaS)

6-1-IaaS :

L'infrastructure as a Service [WAR 15] est un modèle de Cloud Computing qui permet aux entreprises de disposer via un abonnement mensuel d'une infrastructure informatique : serveurs, stockage, sauvegarde, réseau… qui se trouve dans le Datacenter de leur fournisseur.

- **Les avantages :**

1. L'obligation d'acquérir votre propre matériel informatique, de l'installer et de le maintenir.

2. Vous gagnez en flexibilité car vous louez mensuellement des ressources Cloud.

3. Les entreprises utilisant *l'IaaS* bénéficient d'une agilité accrue. Elles peuvent ainsi augmenter ou diminuer leurs consommations en fonction de leurs besoins.

6-2-PaaS :

Plate-forme as a Service permet aux entreprises de disposer d'un environnement informatique disponible rapidement tout en leur laissant la totale maitrise des applications qu'elles installent,

configurent et utilisent. Le *PaaS* est tout à fait adapté pour héberger des applications qui ne sont pas adaptées au modèle *SaaS*.

- **Les avantages :**

1. Le développement et le fonctionnement des applications peuvent se faire sans aucune préoccupation pour les mises à jour de la plateforme et des logiciels qui fonctionnent dessus, qui sont entièrement pris en charge par le prestataire.

2. L'accès à la plateforme se fait directement par navigateur Web sans installation d'un plug-in. L'exécution de l'application s'effectue sur la même plateforme, fournie à distance et par le biais d'une interface Web par votre hébergeur.

6-3-SaaS :

Software as a Service est le modèle Cloud le plus simple pour utiliser un ou plusieurs logiciels. Les entreprises accèdent à leurs applications depuis tout poste connecté, depuis un simple navigateur web. Les clients paient un accès à leurs logiciels sous forme d'abonnement.

- **Les avantages :**

1. Pas d'installation de logiciels, vous accédez à l'application depuis votre navigateur internet.

2. Vous n'achetez plus une licence d'utilisation mais un accès à votre application.

3. Vos logiciels bénéficient de nombreuses mises à jour de l'éditeur, sans maintenance particulière de votre part.

7-Exemples courants *deSaaS, PaaS,* et *IaaS* :

Type de plate-forme	Exemples
SaaS	*Google Workspace, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting*
PaaS	*AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift*
IaaS	*DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)*

Table 1.1 : type des plates-formes

8-Modèles de déploiement :

Un nuage correspond à une infrastructure distante [AND 11] dont on ne connaît pas les détails architecturaux, et qui est connue pour les services informatiques qu'elle offre. aussi, il est courant

d'utiliser le terme un nuage pour désigner l'infrastructure gérée par un prestataire donné. On peut distinguer trois types principaux de modèles de déploiement pour ces nuages : le nuage privé, le nuage public et le nuage hybride.

8-1-Le nuage prive :

L'infrastructure d'un nuage privé n'est utilisée que par un unique client. Elle peut être gérée par ce client ou par un prestataire de service et peut être située dans les locaux de l'entreprise cliente ou bien chez le prestataire, le cas échéant. L'utilisation d'un nuage privé permet de garantir, par exemple, que les ressources matérielles allouées ne seront jamais partagées par deux clients différents.

8-2-Le nuage public :

L'infrastructure d'un nuage public est accessible publiquement ou pour un large groupe Industriel. Son propriétaire est une entreprise qui vend de l'informatique entant que service.

8-3-Le nuage hybride :

L'infrastructure d'un nuage hybride est une composition de deux types de nuages précédemment cités. Les différents nuages qui la composent restent des entités indépendantes à part entière, mais sont reliés par des standards ou par des technologies propriétaires qui permettent la portabilité des applications déployées sur les différents nuages.

Une utilisation type de nuage hybride est la répartition de charge entre plusieurs nuages pendant les pics du taux d'utilisation.

9- La différence entre le nuage privé et le nuage public :

Dans le cas du nuage public, votre nuage ne vous appartient pas entièrement. Un grand nombre de ressources informatiques sont partagées avec de nombreuses entreprises à travers l'ensemble du réseau Internet. Si ce modèle possède de nombreux avantages en termes de réduction des coûts, de collaboration et d'agilité, pour certaines entreprises, en revanche, il soulève, parfois à juste titre et parfois tort, certaines questions sur la sécurité et la confidentialité des données.

De son côté, le nuage privé propose des ressources informatiques dont l'usage est uniquement réservé à votre entreprise. Vous pouvez héberger votre nuage privé soit sur site, dans votre centre de données (en utilisant une virtualisation et une automatisation à grande échelle), soit hors site chez un fournisseur de services de nuage. Le nuage privé possède la plupart des avantages du

nuage public (options de libre-service, relative capacité de montée en charge et facturation interne, par exemple) mais permet davantage de contrôle et de personnalisation du fait que des ressources dédiées sont à

Votre disposition. Il peut offrir encore plus de flexibilité, ce qui peut rendre son coût prohibitif et amoindrir les économies d'échelle pour certaines entreprises.

10-Avantages et inconvénients du Cloud Computing :[LEF10]

10-1-Avantages :

. Un démarrage rapide : Le Cloud Computing permet de tester le business plan rapidement, à coûts réduits et avec facilité.

. L'agilité pour l'entreprise : Résolution des problèmes de gestion informatique simplement sans avoir à vous engager à long terme.

. Un développement plus rapide des produits : Réduisons le temps de recherche pour les développeurs sur le paramétrage des applications.

. Pas de dépenses de capital : Plus besoin des locaux pour élargir vos infrastructures informatiques.

10-2Inconvénients :

. La bande passante peut faire exploser votre budget : La bande passante qui serait nécessaire pour mettre cela dans le Cloud est gigantesque, et les coûts seraient tellement importants qu'il est plus avantageux d'acheter le stockage nous-mêmes plutôt que de payer quelqu'un d'autre pour s'en charger.

. Les performances des applications peuvent être amoindries : Un Cloud public n'améliorera définitivement pas les performances des applications.

. La fiabilité du Cloud : Un grand risque lorsqu'on met une application qui donne des avantages compétitifs ou qui contient des informations clients dans le Cloud.

. Taille de l'entreprise : Si votre entreprise est grande alors vos ressources sont grandes, ce qui inclut une grande consommation du Cloud. Vous trouverez peut-être plus d'intérêt à mettre au point votre propre Cloud plutôt que d'en utiliser un externalisé. Les gains sont bien plus importants quand on passe d'une petite consommation de ressources à une consommation plus importante.

11- Conclusion :

Le Cloud Computing**,** est la moyen d'avoir un gros espace de stockage externe ce qui est bien pratique pour nos appareils qui travaille avec un système embarquées temps réel et qui possède une capacité de stockage limitée comme les Smartphones et les tablettes et les capteurs des engins.

Par contre cette avantage implique que nos données sont accessibles uniquement en ligne, donc nous sommes extrêmement limité en hors connexion.

Avec le Cloud, l'utilisateur est extrêmement dépendant du réseau. Si vous souhaitez accéder à des données disponibles exclusivement sur le Cloud, et que vous êtes bloqué en cas de panne de réseau, vous ne pourrez pas le faire.

En trouve aussi que la sécurité de vos données est également un point d'achoppement concernant le Cloud. Diverses affaires récentes ont démontré qu'il était possible à des hackers talentueux de pirater les serveurs de stockages de grandes sociétés Internet et d'ainsi accéder à des informations confidentielles.

Chapitre 3-

Tolérance Aux Fautes

-Partie I:La Sureté De Fonctionnement.

-Partie II: Mécanismes de tolérance aux fautes.

-Partie III: la Communication dans Les Systèmes Embarqués.

-Partie IV: Protocole CAN (2.0 A).

1-Introduction :

Les réseaux informatiques présentent actuellement un grand intérêt et leur bon fonctionnement nécessite un haut degré de fiabilité et de disponibilité. Cette fiabilité est réalisée au moyen de plusieurs chemins de connexion entre des emplacements dans un réseau de sorte que lorsqu'un chemin tombe en panne, la transmission est réacheminée avec succès. Ainsi la topologie du réseau fournit une structure complexe de chemins redondants quien tournez fournissez la tolérance aux fautes, et ces principes s'appliquent également à la distribution d'énergie.

Généralement, la fiabilité et la disponibilité des grands systèmes sont calculées en utilisant des programmes informatiques tolérants aux fautes. La plupart des environnements industriels, militaire, spatiale... en ces programmes.

Partie I - La sureté de fonctionnement.

I-1-Introduction :

[FTC]La sureté des systèmes est la propriété des systèmes informatiques qui permet à la fiabilité d'être mis sur le service qu'il fournit. Le service rendu par un système est son comportement tel qu'il est perçu par ses utilisateurs; un utilisateur est un autre système (humain ou physique) qui interagit avec le premier.

Une défaillance du système se produit lorsque le service fourni s'écarte du service spécifié, où la spécification de service est une description convenue du service attendu. L'échec s'est produite parce que le système était erroné ; une erreur est cette partie de l'état du système avec respect au processus de calcule qui est susceptible de conduire à l'échec. La cause jugée ou supposée d'une erreur est une faute.

La sureté de fonctionnement est caractérisée par ses attributs, ses entraves et ses moyens.

I-2-Historique :

Beaucoup a été écrit sur le thème de la sureté de fonctionnement, la fiabilité et de la disponibilité depuis son développement au début des années 50. Le calcul tolérant aux fautes a commencé entre 1965 et 1970, probablement avec les très fiables et largement disponibles systèmes de commutation électronique.

L'informatique tolérante aux fautes a été développée pour les besoins critiques des applications militaires et spatiales. Les missions dans l'espace lointain de la NASA sont coûteuses, car elles nécessitent divers plans de redondance et de récupération pour éviter l'échec total. Les progrès de la

conception des avions militaires ont conduit au développement des commandes de vol électronique, et des systèmes similaires ont ensuite été incorporés dans le Airbus 330 et Boeing 777, dont les commandes de vol sont triplées pour permettre à certains éléments de tomber en panne pendant l'exploitation de l'avion. La réputation de l'ordinateur professionnel Tandem est basé sur l'informatique Non-stop, un système de redondance complet qui améliore la fiabilité de stockage informatique moderne utilise des techniques de matrice redondante et des disques indépendants (RAID) pour relier 50 à 100 disques dans un système rapide et fiable.

I-3-Attributs de sureté de fonctionnement :

Les principales grandeurs de la sureté de fonctionnement (*Dependability*) examinées sont la fiabilité, la disponibilité, la maintenabilité et la sécurité. D'autres grandeurs, moins utilisées, comme la durabilité, la continuité, la serviabilité ne seront pas étudiés.

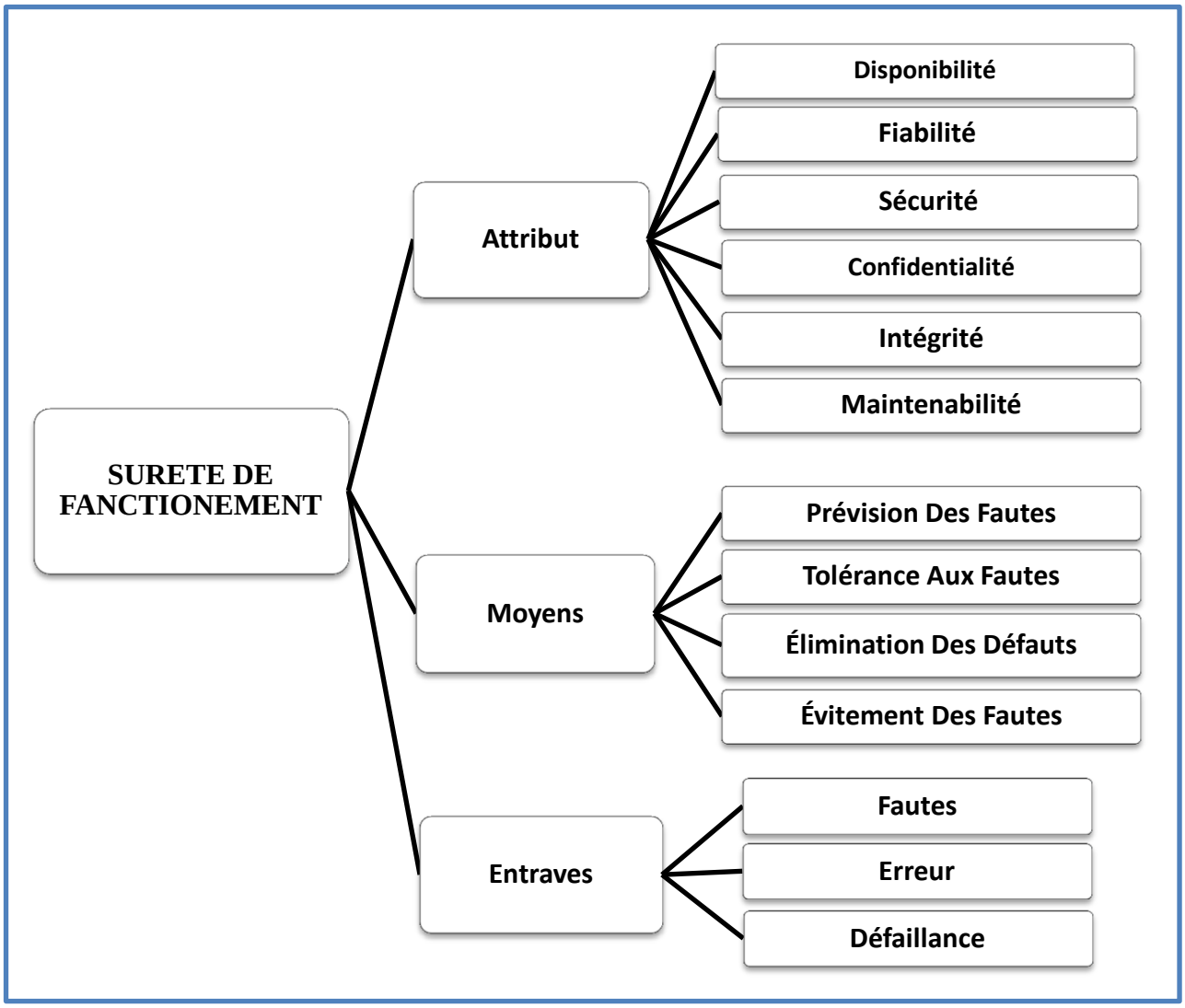


Figure 3.1 : les attributs de sureté de fonctionne.

I-3-1-Fiabilité :

Selon [MEF]La fiabilité (Reliability en anglais) est l'aptitude d'une entité à accomplir les fonctions requises dans des conditions données pendant une durée donnée. Elle est caractérisée par la probabilité R (t) que l'entité E accomplisse ces fonctions, dans les conditions données pendant l'intervalle de temps [0, t], sachant que l'entité n'est pas en panne à l'instant 0.

R(t) = Probabilité pour qu'un système soit continûment en fonctionnement sur une période donnée (entre 0 et t).

Nous admettrons par la suite que le temps est la variable principale dont dépend la fiabilité. Pour certains appareils, il peut être plus judicieux de prendre une autre variable : nombre de cycles d'ouverture-fermeture pour un relais, nombre de tours pour un moteur, nombre de kilomètres pour une voiture, etc.

Selon **[DFT]** la fiabilité est «Le mesure dans laquelle un système critique bénéficie de la confiance de ses utilisateurs » ; Ceci est clairement une définition qui se concentre davantage sur la façon dont l'utilisateur perçoit le système que sur la façon dont le système est en fait digne de confiance. Cela reflète le degré de confiance de l'utilisateur dans le fait que le système fonctionnera comme les utilisateurs l'attendent et en particulier sans échec .

La contrepartie négative de la fiabilité, le manque de fiabilité, représente la probabilité conditionnelle que le système ne fonctionne pas correctement pendant l'intervalle **[t0, t]**, étant donné que le système fonctionnait correctement à l'instant **t0**. Le manque de fiabilité est également connu sous le nom de probabilité de défaillance.

I-3-2-Disponibilité :

La disponibilité (Availability en anglais) est l'aptitude d'une entité à être en état d'accomplir les fonctions requises dans les conditions données et à un instant donné. Elle est caractérisée par la probabilité A (t) que l'entité E soit en état, à l'instant t, d'accomplir les fonctions requises dans des conditions données.

A (t) = P [E non défaillante a l`instant t] Probabilité pour qu'un système soit en fonctionnement à un instant t donné.

[DFT] La disponibilité peut être approximée comme le temps total pendant lequel un système a été capable de fournir le service prévu divisé par le temps écoulé pendant lequel le système a été en

fonctionnement, c'est-à-dire le pourcentage de temps pendant lequel le système est disponible pour fonctions attendues. La disponibilité à l'état d'équilibre peut être prouvée.

Ass = MTTF / MTTF + MTTR.

I-3-3- Maintenabilité :

La maintenabilité (Maintainability en anglais) est l'aptitude d'une entité à être maintenue ou rétablie dans un état dans lequel elle peut accomplir une fonction requise, lorsque la maintenance est réalisée dans des conditions données avec des procédures et des moyens prescrits. Elle est caractérisée par la probabilité M (t) que l'entité E soit en état, à l'instant t, d'accomplir ses fonctions, sachant que l'entité était en panne à l'instant 0.

M (t) = P [E est réparé sur [0, t]] Probabilité pour qu'un système en panne à l'instant 0 soit réparé à l'instant t.

[DFT] La maintenabilité est une fonction du temps représentant la probabilité qu'un système défaillant sera réparé dans un temps inférieur ou égal à t. La maintenabilité peut être estimée comme : **M (t) = 1 - exp -µt**,

µ étant le taux de réparation, supposé constant

I-3-4- Sécurité :

La sécurité (Safety en anglais) est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. Elle est caractérisée par la probabilité S (t) que l'entité E ne laisse pas apparaître dans des conditions données, des événements critiques ou catastrophiques.

S (t) = P [E évite des évènements critiques ou catastrophiques sur [0, t]]

Probabilité pour qu'un système soit continûment en fonctionnement non catastrophique sur une période donnée (entre 0 et t).[MEE] Il est à noter que dans le domaine de l'informatique la sécurité a souvent deux facettes :

- **la sécurité-innocuité (Safety en anglais)** qui vise à se protéger des défaillances catastrophiques
- **la sécurité-confidentialité (Security en anglais)** qui correspond à la prévention d'accès ou de manipulations non autorisées de l'information et concerne la lutte contre les fautes intentionnelles.

I-3-5- L'intégrité :

Correspond à l'absence d'altération inappropriée de l'information. Les propriétés d'un système sûr de fonctionnement sont souvent corrélées à son domaine d'application. Ainsi, un système de contrôle-commande d'un avion est dit sûr de fonctionnement, si et seulement s'il est d'une part disponible et d'autre part fiable. D'autres attributs comme la confidentialité ou l'intégrité peuvent être plus appropriés pour qualifier d'autres systèmes sûrs de fonctionnement, comme les applications bancaires ou du domaine de la santé publique.

I-4-Grandeurs moyennes caractéristiques de la sûreté de fonctionnement :

Selon **[DFT]**Si un système tombe en panne, il est logique de se demander combien de temps le système peut devrait fonctionner sans problèmes. Un tel chiffre s'appelle le temps moyen avant l'échec

(MTTF). MTTF est défini comme le temps prévu pendant lequel un système fonctionnera avant l'apparition de son premier échec.

Une autre propriété importante est le temps moyen de réparation (MTTR). MTTR est défini

Comme le temps moyen requis pour réparer un système. Il est souvent spécifié par des moyens d'un taux de réparation **μ**, c'est-à-dire le nombre moyen de réparations effectuées par unité de temps.

Le temps moyen entre les pannes (MTBF) est le temps moyen entre deux défaillances consécutives d'un système. Ceci est légèrement différent de MTTF qui concerne lors de la toute première panne d'un système. La relation suivante est vraie: **MTBF = MTTF + MTTR**

Comme il est généralement vrai que MTTR est une petite fraction de MTTF, il est généralement autorisé pour supposer que MTBF ≈ MTTF.

I-4-1- Les entraves : [FTC]

La tolérance aux fautes est réalisée par le traitement des erreurs et par le traitement des fautes. Le traitement d'erreur vise à supprimer les erreurs de l'état de calcul, si possible avant l'occurrence de l'échec; le traitement des fautes vise à empêcher la réactivation des fautes. voilà quelque notion sur les fautes, les erreur et les défaillance. La vue structurelle d'un système permet de préciser la pathologie des fautes. La création et les mécanismes de manifestation des fautes, des erreurs et des échecs peuvent être résumés comme suit:

I-4-2-Les fautes : [LLY93]

Une Faute est un défaut, ou une imperfection, ou un manque dans un système, logiciel ou composant matériel. Il est défini de manière générique comme la valeur jugée ou hypothétique la cause dimensionnée d'une erreur. Les fautes peuvent avoir leur origine dans les limites du système (fautes internes) ou à l'extérieur, c'est-à-dire dans l'environnement (fautes externes). En particulier, la faute interne est dite active lorsqu'elle produit une erreur, et dormante (ou latente) quand ce n'est pas le cas. Une faute dormante devient une faute active lorsqu'elle est activée par le processus de calcul ou l'environnement. La latence des fautes est définie comme la longueur de temps entre l'apparition d'un défaut et l'apparition du correspondant erreur, ou le laps de temps entre l'apparition d'une faute et sa suppression.

[DFT]Les fautes peuvent être classées selon cinq points de vue cause phénoménologique, nature, phase de création ou d'occurrence, situation par rapport aux limites du système, persistance. Toutes les combinaisons ne peuvent pas donner passer à une classe de défauts ce processus ne définit que 17 classes de défauts, résumées dans la **Figure12**. Ces classes ont été divisées en trois «groupes», connus sous le nom de classes de défauts combinées. Les classes de pannes combinées les plus pertinentes dans ce chapitre sont maintenant brièvement caractérisées:

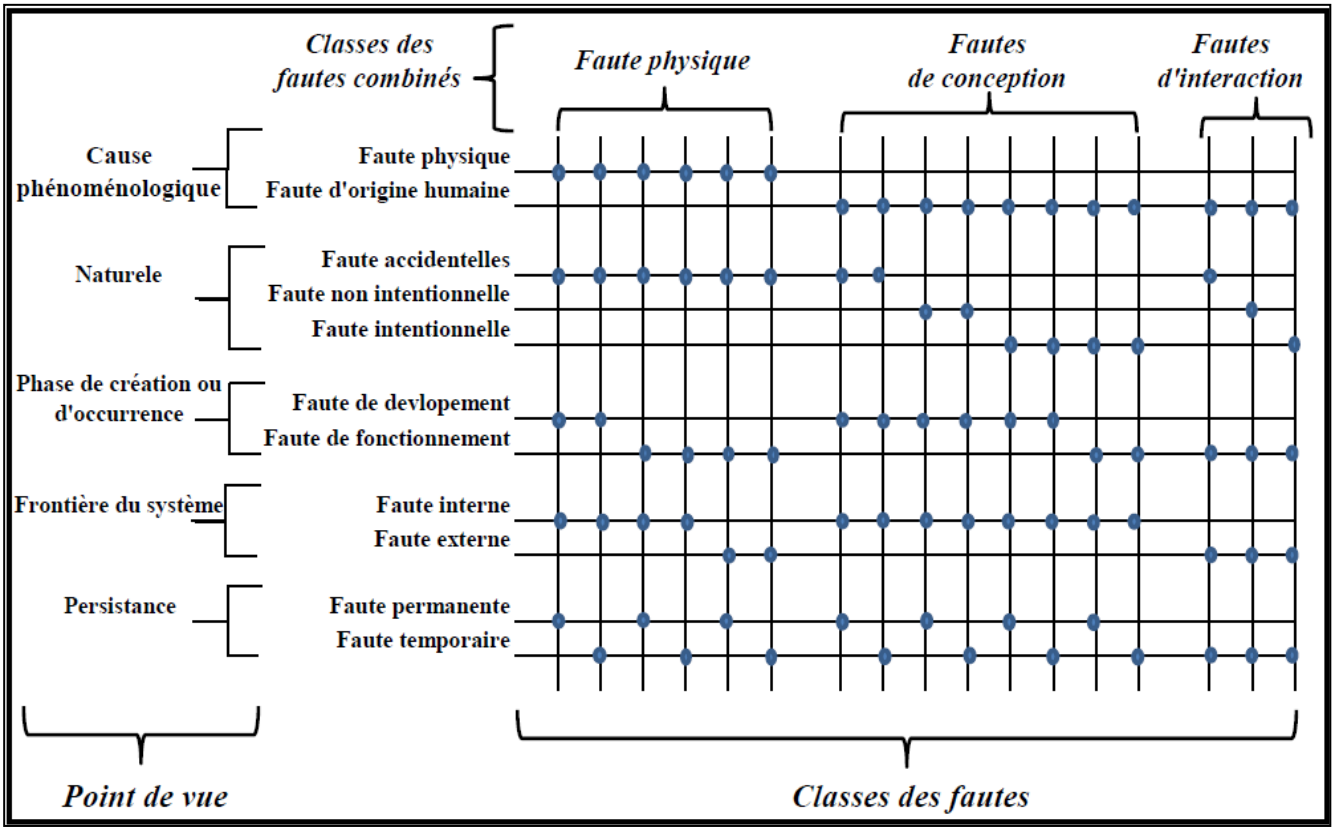


Figure 3.2 : schéma de classification des fautes [DFT].

Les classes de pannes combinées les plus pertinentes dans ce chapitre sont maintenant brièvement caractérisées:

I-4-2-1-Fautes physiques:

- Fautes physiques permanents, internes. Cette classe concerne les défauts qui ont leur origine dans les composants matériels et sont continuellement actifs. Un exemple typique est donné par la Faute correspondant à un composant usé.
- Fautes physiques temporaires, internes (également appelés Fautes intermittents). Ce sont généralement des Fautes physiques internes qui deviennent actifs en fonction d'une condition ponctuelle.
- Fautes physiques permanents, externes. Ce sont des défauts induits sur le système par l'environnement physique.
- Fautes physiques temporaires, externes (également appelés Fautes transitoires) Ce sont des Fautes induits par des phénomènes environnementaux.

I-4-2-2-Fautes de conception:

Fautes de conception intentionnels, mais non malveillants, permanents / temporaires. Ce sont essentiellement des compromis introduits au moment de la conception. Un exemple typique est un dimensionnement insuffisant (sous-estimation de la taille d'un champ donné en un protocole de communication, et ainsi de suite).

- Fautes de conception accidentels, permanents (également appelés fautes systématiques ou Bohr bugs), des algorithmes défectueux qui se transforment systématiquement en les mêmes erreurs en présence des mêmes conditions d'entrée et états initiaux par exemple, un diviseur qui peut entraîner une erreur de division par zéro.
- Fautes de conception accidentelle et temporaire (connus sous le nom de Heisen bugs, pour «bugs of Heisenberg », d'après leur caractère insaisissable), alors que les fautes systématiques ont un comportement évident et déterministe, ces bugs dépendent de combinaisons subtiles de l'état et de l'environnement du système.

I-4-2-3- Fautes d'interaction:

- Fautes temporaires, externes, opérationnels, d'origine humaine, accidentels. Ces en exclure les Fautes de l'opérateur, dans lesquels un opérateur n'exécute pas correctement son rôle dans l'exploitation du système.

- Fautes temporaires, externes, opérationnels, d'origine humaine et non malveillants «négligence, interactions ou problèmes d'utilisation incorrecte. (exemples incluent mal mots de passe choisis et mauvais paramétrage du système).
- Fautes malveillants temporaires, externes, opérationnels, d'origine humaine. Cette classe inclut les fautes de réplication dite malveillants, c'est-à-dire les fautes qui se produisent lorsque les informations répliquées dans un système deviennent incohérentes, soit parce que les répliques censées fournir des résultats identiques ne le font plus, ou parce que l'agrégat des données des différentes répliques n'est plus cohérent avec les spécifications du système.

I-5-Le traitement des fautes :

Le Traitement des Fautes [SDT] est destiné à éviter qu'une ou des fautes ne soient activées à nouveau. La première étape du traitement de faute est le diagnostic des fautes, qui consiste à déterminer les causes des erreurs, en termes de localisation et de nature. Puis viennent les actions destinées à remplir l'objectif principal du traitement de faute: empêcher une nouvelle activation des fautes, donc de les passiver. Ceci est accompli en retirant les composants considérés comme fautifs du processus d'exécution ultérieur des Erreur [FTC], cette partie de l'état du système avec respect au processus de calcule qui est susceptible de conduire à l'échec. La cause jugée ou supposée d'une erreur est une faute.

Une erreur est donc la manifestation d'un défaut dans le système, et une défaillance est l'effet d'une erreur sur le service.

Une erreur peut être latente ou détectée, soit par un algorithme ou un mécanisme de détection au sein du composant. Une erreur peut se propage en général, à partir d'un composant à un autre; en se propageant, une erreur crée d'autres «nouvelles» erreurs. Une erreur peut donc provenir.

I-6-Mécanisme de traitement des erreurs :

Le Traitement des Erreurs [SDT] est destiné à éliminer les erreurs, si possible avant qu'une défaillance ne survienne. Deux méthodes sont possibles pour traiter les erreurs:

- **le recouvrement** : où l'on substitue un état exempt d'erreur à l'état erroné (par point de reprise ou par poursuite sur un état prédéfini),

- **la compensation** : où l'état erroné comporte suffisamment de redondance pour permettre la délivrance d'un service approprié.

Lorsque le recouvrement est utilisé, il faut pouvoir détecter l'erreur. Au contraire, la compensation, qui permet le masquage des erreurs, s'en affranchit. Cependant les mises en œuvre pratiques de techniques de compensation comportent généralement de la détection d'erreur.

Une défaillance du système se produit lorsque le service fourni s'écarte du service spécifié, où la spécification de service est une description convenue du service attendu. L'échec s'est produit parce que le système était erroné.

Une défaillance de composant se produit lorsqu'une erreur affecte le service fourni par le composant en réponse à une ou plusieurs demandes d'un autre composant. Une défaillance des composants peut entraîner une panne du ou des composants auxquels il fournit le service. Ces mécanismes permettent de compléter la «chaîne fondamentale»:

En conclusion, nous observons que les fautes, les erreurs et les échecs sont tous des circonstances indésirables. L'affectation des termes fautes, erreur, défaillance prend simplement en compte l'usage courant:

a) fautes évitement, tolérance et diagnostic,

b) détection et correction des erreurs,

c) taux d'échec. Il est à noter que les étiquettes «fautes» et «erreur» sont parfois inter changées.

I-7-Techniques pour la construction de système sûr :

La réalisation d'un système informatique [FCT] fiable nécessite l'utilisation combinée d'un ensemble des méthodes qui peuvent être classées en Techniques préventives ou curatives, que nous décrivons brièvement ici :

a- L'évitement des fautes (*Fault avoidance*) **:**comment éviter, par construction, l'introduction des fautes; Certaines fautes sont imprévisibles ou inévitables comme par exemple les fautes humaines, le vieillissement du matériel ou les catastrophes naturelles. Ainsi, la tolérance aux fautes est un mécanisme indispensable afin d'atteindre les conditions de fiabilité extrême des systèmes critiques.

b-Tolérance aux fautes *(Fault tolerance*):Comment fournir, par redondance, un service conforme à la spécification malgré les fautes.

I-8-Techniques pour la validation de systèmes sûrs :

La validation d'un système sûr et fiable se réalise par l'une des deux méthodes suivantes :

I-8-1-Prévision des fautes (*dépendability évaluation*):

Comment estimer, par évaluation, la présence, la création, et les conséquences des fautes. L'ensemble de l'architecture considérée comme un tout continue par l'utilisation de redondances et de rendre le service attendu en dépit de l'existence de fautes.

I-8-2-Elimination des fautes (*fault removal*):

Comment minimiser, par vérification, la présence de faute en utilisant des tests et par validation pendant la phase de conception et de réalisation, et par la maintenance durant la vie opérationnelle du système [FCT].Le raisonnement précédent explique pourquoi, dans les définitions données, la suppression des fautes et les fautes les prévisions sont rassemblées en validation. La validation est souvent limitée à ce qui a été appelé vérification; dans ce cas, ces deux termes sont souvent associés, la distinction étant lié à la différence entre «construire le système correctement» (lié à la vérification) et «construire le bon système» (lié à la validation). Ce qui est proposé est simplement une extension de cette concept ; la réponse à la question "est-ce que je construis le bon système?" étant complété par "pour combien de temps cela sera-t-il vrai?".

De plus, la suppression des défauts est généralement étroitement associée à la défaillance évitement, tous deux constituant une prévention des fautes. En plus de souligner la nécessité de valider le procédures et mécanismes de tolérance aux fautes, considérant l'élimination des fautes et la prévision des fautes comme deux constituants d'une même activité ; la validation est d'un grand intérêt en ce qu'elle permet une meilleure compréhension de la notion de couverture, et donc d'un problème important introduit par la récursivité, la validation de la validation, ou comment gagner en confiance dans les méthodes et outils utilisés pour renforcer la confiance dans le système. La couverture se réfère ici à une mesure de la représentativité des situations auxquelles le système est soumis lors de sa validation par rapport à aux situations réelles auxquelles il sera confronté au cours de sa vie opérationnelle.

Parmi les méthodes mentionnées ci-dessus, la tolérance aux fautes représente l'outil de base pour

Les techniques et les outils présentés dans ce chapitre. Pour cette raison, il est discuté dans plus de détails dans la suite.

Partie II-Mécanismes de tolérance aux fautes.

II-1-Introduction :

Tous les éclaircissements qui apparaissent dans les définitions de base sont en fait des objectifs qui ne peuvent pas être pleinement atteint, car toutes les activités correspondantes sont des activités humaines, et donc imparfaites. Ces imperfections apportent des dépendances qui expliquent pourquoi il ne s'agit que de l'utilisation combinée du ces préférence à chaque étape du processus de conception et de mise en œuvre qui peuvent conduire à un système informatique fiable. Ces dépendances peuvent être esquissées comme suit ; malgré l' évitement des fautes au moyen de méthodologies de conception et de règles de construction (imparfaites pour être réalisable), des fautes se produisent; d'où la nécessité de supprimer les fautes; la suppression des fautes est elle-même imparfaite, tout comme les composants standard du système, d'où la nécessité de prévoir les fautes; notre augmentation de la dépendance vis-à-vis des systèmes informatiques entraîne une tolérance aux fautes, qui à son tour nécessite des règles de construction, et donc élimination des fautes, prévision des défauts, etc.[FCT]

II-2- Définitions :

Diverses idées issues de l'informatique tolérante aux fautes sont maintenant utilisées dans presque tous les ordinateurs commerciaux, militaires et spatiaux les systèmes; dans les industries du transport, de la santé et du divertissement; dans les établissements d'enseignement et de gouvernement; dans les systèmes téléphoniques; et dans les fossiles et centrales nucléaires. Les développements rapides de la microélectronique ont conduit à des conceptions complexes; par exemple, une automobile de luxe peut avoir 30 à 40 microprocesseurs connectés par un réseau local! Ces dessins doivent être réalisés en utilisant techniques tolérantes aux fautes pour fournir une fiabilité, une disponibilité et une sécurité logicielles et matérielles significatives

Le calcul tolérant aux fautes est un terme générique décrivant des techniques de conception redondantes avec des composants en double ou des calculs répétés permettant un fonctionnement ininterrompu **(tolérant)** en réponse à une défaillance de composant **(fautes)**. Parfois, les catastrophes du système sont causées par le non-respect des principes de redondance et l'indépendance de l'échec, qui est évidente rétrospectivement.

II-3-Architecture :

En Concédèrent que l'architecture de la tolérance aux fautes (la manière dont un système est organisé). Il est nécessaire de couvrir les techniques requises pour analyser la fiabilité et la disponibilité des systèmes tolérants aux fautes. la comparaison des conceptions de ses systèmes nécessite un compromis entre le coût, le poids, volume, fiabilité et disponibilité. Les fondements

Mathématiques de ces analyses sont la théorie des probabilités, la théorie de la fiabilité, les taux de défaillance des composants et fonctions de densité de défaillance des composants.

[DFT]Selon (Anderson & Lee, 1981), la tolérance aux fautes peut être décomposée en deux sous-techniques: le traitement des erreurs et le traitement des fautes.

Le traitement des erreurs vise à supprimer les erreurs de l'état de calcul (si possible, avant l'apparition de la panne). Il peut être basé sur les primitives suivantes (Laprie, 1995):

Détection d'erreur, qui se concentre sur la détection de la présence dans le système de latente erreur avant leur activation. Cela peut être fait, par exemple, au moyen de auto-tests ou par comparaison avec des calculs redondants (*Rushby, 1994*).

• Diagnostic d'erreur, c'est-à-dire évaluation des dommages causés par les erreurs détectées ou par des erreurs propagées avant la détection.

• La récupération d'erreur consiste en des méthodes permettant de remplacer un état erroné par un état sans erreur. Ce remplacement prend l'une des forme sa montrer dans le reste de cette partie de ce chapitre 3.

II-4-Mécanismes de redondance pour la tolérance aux fautes :

On peut arriver à obtenir des systèmes ayant un bon niveau de SdF soit en fiabilisant les composants et l'architecture du système, soit en mettant en œuvre des mécanismes de redondance (les deux options n'étant pas exclusives l'une de l'autre). Quoi qu'il en soit, les méthodes et moyens mis en œuvre dépendent essentiellement du type de fautes considéré et du nombre de fautes que l'on désire prendre en compte. [SDT]

En effet, un problème fondamental est l'efficacité des redondances. Une redondance n'est effective, c'est-à-dire à même de fournir la capacité de tolérer une faute que:

- pour les types de fautes pour lesquelles elle a été conçue,
- si les mécanismes de mise en œuvre de la redondance fonctionnent correctement à partir de l'instant où une faute s'est manifestée.

II-4-1-Laredondance dynamique :

Encore appelée redondance sélective, la redondance dynamique est la forme la plus "naturelle" de redondance. Elle consiste à mettre en parallèle n+1 unités (pour tolérer n fautes): le système n'est défaillant que si toutes les unités le sont.

Un tel type de redondance nécessite que les éléments détectent la faute lorsqu'elle survient dans l'élément actif et de basculer sur un autre élément. Il y a donc interruption des fonctions lorsqu'une faute survient. La question est de savoir qui détecte et décide du basculement (application ou système de communication). Il existe deux méthodes pour mettre en œuvre un tel type de redondance:

II-4-1-1-La redondance dynamique active:

Les n unités effectuent les mêmes tâches, les unités en attente sont toujours prêtes à prendre la relève en cas de faute dans l'unité active.

II-4-1-2-La redondance dynamique passive:

L'unité en attente effectue des tâches différentes de l'unité active, généralement de moindre criticité et ne nécessitant pas de redondance; lorsqu'une faute survient dans l'unité active, une des unités en attente doit être activée afin de poursuivre l'exécution des tâches interrompues.

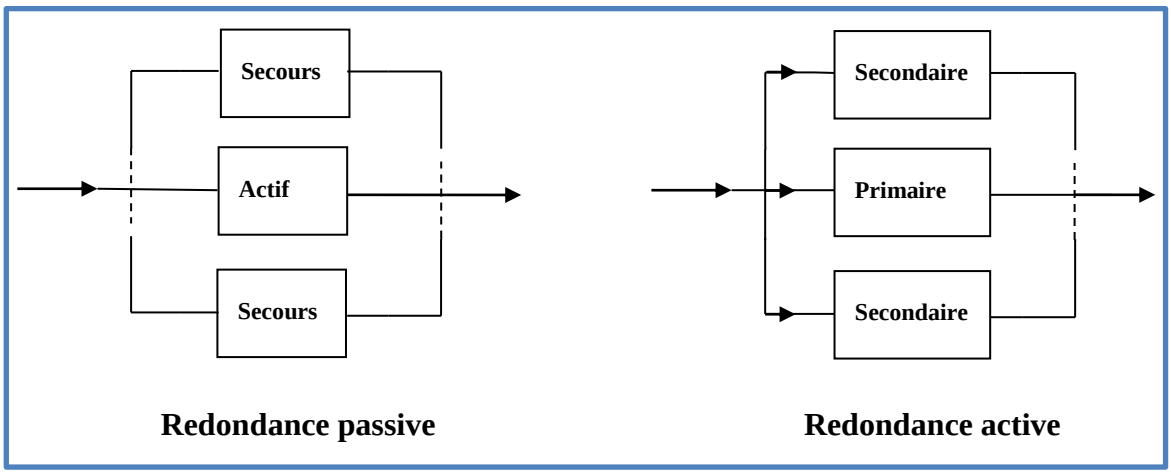


Figure 3.3 : Schéma deux types de redondance dynamique.

II-4-2-La redondance statique :

La deuxième grande catégorie de redondance, dite statique ou encore massive, fait appel à une fonction de majorité: on dispose de 2n+1 (où n est le nombre de fautes tolérées) unités en parallèle, et le système fonctionne tant que n+1 au moins des unités fonctionnent. La fonction de majorité est assurée par un vote "2n-l parmi 2 " Lorsqu'une faute survient, les fonctions ne sont pas

interrompues: la faute est masquée. L'application la plus connue d'une telle structure est le TMR (Triple Modular Redundancy), où le vote est de 2 parmi 3, et une faute unique est tolérée.

Ces techniques de recouvrement sont généralement accompagnées de détection et signalisation d'erreur afin de pouvoir assurer la réparation de l'unité défaillante le cas échéant.

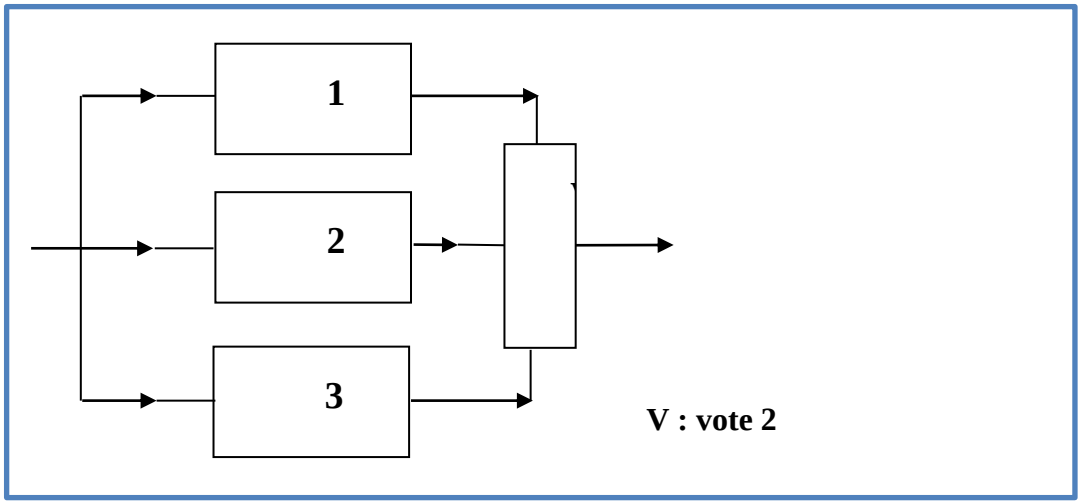


Figure 3.4 : Schéma de redondance statique (T.M.R)

II-5-Conclusion :

Quand le thème de la sûreté de fonctionnement sera abordé dans la suite de cette étude, ce sera uniquement sous l'aspect "tolérance aux fautes - traitement des erreurs". De plus, les techniques qui nous intéressent sont celles applicables lors des systèmes de communication dans le système embarqué. D'une manière générale, la distinction entre une tolérance aux fautes du système de communication et un système de communication permettant aux applications de tolérer les fautes devra être faite. Dans le premier cas, c'est les faute des bus et des canaux de communication réseau qui se fait tolère afin d'offrir aux applications, de manière transparente, un service sûr de fonctionnement).

Dans le second cas, le système de communication offre un ensemble d'outils aux applications pour leur permettre de construire un système global sûr de fonctionnement.

Partie III- La communication dans les systèmes embarqués.

III-1-Introduction :

Avec l'avènement grandissant des liaisons radiofréquences, il est nécessaire d'inclure également dans cet inventaire les nouveaux protocoles de communication permettant de réaliser des réseaux « sans fil » (radiofréquence), de façon à pouvoir étendre le champ et les distances de communication. Très souvent, les réseaux commencent à être épris de certaines libertés métriques (distances de l'ordre de 10 à 50 m), hors tout cordon ombilical et les passerelles de type wired/wireless (câblé/sans fil) sont déjà légions sur le marché (par exemple les passerelles de type CAN/Bluetooth ou Wifi).

Les protocoles les plus simples ont été élaborés pour des raisons de coût et de fonctionnalités. Bien entendu, les performances en termes de bande passante, de débit, de sécurité et de fonctionnalités sont plus restreintes. Parmi ceux-ci, les plus anciens, bien connus, sont les protocoles I2C, D2B et SPI et le LIN.

III-2-La communication comment ça marche :

D'autre part la communication entre les systèmes embarqués se fait par des méthodes compliquées ;Il s'agit des protocoles souvent très performants et très spécialisés, orientés vers des applications particulières. Pour des applications câblées, (LIN,TTCAN, FlexRay, Safe-by-Wire IP5, IEEE 1394, I2C/D2B (ancien),MOST, Ethernet automobile, CPL (powerline)..), Ce sont des réseaux rapides, performants, orientés applications dites temps réel, parfois auto-alimentés, sécurisés, redondants, ainsi que des protocoles pour applications radiofréquences tels que DECT, Bluetooth, ZigBee, Wifi, FC, RKE, PKE, Passive Go, TPMS, etc.

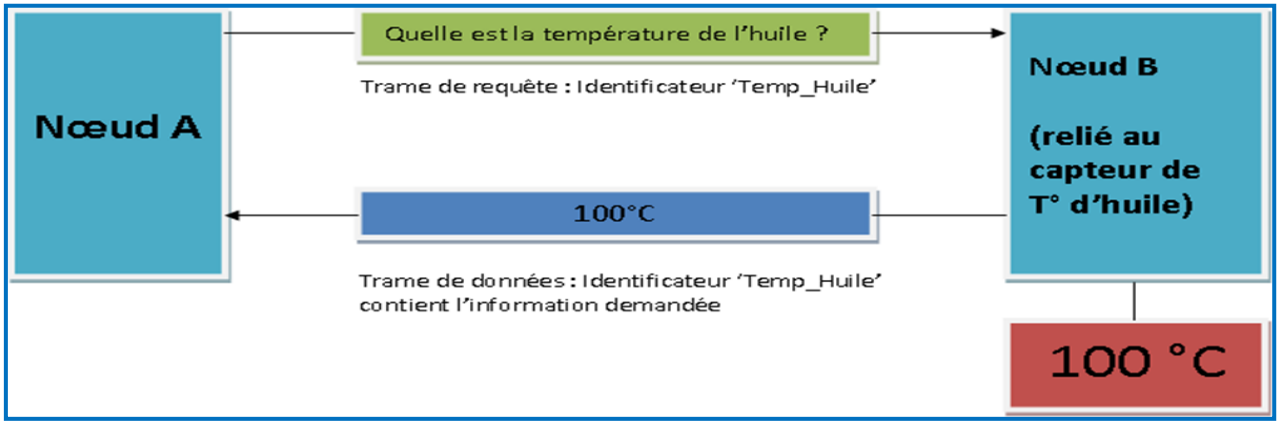


Figure 3.5 : exemple de Communication entre deux nœuds

III-3-Conséquences des contraintes temporelles sur le système de communication :

Un système qui doit respecter des contraintes de temps doit être à même d'assurer deux points fondamentaux:

- Offrir des performances globales suffisantes, par le biais d'un temps de latence faible (le temps pour qu'une information transite d'un point à un autre doit être inférieur à la limite spécifiée);

- Savoir gérer ces contraintes temporelles (par exemple assimiler un dépassement de délai à une faute et le traiter en conséquence).

III-4-Conséquences de la tolérance aux fautes sur le système de communication :

L'un des mécanismes que l'on peut mettre en œuvre pour tolérer les fautes est la redondance pour les réseaux de communication.

D'une part, la redondance du médium de communication (nécessaire pour offrir un réseau tolérant les fautes) implique des protocoles sachant la gérer: généralement, il s'agit de redondance dynamique.

- Soit la redondance est active, et il faut alors envoyer les messages sur deux voies différentes et prendre en compte à la réception l'arrivée d'un même message sur deux canaux distincts;
- Soit la redondance est passive, ce qui est le cas le plus fréquent, car le plus simple à mettre en œuvre, et il faut alors gérer le basculement du réseau primaire au secondaire en cas de faute (ce qui peut être fait par l'application ou par le système de communication).

D'autre part, la redondance des calculateurs implique l'envoi de messages identiques à des entités différentes mais assurant la même fonction, d'où un besoin du service de diffusion. Ce service peut être pris en charge:

- par F application (cette dernière prend alors en compte le fait d'avoir à exécuter n fois la même tâche).
- par le système de communication qui offre alors un service aux applications pour la tolérance aux fautes (par le biais d'un adressage adéquat, il émet le même message à plusieurs destinataires).

La technologie de cloud computing nous apporte un nouveau mode de service à l'usage des données, des applications et des ressources informatiques via le réseau.

III-5-Conclusion :

Les systèmes embarqués sont aujourd'hui au cœur de la convergence électronique, informatique, télécommunications et réseaux. Cette convergence impose de se familiariser avec des disciplines variées, notamment le protocole CAN et l'un des systèmes qui assure une performance majeur aux niveaux de transmission des messages entre les nœuds à travers des Canales de communication.

Ce choix se justifie aisément par le fait que les systèmes embarqués sont aujourd'hui massivement communicants et que le domaine des communications et réseaux constitue un des principaux secteurs de l'embarqué.

Partie IV- Protocole CAN.

IV-1-Introduction :

Le bus CAN (Control Area Network) est un moyen de communication série qui supporte des systèmes embarqués temps réel avec un haut niveau de fiabilité. Ses domaines d'application s'étendent des réseaux moyens débits aux réseaux de multiplexages faibles coûts. Il est avant tout à classer dans la catégorie des réseaux de terrain utilisé dans l'industrie.

Dans cette partie de chapitre, nous tenons à présenter une synthèse sur le protocole CAN.

A fin essayer garantir le but d'assurer la transparence du produit et une plus grande flexibilité de mise en œuvre, le protocole CAN (fournissant une « liaison série asynchrone avec multiplexage temporel des données») est décrit et divisé essentiellement selon la division de la norme en différentes couches de l'ISO/OSI (Organisation internationale de normalisation) / (Interconnexion de systèmes ouverts).

Sa conception réseau est une version simplifiée du modèle OSI. En effet, nous n'aurons pas besoin d'interconnecter différents type de réseaux

IV-2-Protocole Can et Couches OSI (*Open Systems Interconnection*) :

Pour le protocole CAN, le document de référence développé par la société Bosch définie les deux premières couches (dites couches basses).

La Couche Physique : La première couche du modèle a pour but de conduire les éléments binaires jusqu'à leur destination sur le support physique. Elle fournit les moyens matériels nécessaires à l'activation, au maintien et à la désactivation de ces connections physiques. Cette couche gère la représentation du bit (codage, timing, synchronisation), et définit les niveaux électriques, optiques,... des signaux ainsi que le support de transmission.

La Couche Liaison : elle fournit les moyens fonctionnels nécessaires à l'établissement, au maintien et à la libération des connexions entres les entités du réseau. Cette couche (aussi appelée couche de communication de données, Data Link Layer) devra notamment corriger les erreurs qui ont pu se produire au premier niveau (même s'il est impossible de corriger toutes les erreurs).

La Couche Application :c'est la dernière couche du modèle OSI. Elle donne aux applications le moyen d'accéder aux couches inférieures. Cette couche n'est bien sûr pas vide pour le protocole CAN, mais sa spécification est laissée à l'utilisateur.

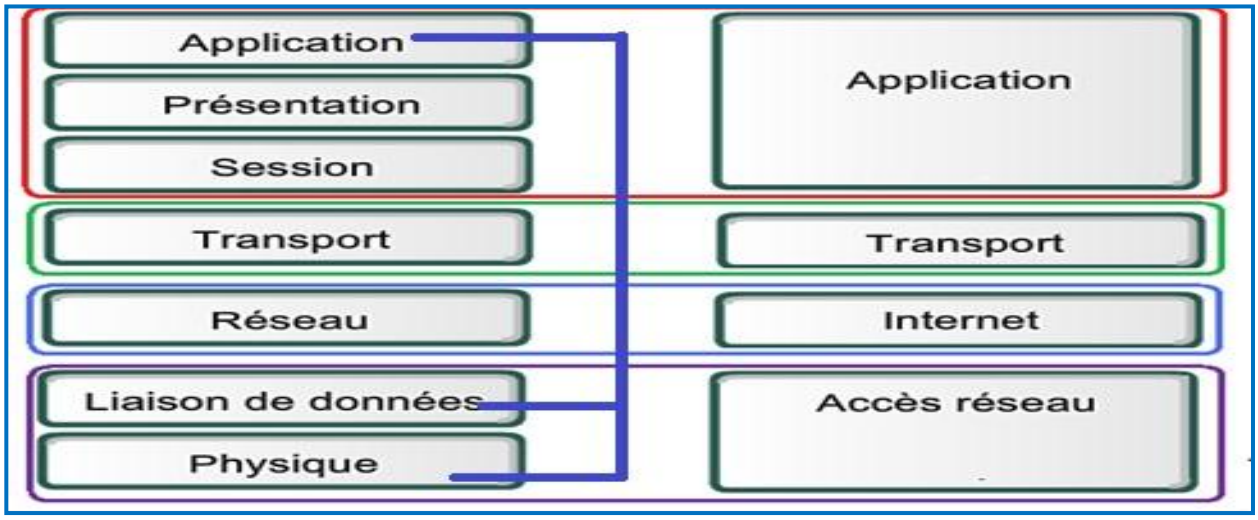


Figure 3.6 : les 3 couches OSI utilisées par le bus CAN

IV-3-Les principales propriétés de la structure protocolaire du CAN :

La structure du protocole du bus CAN possède implicitement les principales propriétés suivantes :

- hiérarchisation des messages.
- garantie des temps de latence.
- souplesse de configuration.
- réception de multiples sources avec synchronisation temporelle.
- fonctionnement multimètre.
- détections et signalisations d'erreurs.
- retransmission automatique des messages altérés dès que le bus est de nouveau au repos.
- Distinction d'erreurs : d'ordre temporaire ou de non-fonctionnalité permanente au niveau d'un nœud, déconnexion automatique des nœuds défectueux.

IV-4-Topologie du CAN :

Il s'agit d'une topologie en BUS. C'est à dire que tous les nœuds sont connectés au même câble. L'avantage d'utiliser un BUS est une réduction significative du câblage. Cela permet également d'insérer un nœud comme on le souhaite dans l'architecture sans avoir a déclarer celui-ci.
Le principal désavantage de cette technologie est la défaillance de l'ensemble des nœuds en cas de rupture d'un câble.
Ci-dessous, voici un schéma qui montre un ensemble de noeuds autours d'un BUS:

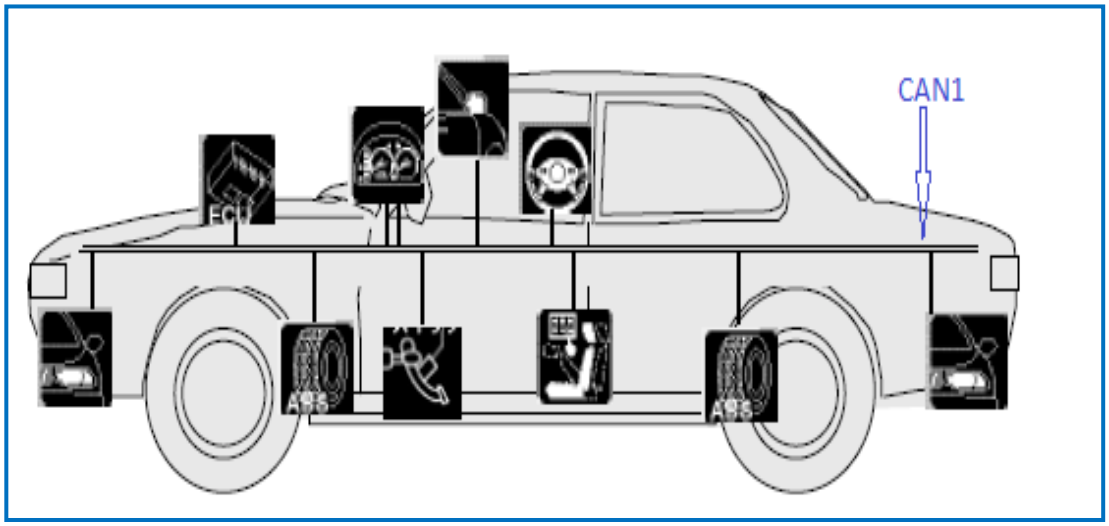


Figure 3.7 : exemple bus CAN dans une automobile

IV-5- la couche physique du CAN

IV-5- 1-Les principes généraux –CAN 2.0A-CAN 2.0B :

Dans le cas d'applications utilisant des liaisons filaires, le support de communication se présente très souvent sous la forme d'une paire différentielle parallèle ou torsadée comme montrée dans la Figure 18, et donc avec des niveaux électriques bien définis, décrivant les niveaux logiques appelés « 0 » et « 1 ». Lorsque le bus est inactif, il n'y a aucune activité et l'écran de l'oscilloscope reste désespérément vide, comme dans le cas de tout bus asynchrone. Mais ce sont des « stations d'action » quand il commence à bouger.

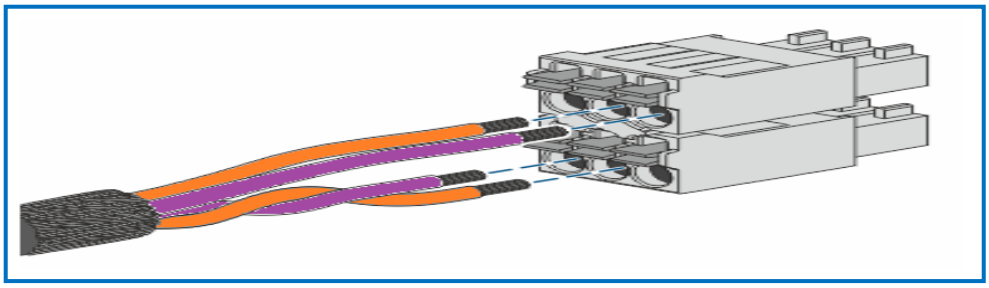


Figure 3.8 : exemple d'un câblage torsadé

IV-5-2-Spécification du câble torsadé :

- Vitesse de propagation nominale : 5 $ns.m^{-1}$
- Impédance nominale : 70 $mOhm.m^{-1}$
- Section : dépendante de la distance et du nombre de nœuds :
- De 0,1 mm^2 pour 40m à 0,4 mm^2 pour une centaine de nœuds à 450m.

IV-5-3-Signal :

Tension différentielle avec retour commun

- Niveau Récessif : $V_{CAN\_H} - V_{CAN\_L} = 0V$ (tolérance : de -500mV à +50mV)
  - ✓ Seuils de détection : Si V_{CAN_H} est inférieur ou égale à $V_{CAN\_L} + 0,5V$
- Niveau Dominant : $V_{CAN\_H} - V_{CAN\_L} = 2V$ (tolérance : de +1,5V à +3V)
  - ✓ Seuils de détection : Si V_{CAN_H} est au moins supérieur à $V_{CAN\_L} + 0,9V$

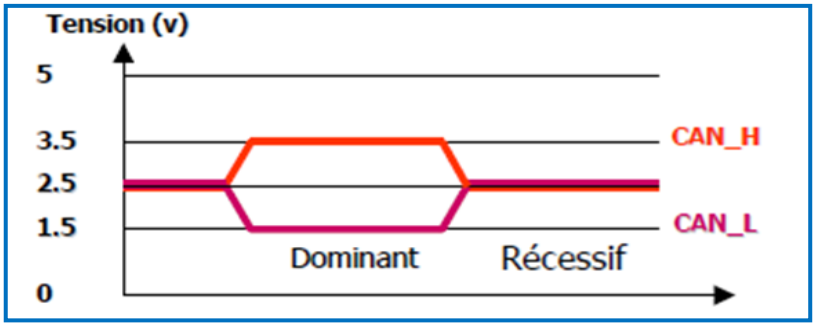


Figure 3.9 : représentation de signal dans l'oscilloscope (CAN 2.0 A).

- Adaptation de ligne nécessaire : RT (Résistance de Terminaison) : 120 Ohm

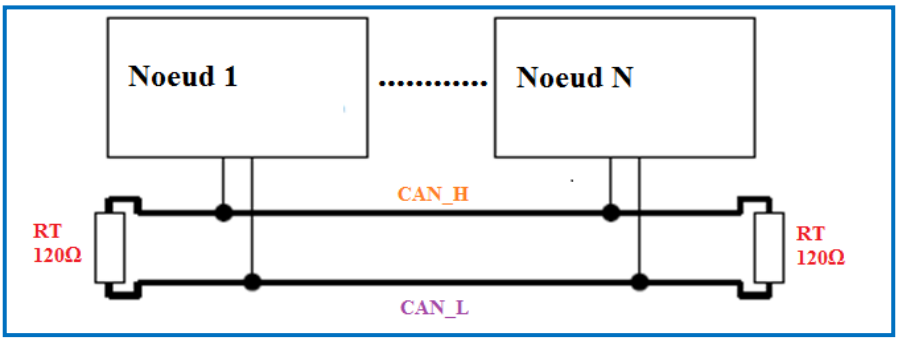


Figure 3.10 : résistance de terminaison

IV-5-4-Relation entre vitesse et distance :

Distance inversement proportionnelle à la vitesse : plus la vitesse augmente, plus la longueur max. du réseau diminue.

- ✓ **Cause:**

-le comportement temps-réel des phases d'arbitrage.

- le comportement temps-réel des phases de détection d'erreur.

- ✓ **Vitesse de communication :**

CAN 2.0B : jusqu'à 1Mb.s^{-1} (jusqu'à 125 kb.s^{-1} uniquement en **CAN 2.0A**)

- ✓ **Distances théoriques :**

- > 1km pour 10kb.s^{-1}

-< 40m pour 1Mb.s^{-1}

IV-6-Le protocole :

C'est la « langue » utilisée pour communiquer, c'est tout ce qui concerne le concept de communication du bus CAN est celui de la diffusion d'information (broadcast) :

Comme dans la plupart des protocoles, il est nécessaire d'utiliser un vocabulaire adapté à la situation. Il faut donc définir un certain nombre de termes et de règles de fonctionnement concernant le protocole CAN.

Notes : Bien qu'il existe de légères différences dans leurs significations, les termes nœud, station et participant sont utilisés de manière interchangeable pour désigner un ensemble connecté au réseau.

IV-6-1-Nœud d'envoi (station) :

ISO : sous-ensemble connecté à un réseau de communication, capable de communiquer sur le réseau selon un protocole de communication. Dans le cas du protocole CAN

Un nœud (station) générant un message est appelé « émetteur » de ce message,

Une station est appelée « expéditeur » jusqu'à ce que le bus soit inactif ou jusqu'à ce que l'unité ait perdu l'arbitrage.

IV-6-2-Nœud de réception (station) :

Contrairement au cas précédent, un nœud (station) est appelé « récepteur » d'un message s'il n'est pas l'« expéditeur » d'un message et si le bus n'est pas libre.

IV-6-3-Valeurs du bus :

Le bus peut avoir l'une des deux valeurs logiques complémentaires définies non pas comme « 0 » et « 1 », mais comme ce qu'on appelle des valeurs « dominantes » et « récessives ».

En cas de transmission simultanée d'un bit dominant et d'un bit récessif, le résultat de la valeur du bus sera « dominante ».

IV-6-4-Messages :

Chaque information est véhiculée sur le bus à l'aide d'un message (trame de bits) de format défini mais de longueur variable (et limitée). Dès que le bus est libre (bus idle), n'importe quel nœud relié au réseau peut émettre un nouveau message.

IV-6-5-Routage des informations :

Des nœuds peuvent être ajoutés au réseau sans qu'il n'y ait rien à modifier tant au niveau logiciel que matériel. Chaque message possède un identificateur (identifier) qui n'indique pas la destination du message mais la signification des données du message. Ainsi tous les nœuds reçoivent le message, et chacun est capable de savoir grâce au système de filtrage de message si ce dernier lui est destiné ou non. Chaque nœud peut également détecter des erreurs sur un message qui ne lui est pas destiné et en informer les autres nœuds.

IV-6-6-Débit binaire du bus :

Selon ISO, le débit binaire du bus est le nombre de bits par unité de temps pendant la transmission, le débit binaire CAN (souvent appelé à tort « vitesse ») peut différer d'un système à l'autre, mais il doit être uniforme et constant pour tout réseau. Vous devriez donc faire attention à la compatibilité inter-réseaux, car plusieurs réseaux coexistent souvent exister au sein d'un même système unique, ce qui oblige à créer des passerelles entre eux.

IV-6-7-Demande d'une trame de données :

Un nœud peut demander à un autre nœud d'envoyer une trame de données, et pour cela il envoie lui-même une trame de requête. La trame de données correspondant à la trame de requête initiale possède le même identificateur.

IV-6-8-Fonctionnement multimètre :

Lorsque le bus est libre, chaque nœud peut décider d'envoyer un message. Seul le message de plus haute priorité prend possession du bus.

IV-7- La Priorités et Arbitrage :

Les identificateurs de chaque message permettent de définir quel message est prioritaire sur les autres.

Le problème de l'arbitrage résulte du fonctionnement multimètre. Si deux nœuds ou plus tentent d'émettre un message sur un bus libre il faut régler les conflits d'accès. On effectue alors un arbitrage bit à bit ; Seule la trame prioritaire est émise mais la 2ème n’est pas détruite, elle sera

remise dès que le bus sera libre (non destructif) tout au long du contenu de l'identificateur. Ce mécanisme garantit qu'il n'y aura ni perte de temps, ni perte d'informations. Dans le cas de deux identificateurs identiques, la trame de données gagne le bus. Lorsqu'un bit récessif est envoyé et qu'un bit dominant est observé sur le bus, l'unité considérée perd l'arbitrage, doit se taire et ne plus envoyer aucun bit. L'arbitrage est qualifié de CSMA/CA (*Carrier Sense Multiple Access - Collision Avoidance*).

IV-7-1- Rôle des bits dans le champ d'arbitrage:

- **Le bit SOF** (début de trame de données) : C'est dominant il signale à toutes les stations le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos.
  Toutes les stations doivent se synchroniser sur le front avant la transition du bit de départ.
- **Identificateur :** La longueur de l'identificateur est de 11 bits, les bits sont transmis dans l'ordre de ID_10 à ID_0 (le moins significatif est ID_0). Par ailleurs les 7 bits les plus significatifs (d'ID_10 à ID_4) ne doivent pas être tous récessifs.
  ID = 1111111XXXX (X valeur indéterminée), c'est-à-dire un nombre maximal d'identificateurs de : (211 - 24) = 2048 - 16 = 2032 combinaisons.
- **Le bit RTR :** Lors d'une trame de donnée le bit de remote transmission requeté (RTR) doit être dominant.

IV-8-Caractéristiques de protocole BUS CAN :

IV-8-1-Le bit CAN :

Pour éviter de devenir empêtré dans les rouages de la théorie, en concept "prédéfinissons". Il y a des "bits" et des "bits" ! Pour définir un peu, il faut définir sa durée et la manière dont son contenu (son valeur) est codé dans cette durée. Il existe de nombreuses manières d'encoder un bit ;

IV-8-1-1- Le codage NRZ :

bits dominants et récessifs :La succession de bits transitant sur le bus est codé avec la méthode du NRZ (*Non Return To Zero*).
Pendant la durée totale du bit, le niveau de tension de la ligne est maintenu, c'est à dire que pendant toute la durée durant laquelle un bit est généré, sa valeur reste constante qu'elle soit dominante ou récessive.

IV-8-1-2- Le bit Stuffing :

Une des caractéristiques du codage NRZ est que le niveau du bit est maintenu pendant toute sa durée. Cela pose des problèmes de fiabilité si un grand nombre de bits identiques se succèdent. La technique du Bit Stuffing impose au transmetteur d'ajouter automatiquement un bit de valeur opposée lorsqu'il détecte 5 bits consécutifs dans les valeurs à transmettre.

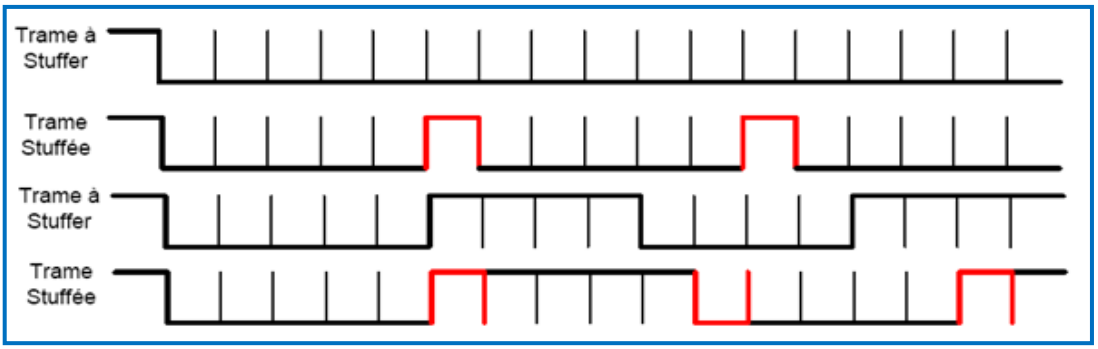


Figure 3.11 : Schéma de bit Stuffing .

IV-8-2-Identifiant :

L'identifiant n'indique en aucune manière la destination du message mais décrit (en sens de « donner une idée générale ou précise ») du contenu et du sens des données. Cette implique que chaque nœud est capable de décider si oui ou non le message transporté sur le bus lui est pertinent. Cette fonction est (ou sera, ou doit être) exécutée par un message électronique dispositif de filtrage. En d'autres termes, les messages sont envoyés dans des ténèbres impénétrables, au cas où quelqu'un est intéressé.

IV-8-3-Sécurité de transmission ave le contrôle CRC :

Dans le but d'obtenir la plus grande sécurité lors de transferts sur le bus, des dispositifs de signalisation, de détection d'erreurs, et d'autotests ont été implémentés sur chaque nœud d'un réseau CAN. On dispose ainsi d'un monitoring bus (vérification du bit émis sur le bus), d'un CRC (Cyclic Redundancy Check), d'une procédure de contrôle de l'architecture du message, d'une méthode de Bit-Stuffing. On détecte alors toutes les erreurs globales, toutes les erreurs locales au niveau des émetteurs, jusqu'à 5 erreurs aléatoires réparties dans un message. La probabilité totale résiduelle de messages entachés d'erreurs est inférieure à$4.7*10^{-11}$.

IV-9-Schéma de principe d'un module émetteur/récepteur :

Le Transciever possède un élément de détection des anomalies des tensions sur le bus. Cette configuration permettant de diagnostiquer les défauts et ainsi de continuer la transmission (ou la réception). De par la présence de ces éléments intégrés de diagnostic, ce réseau tolère certains défauts, d'où le nom de *CAN fault tolerant* attribué au *CAN low speed*.

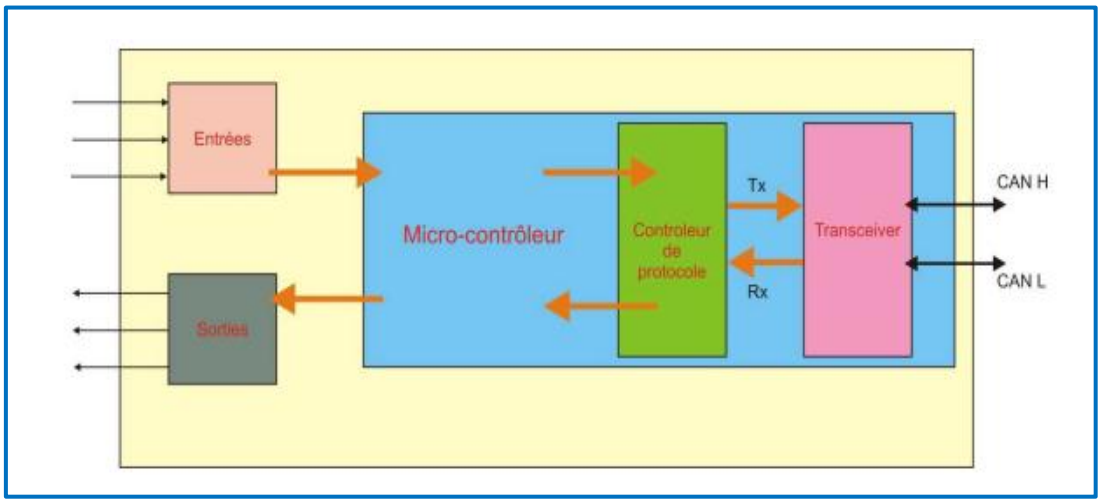


Figure 3.12 : Schéma de principe d'un module émetteur/récepteur.

IV-10-Transfert de messages

Le transfert des messages s'effectue et contrôlé à l'aide de quatre types de trames spécifique et un intervalle de temps les séparant

- Trames de données (*dataframe*)
- Trames de requête (*Remoteframe*)
- Trames d'erreur (*Errorframe*)
- Trames de surcharge (*Overloadframe*)
- Inter trames (*Interframes*)

IV-10-1- Trame de donnée et trame de requête:

- Une trame de données est une trame qui transporte, comme son nom l'indique des données la plus utilisée est de type standard CAN 2.0A
- Une trame de requête est émise par un nœud désirant recevoir une trame de données (l'identificateur est le même pour les deux trames dans ce cas), elle comporte les mêmes champs comme une trame de requête mais avec le bit RTR=1
- La trame de donnée est prioritaire sur la trame de requête de même identificateur

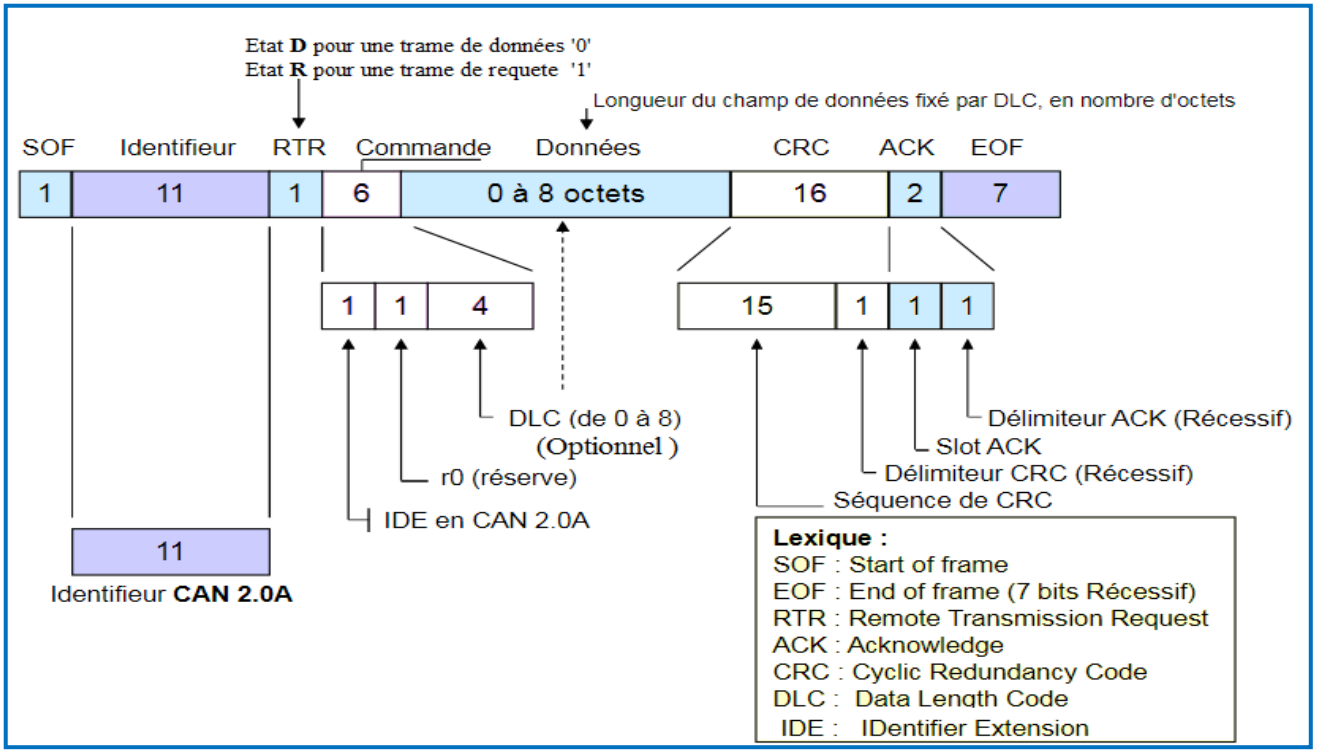


Figure 3.13 : la structure et la forme de trame donnée et trame de requête

IV-10-2- La Trame d'erreur

Pour différentes raisons, comme l'existence de fortes perturbations ou de pertes importantes lors de la transmission, le protocole CAN dispose d'un système de gestion des erreurs locales.

Le principe du bit Stuffing vu précédemment permet de localiser une erreur et un nœud qui détecte ce type d'erreur transmettra aux autres nœuds un message dit « Error Flag » contenant six bits de même polarité.

Après avoir transmis le message *Error Flag*, le nœud essaiera à nouveau de transmettre le message, et si aucun message de priorité supérieure ne prend la main sur le réseau ce nouveau message est transmis 23 bits au plus après.

Les bits formant l'Erreur Flag sont dominants et écrasent donc les données contenues dans la Data Frame. Ils provoquent la retransmission de cette dernière. Dans le cas d'erreurs successives, il y aura superposition de (Erreur Flags.)

Les 8 bits de l'Errer Délimiter donnent l'autorisation aux nœuds du réseau de reprendre leurs communications.

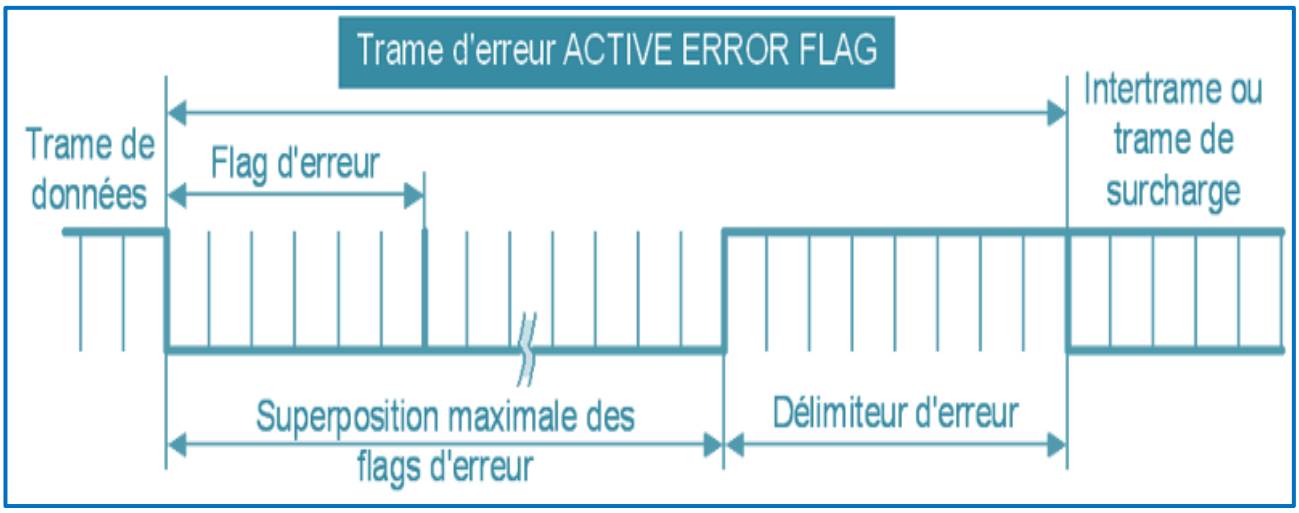


Figure 3.14 : structure de la trame d'erreur

IV-10-3- Trames de surcharge :

Utilisées pour demander un intervalle de temps supplémentaire entre la précédente et trames de données suivantes ou trames distantes.

- La trame ne comprend que 2 champs....
- Le champ «flag» de surcharge (OLF)
- Le champ de surcharge est composé de 6 bits dominants
- Le champ délimiteur - 7 bits récessifs
- Le champ de surcharge transgresse la loi du bit Stuffing

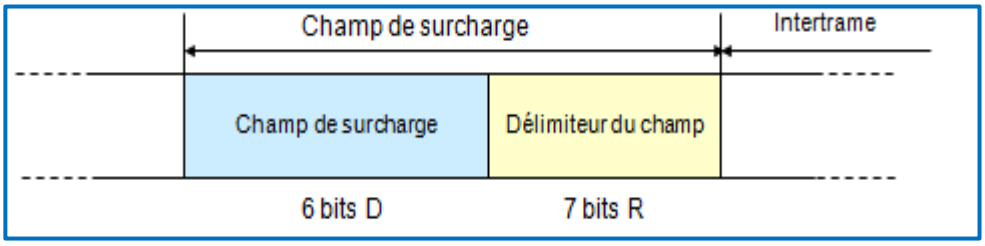


Figure 3.15 : structure de la trame surcharge

IV-10-4- Inter-trames :

C'est les suites des bits récessifs qui séparent (dans le temps) les trames de données et les trames distantes des précédentes trames par un espace inter-trames.

- Les trames Données et Requête sont précédées par une période d'inter-trame
- Les trames Erreur et Surcharge ne sont pas précédées par une période d'inter-trame
- La période est composée de 2 ou 3 champs
- Champs: Intermission, Bus Idle et éventuellement Suspend la transmission

Toutes ces trames transportent des informations véhiculées sur le bus au niveau le plus bas de la physique couche à l'aide de « bits ». Afin de s'assurer que certaines phases des différentes trames et le traitement des erreurs sont mieux compris, il sera utile d'examiner les particularités des bits et leur procédure de codage dans le protocole CAN.

IV-11-Les principes du multiplexage :

C'est tout ce qui concerne l'acheminement des trames, Les trames sont distribuées sur le bus

les « récepteurs » consultent l'identité de la trame (champ d'identification de la trame) et seuls ceux qui sont concernés par la trame, utilisent ses informations.

Les échanges de trame, donc de bits, doivent se faire à un rythme bien précis. Pour ce faire chacun des boîtiers possèdent une horloge interne (quartz),les boîtiers récepteurs doivent caler leur horloge sur celle de l'émetteur, Il se peut que 2 boîtiers veuillent émettre une trame en même temps sur le bus ; une trame est forcément prioritaire sur l'autre, c'est le principe d'arbitrage.

IV-12- La méthode d'arbitrage :

Le champ «Identifieur» sert pour l'arbitrage entre trames si deux nœuds ou plus tentent d'émettre un message sur un bus libre il faut régler les conflits d'accès. On effectue alors un arbitrage bit à bit (non destructif) tout au long du contenu de l'identificateur, Il est à la base du principe d'arbitration. L'arbitrage concerne aussi le bit RTR

Ce mécanisme garantit qu'il n'y aura ni perte de temps, ni perte d'informations.

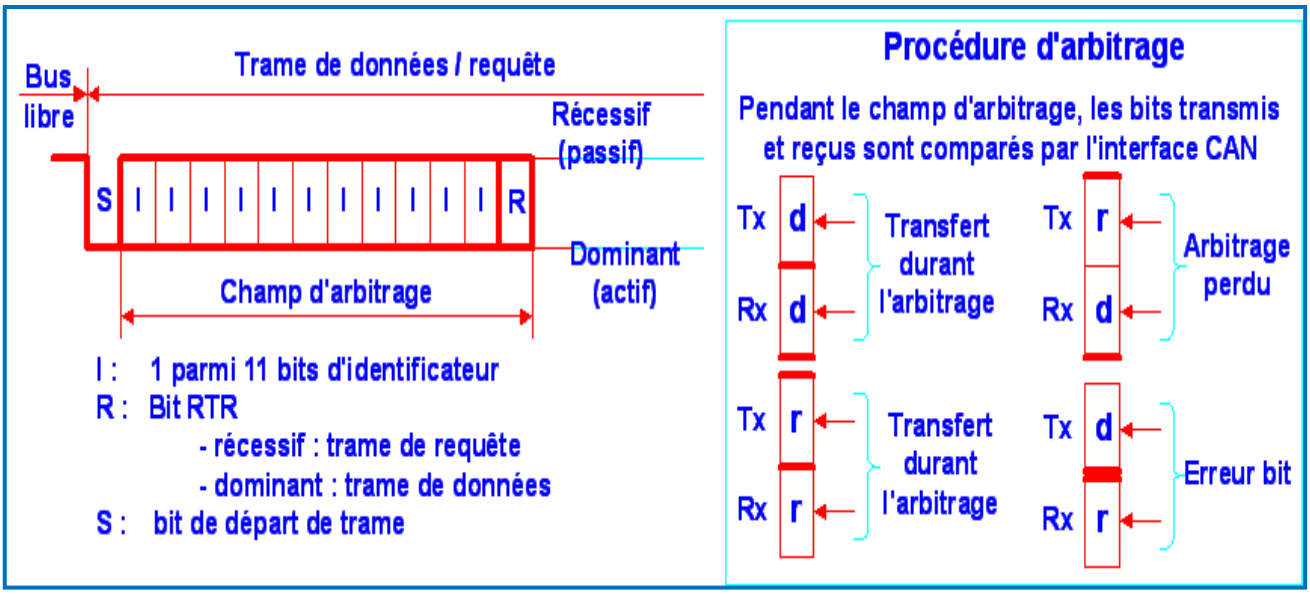


Figure 3.16 : Principe d'arbitrage bus CAN

IV-13-Traitement des erreurs :

Il est utile de connaître à tout moment le type des erreurs, la fréquence des erreurs,le niveau de gêne provoqué par les erreurs, le retour à une activité bus sans erreurs pour ça il nous faut un mécanisme de détection, etun système tolérance aux fautes.

IV-13-1-Mécanismes de détection des erreurs :

Le CAN implémente cinq mécanismes de détection des erreursFigure21**,**

2 au niveau bits (le ”bit monitoring” et le ”bit stuffing”) et 3 au niveau messages (vérification du CRC, de la forme des trames et de l’acquittement).

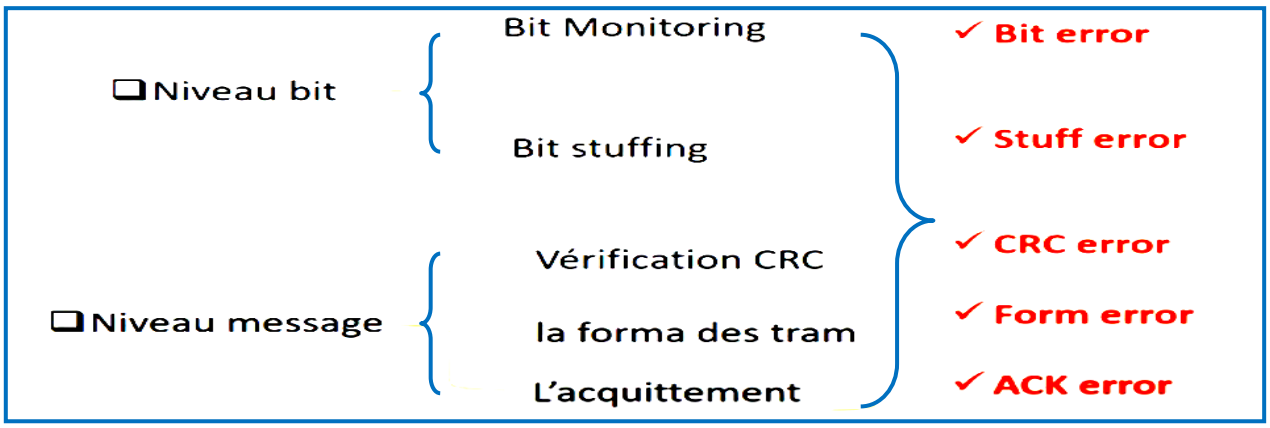


Figure 3.17 :les types des erreurs détectées par le Protocol CAN

Ces cinq types d’erreurs différents qui peuvent être détectée par un nœud sont:

- **Bit erreur:** Un nœud envoyant un bit sur le bus regarde aussi en même temps les bits qu’il reçoit (Bit monitoring). Il considère comme une erreur de bit lorsque le bit envoyé est différent du bit reçu, à l’exception de l’envoi d’un bit récessif durant l’arbitrage (cas de la perte d’arbitrage) ou pendant l’ACK Slot (trame acquittée).
- **Stuff erreur:** Le nœud détecte une erreur de Stuffing lorsqu’il reçoit 6 bits consécutifs de même valeur dans une partie d’un message qui devrait être codé avec la méthode du bit Stuffing.
- **CRC erreur:** Une erreur de CRC est d´détectée lorsque le CRC calculé par un récepteur est différent de la valeur du CRC contenu dans la trame.
- **Forme erreur:** Une ”Forme erreur” est détectée lorsqu’un bit qui devrait être à une certaine valeur est à une valeur différente (un délimiteur par exemple).
- **ACK erreur :** Le transmetteur détecte une erreur d’acquittement lorsqu’il ne reçoit pas de bit dominant pendant l’ACK Slot.

La détection de ses 5 types d'erreurs se varier selon le rôle de nœud (émission / réception)

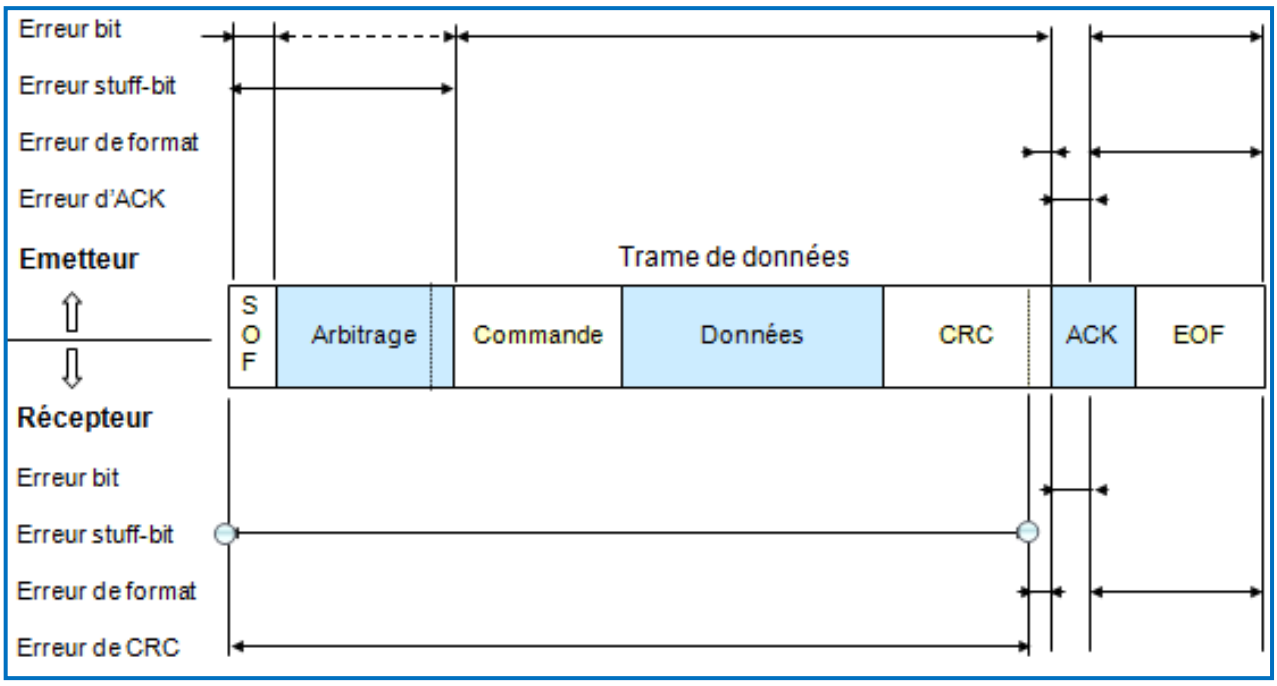


Figure 3.18 : les types des erreurs sur certains champs

IV-14-Conclusion

Afin d'avoir un système fiable le plus possible et tolérant aux faute en Modélisant une conceptualisation et amélioration des spécifications produisant un modèle du matériel et du logiciel capable de le rendre sûr et plus robuste.

Apres le déploiement du protocole CAN dans divers domaine des études se fait dans le but de réaliser des prototypes est de pouvoir simuler le fonctionnement d'un système embarqué en temps réel.

Chapitre 4 -

MODELE PROPOSE

1-Introduction

Après avoir étudier les systèmes embarque en temps real qui utilise Bus CAN comme protocole de communication et Cloud Computing, le travail consiste à la mise en œuvre d'une solution tolérance aux fautes et tolérance aux pannes qui se charge principalement de garder le réseau fonctionnel sans pannes et sans fautes; Sachant que après l'évaluation d'un système mono bus nous allons essayer d'améliorer la performances, la fiabilité et la disponibilité de la transmission des messages entre les nœuds en temps considéré et de sauvegarder l'état des nœuds périodiquement dans un Cloud en utilisant le protocole TCP/IP à travers un réseau internet.

Ce pendant notre proposition, qui sera décrite tout au long de ce chapitre dans le but de faire déployer une simulation d'un modèle de système qui sera basé sur le protocole Bus CAN et qui consiste à connecter *M* nombre de nœuds entre eux avec N nombre de bus.

En fin on va expérimenter si les mécanismes de la tolérance aux fautes et pannes fonctionnent correctement ou non.

2-Description de l'approche proposée :

Dans cette section, nous présentons notre approche proposée pour la mise en œuvre d'un modèle d'un système embarque en temps real qui utilise le Protocol CAN comme méthode de transmission et de traitement des messages. Dans le système mono bus, si ce bus cesse de fonctionner alors le système arrête d'une façon extrêmement critique ; le cas où on a plusieurs nœuds soit en concurrence entre eux pour gagner la priorité d'envois, ici les nœuds dominés faiblir leur chance d'émission des requêtes et aussi leur chance de réceptionner des réponses dans le temps désiré; si le bus avait un bruit de signal dans une période fréquente, alors les trames transmises sont soumis à des erreurs aléatoires.

La proposition pour tolérer ces fautes et pannes est d'augmenter le nombre des bus de transmission pour éviter l'arrêt du système, affronter le bruit et de minimiser le temps de latence pour les messages importants en engendrant la gestion des priorités de choisir un bus pour la transmission avec le principe de l'arbitrage déjà étudier et la gestion de tampon de message.

Durant le déroulement de notre système, on détermine une période de sauvegarde de l'état de nœuds dans un Cloud pour la traçabilité du système et l'accessibilité facile aux résultats ; la surveillance et l'analyse de ces résultats nous aide à prendre la décision adéquate.

2-1-Conception de l'approche Proposée :

Nous présentons les outils utilisés pour la mise en œuvre de Modèle de système aux N bus et M nœuds, nous commençons tout d'abord par montrer les caractéristiques principales:

2-1-1-Bus : un canal de deux paires torsadées

2-1-2-Nœud : un composant qui se caractérisé par :

- **N° :** c'est le numéro de nœud.
- **IDs** : chaque nœud comporte 03 identificateurs des trames qu'ils possèdent :
  - ✓ Cycle : temps écoulé pour génère une requête.
  - ✓ TTL : (time to live) temps d'attente de réponse.
  - ✓ TYPE : trame de requête (R), trame de donnée (D).
- **Lampes témoins :** ce sont 04 indicateurs qui représentent :
  - ✓ Emission **:** représente l'état d'émission générale de nœud.
  - ✓ Réception **:** représente l'état de réception générale de nœud.
  - ✓ Fonctionnement interne: représente le fonctionnement de contrôleur de nœud.
  - ✓ Gestion des Erreur: Représente la gestion d'une erreur de transmission par le nœud.
- **Transciever :** il gère la transmission (émission, réception) des messages stocker dans la mémoire tampon au niveau de canaux disponible; il est constitué de :
  - ✓ **Canaux** : Points d'interconnexion avec les N bus.
  - ✓ **Tampon** : est une zone mémoire stocke le message généré par le contrôleur de nœud pour l'émission et la réception des messages.
  - ✓ **Vecteur de priorité :** contient la priorité d'émission de chaque point d'interconnexion, tel que la priorité très élevé soit A et la plus basse soit D ; ce caractère se changent au cours de temps.

2-1-3-Nœud Principal :

C'est un nœud intermédiaire qui se charge de sauvegarder périodiquement l'état (information interne) des autres nœuds dans le Cloud via l'internet; le nœud principal est relier avec tous les autres nœuds avec un réseau TCP/IP en utilisant un Switch.

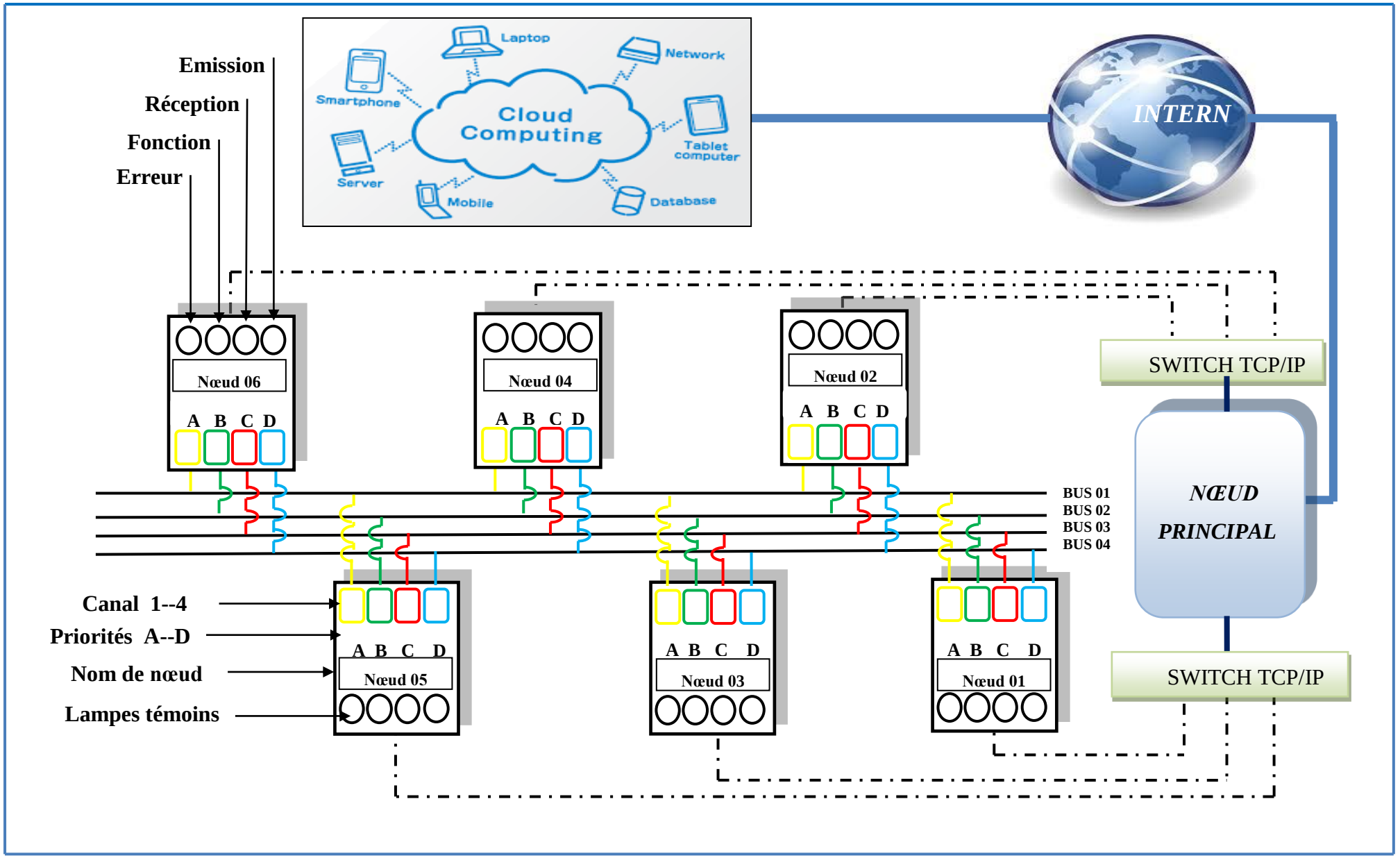


Figure 4.1 : Schéma de la conception propose

2-2-Les contraintes de l'approche et règles de fonctionnement :

Pour réaliser cette approche on a posé quelques contraintes comme suit :

- Le nombre des bus minimale est 1.
- Existe un seul nœud principal.
- Tous les autres nœuds sont similaires et équiprobables sauf le nœud principal.
- Chaque ID représente une trame de données et une trame de requête.
- Les IDs dans un Nœud ne sont pas identiques.
- Chaque nœud possède un seul ID pour la trame de données et peut avoir 01 ou plusieurs ID de trame de requête.
- Chaque ID qui représente une trame de données n'existe que dans un seul nœud.
- Chaque ID qui représente une trame de requête peut exister dans un ou plusieurs nœuds.
- Le Cycle de ID de trame de données est toujours 0.
- Le Cycle de ID de trame de requête est supérieur de 0.
- Le Cycle de ID de trame de requête est se varié d'un nœud à l'autre.
- Le TTL d'un ID est toujours identique dans les nœuds qui possède cet ID.
- Le nombre des canaux de Transciever est égale au nombre des bus.
- Chaque canal de Transciever est interconnecté avec un seul bus.
- Chaque nœud possède un vecteur de priorité de Choix de bus, la taille de ce vecteur est N (N : nombre de bus).
- Chaque nœud possède une seule zone tampon.
- Les collisions de transmission sont acceptées.
- Les collisions sont détectées par les nœuds.
- L'arbitrage est la stratégie qui gère les collisions.
- La trame de données se génère seulement après l'arrivage d'une trame de requête.
- La trame de requête se génère seulement après l'écoulement de cycle.
- Apres avoir générer une trame, le contrôleur de nœud déplace cette trame dans le tampon.

- Le Transciever envoi la trame depuis le tampon s'il existe un bus libre en respect la priorité du canal.
- Lorsque l'achèvement d'émission d'une trame à travers un canal, la priorité de ce dernier est devenu 'A' et les priorités de reste des canaux se décalent ('A' devient 'B','B' devient 'C', 'C' devient 'D').
- Après l'écoulement de CàR et la trame existe dans le tampon, le Transciever retire cette trame depuis le tampon.
- Après la réception d'une trame de requête, le nœud doit générer une trame de données s'il possède le ID de cette requête comme un ID de données.
- Durant la réception d'une trame la vérification des erreurs se fait momentanément, s'il existe une erreur il faut la signaler à l'émetteur.
- Apres la réception d'une trame d'erreur par l'émetteur à travers un canal, la priorité de ce dernier est devenu 'D' et les priorités de reste des canaux se décalent ('B' devient 'A','C' devient 'B', 'D' devient 'C') et puis renvoie la trame en respect la nouvelle priorité des canaux.

3- Simulation :

La simulation est l'un des outils d'aide à la décision les plus efficaces elle serve à comprendre le comportement des protocoles et d'améliorer leurs performances, d'une part elle permette à d'autres personnes de vérifier l'exactitude et la validité de l'analyse et du modèle considéré.

Mais d'autre part pendant la simulation, l'environnement et les protocoles considérés peuvent devenir plus complexes et par conséquent, plus réalistes.

Cependant, Notre simulation permet d'illustrer la méthode de la transmission des messages et leur déplacement entre les nœuds à travers 04 bus.

L'interface graphique se fait pour visualiser le déroulement de la simulation, et évaluer le taux de performance; Nous fournissons des informations plus détaillées sur le modèle proposé dans la suite.

3-1-Environnement de développement :

Dans cette partie nous allons citer l'environnement logiciel (Software) et matériel (Hardware) utilisés.

3-1-1- Langage de programmation :

Nous avons conçu notre application de simulation avec le langage de programmation Object Pascal sous RAD Delphi. Delphi est un environnement de développement de type RAD (Rapide Application Développent) ; il permet de réaliser rapidement et simplement des applications avec des interfaces graphiques flexibles sous Microsoft Windows, Apple Mac Os, Apple IOS et Androïde. Nous avons opté la version 10.3(Embarcadero) de Delphi car elle fournit un ensemble d'outils nécessaires pour déployer, développer et tester des applications, notamment une importante bibliothèque de composants réutilisables, une suite d'outils de conception, de modèles d'applications, de fiches et d'experts de programmation que les versions précédentes du Delphi ne possédaient pas.

3-1-2-Les Avantages de Delphi :

Delphi apporte une grande souplesse au développeur ; lorsque Delphi génère un fichier (.exe), il s'agit d'un vrai exécutable. Aucun autre fichier n'est nécessaire pour l'exécution. Vous obtenez donc une application plus propre, et plus facile à distribuer et à maintenir ; vous n'avez à distribuer qu'un seul fichier, indépendant de DLL ou d'autres fichiers. Pour les entreprises soucieuses de l'établissement de normes et de standards, Delphi est également utile; supposons que vous écriviez des applications Delphi dans une entreprise de 5000 employés, chaque fois que vous devez déployer une nouvelle application vous devez envoyer un fichier de 1MOct à tous les utilisateurs, ceci peut vite encombrer le réseau mais heureusement Delphi vous permet de regrouper les composants standards dans un paquet ; Une fois ce paquet installé sur les machines, vous pouvez l'utiliser pour toutes les nouvelles applications que vous déploierez. Dès lors, il vous suffit d'envoyer un fichier exécutable de 200 ko au lieu de 1 Mo. Cette technique est l'une des nouvelles fonctionnalités de Delphi 10.3 et permet de minimiser la taille des applicatifs transmis à chaque machine dès lors qu'un paquet standard a été installé.

Delphi vous offre donc un compilateur optimisé qui vous donnera une application rapide sans qu'il soit nécessaire de fournir plus d'efforts pour optimiser le programme qu'il n'en avait fallu pour l'écrire.

3-1-3- Environnement matériel :

- Dans notre projet nous avons utilisé un LapTopHP630 caractérisé par :
- Micro-Processeur : Intel(R) Core (TM) i3 CPU M 380 @ 2.53GHz.
- Mémoire : 4 Go DDR3.

- Carte Graphique : Intel HD Graphique 1536 Mo.
- Disque Dure : SATA Samsung500 Go.
- Ecran : Intégré 15,6 pouces (1336 x 768).
- Système d'exploitation : Windows 10 64 bits.

3-2- Réalisation de l'application :

Notre application de simulation repose sur 06 parties capitales dont l'interface Principale (Figure 4.2**)** repartie comme suit :

3-2-1-Menu : un menu simple constitué de

3-2-1-1-Fichier : elle est composée de :

a- **Ouvrir un modèle**: pour objectif d'ouvrir un fichier d'un modèle de simulation.

b- **Enregistrer le modèle**: après tester un modèle de simulation et le modifier on peut le sauvegarder.

c- **Quitter** : Fermer l'application.

3-2-1-2-Aides : fenêtre d'explication sur l'utilisation de l'application.

3-2-1-3-Apropos : information sur les concepteurs de l'application.

3-2-2-Paramètre : d'après cette partie la création d'un nouveau modèle de simulation se fait à partir de la détermination des paramètres suivants :

3-2-2-1-Modèle prédéfinie : c'est un modèle de simulation de 08 nœuds intégré dans l'application.

3-2-2-2-Nombre des Nœuds : désigne le nombre des nœuds et au même temps le nombre de identificateurs de message (IDs).

3-2-2-3-Sauvegarde Cloud : choisir pour enregistrer l'état des nœuds dans la période déterminée ce dessous ce forme d'un fichier texte dans le répertoire racine de l'application.

3-2-2-4 Période : désigne le nombre des pulses périodique pour sauvegarder l'état des nœuds dans Cloud.

3-2-2-5 Créer le Mappe / Effacer : bouton pour la création graphique de notre modèle de simulation choisi ou l'initialiser.

3-2-3-Contrôler : c'est un ensemble d'outils pour contrôler notre simulation tel que :

3-2-3-1 Lancement /Arrêt : bouton pour démarrer ou arrêter la simulation.

3-2-3-2 Pause / Reprendre : bouton pour suspendre ou reprendre la simulation.

3-2-3-3 Fréquence : désigne le durée d'une pulse.

3-2-3-4 Générer les Erreurs : c'est pour objectif d'insertion quelques erreurs aléatoires durant la simulation; il existe des erreurs permanentes et non permanentes.

3-2-4-Fenêtres : c'est un ensemble de boutons qui permet d'afficher et de cacher les différentes fenêtres auxiliaire.

3-2-5- Barre d'état : montre l'instance actuelle (Pulse) et la vitesse d'émission/Réception.

3-2-6-Espace de travail : c'est un espace dans lequel les fenêtres auxiliaires apparaitre.

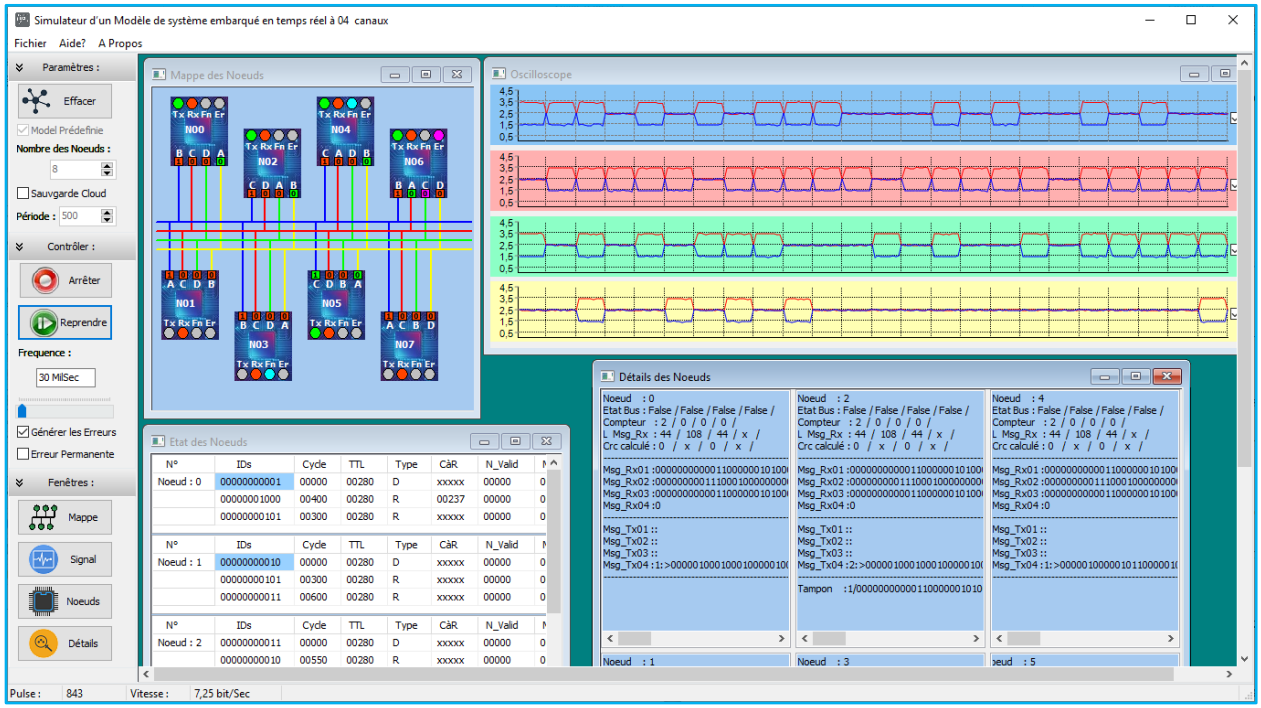


Figure 4.2 :L'interface Principale

4-Espace de travail description détailler :

4-1-Mappe des Nœuds :

Ce schéma illustre généralement l'état global de fonctionnement de notre modèle de simulation (figure 4.3) ; Il contient des nœuds interconnecte entre eux avec 04 canaux à travers des bus colorés en Bleu, Rouge, Vert et jaune; chaque nœud possède un ensemble des éléments qui indique son état de fonctionnement à l'instance donnée (Pulse), ce nœud est caractérisé par :

4-1-1-Nom de nœud : C'est l'identificateur de ce nœud ;

4-1-2-Panneau d'état global : Est composé de 04 LEDs (lampes d'indication) :

a- LED Tx : Indicateur d'émission détermine l'état d'émission générale de nœud; cet indicateur possède 02 couleurs possibles :

-**Vert** : Lorsqu'on a un ou plusieurs canaux en état d'émission cet indicateur s'allume en couleur Vert.

-**Gris** : Cet indicateur ne s'allume pas dans le cas où le nœud n'est pas actif ou il n'y a pas aux moins un canal en état d'émission.

b- LED Rx : Indicateur de réception détermine l'état de réception générale de nœud; cet indicateur possède 02 colleurs possibles :

-**Orange :** Lorsqu'on a un ou plusieurs canaux en état de réception cet indicateur s'allume en couleur Orange.

-Gris : Cet indicateur ne s'allume pas dans le cas où le nœud n'est pas actif ou il n'y a pas aux moins un canal en état d'émission.

c- LED Fn : Représente le fonctionnement interne de contrôleur de nœud lorsque il a reçu une trame de requête il doit génère une trame de donnée, ou bien lorsque il a reçu une trame de donnée il exploite la valeur de donnée ; cet indicateur possède 02 colleurs possibles :

-Cyan : Cet indicateur s'allume en couleur Cyan lorsque le nœud a reçu une trame de requête ou bien de donnée identifier par lui.

-Gris : Cet indicateur ne s'allume pas dans le cas où le nœud n'est pas actif ou il n'y a pas une réception de la trame identifiée par ce nœud.

d-LED Er : Représente la détection d'une erreur de transmission par le nœud ;cet indicateur possède 02 colleurs possibles :

-**Magenta:** Après le traitement de la trame en cours de réception, cet indicateur s'allume en couleur rouge en cas de détection d'une erreur.

-Gris : Cet indicateur ne s'allume pas dans le cas où le nœud n'est pas actif ou il n'y a pas une erreur détecte.

4-1-3- Panneau de Transciever :

Il joue le rôle de l'émetteur-récepteur des trames à travers chaque canal, il est composé de 04 canaux numérotés de 01 à 04 et de gauche à droite ; chaque canal représentée par un indicateur quipossède04 colleurs possibles :

-Vert : L'indicateur s'allume en couleur Vert lorsque le nœud est en train d'envoie une trame à partir de ce canal.

-Orange : L'indicateur s'allume en couleur Orange lorsque le nœud est en train de recevoir une trame à partir de ce canal.

-Magenta: L'indicateur s'allume en couleur Magenta en cas de détection d'une erreur de réception à partir de ce canal.

-Gris : Cet indicateur ne s'allume pas dans le cas où le nœud n'est pas actif ou il n'y a pas une réception dans ce canal.

4-1-4-Vecteur de priorité :

Détermine les priorités de choix de canal en cas d'émission tel que cette priorité représentée par un alphabet de 'A' à 'D' situé aux dessus de l'indicateur du canal ; la priorité la plus élevée est 'A' et la plus basse est 'D'.

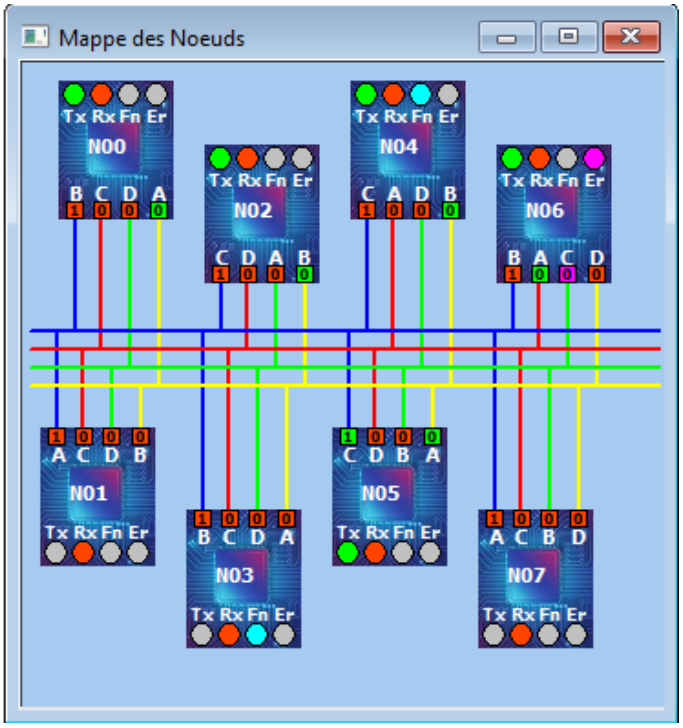


Figure 4.3 : Mappe Des Nœuds.

4-2-Oscilloscope :

La représentation de signal électrique transmis par le canal à travers son bus est dessiné se forme de 02 courbes colorées en rouge et bleu, la courbe rouge représente *Can_Low* et la courbe bleu représente *Can_Hight* (figure 4.4); cas à coché de chaque bus détermine la disponibilité de bus ce signal est formé de 02 valeur binaire tel que :

*Le bit 0 ou bit dominant est symbolisé par :

-Amplitude de *Can_Hight* (V_Can_H) = 3.5v.

-Amplitude de *Can_Low* (V_Can_L) = 1.5v.

*Le bit 1 ou bit récessif est symbolisé par :

-Amplitude de *Can_Hight* (V_Can_H) = 2.5v.

-Amplitude de *Can_Low* (V_Can_L) = 2.5v.

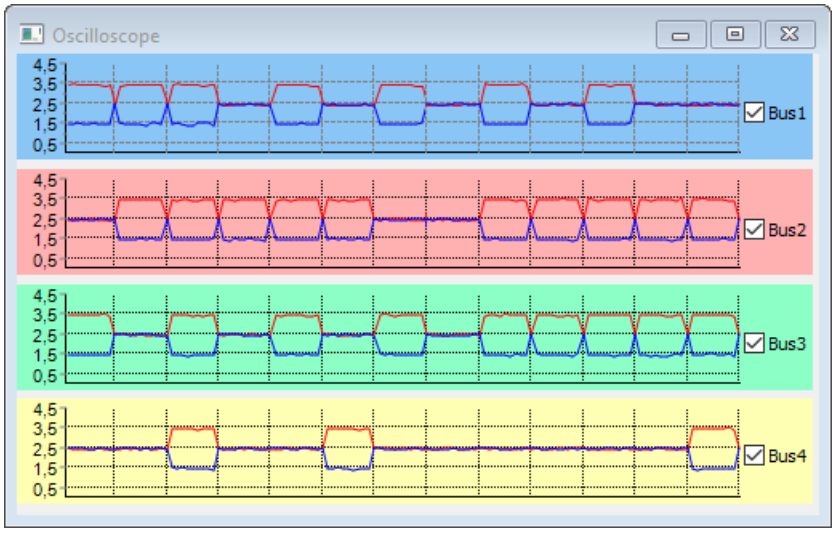


Figure 4.4 : Oscilloscope

4-3-Etat des Nœuds :

Les informations disponibles sur un nœud sont intégrées dans la mémoire de contrôleur ; ces informations représentent l'état global de chaque nœud dans le contexte quantitatif ; pour avoir dévoilé ces informations il faut les représenter sur un tableau structuré (figure 4.5); ils sont repartis sur 02 catégories comme suit :

4-3-1-Information Statique : Ce sont les information intégrés par l'usine de fabrication et on ne peut pas les changer.

a- **N° :** C'est le numéro de nœud.

b- **IDS :** Ce sont les identificateurs des messages reconnus par ce nœud, ces identificateurs sont divisés en 02 types (ID donnée, ID requête).

c- **Cycle :** C'est le temps écoulé pour générer une trame pour les ID requête, par contre les ID données sont nulle puisque la trame de données se généré lorsque le nœud reçoit une trame de requête avec un ID identifier par le nœud.

d- **TTL (Time to live) :** Désigne le temps pendant lequel une trame est censée exister dans le tampon avant d'être rejeté par le transciever.

e- **Type ID :** Se varier entre 02 types, type 'D' signifier les ID données et 'R' signifier ID requête.

4-3-2-Information dynamique : Ce sont les information qui se changer avec le déroulement de fonctionnement de nœud.

a- **CàR :** Signifier Compte à Rebours, indique le temps restant entre le moment de l'ajout du trame dans le tampon et le TTL (CàR max = TTL, CàR Min = 0) ;

b- **N_Valid :** Un compteur qui montre le nombre des trames non transmises durant leur TTL concernant les ID données et le nombre des trames n'a aucune réponse concernant les ID requêtes, après les supprimées depuis le tampon.

c- **Msg_E :** Un compteur qui montre le nombre des trames envoyées avec un ID par le nœud.

d- **Msg_R :** Un compteur qui montre le nombre des trames reçus et identifier par le nœud.

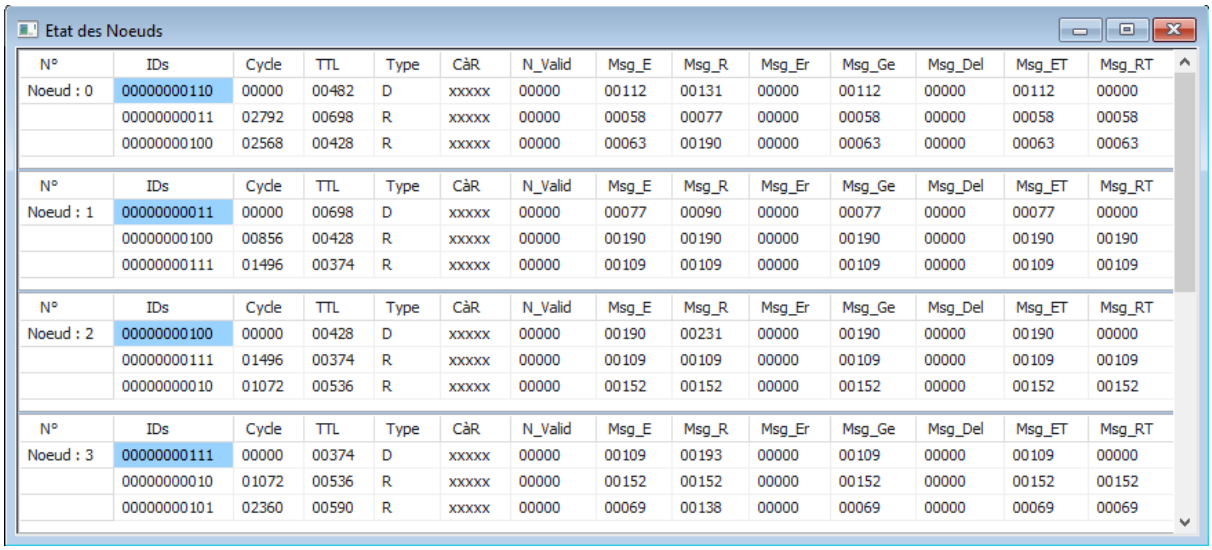
e- **Msg_Er :** Un compteur qui montre le nombre des trames ayant une erreur de transmission.

f- **Msg_Ge :** Un compteur qui montre le nombre des trames générées de chaque ID par ce nœud.

g- **Msg _DL :** Un compteur qui montre le nombre des trames non envoyées et puis supprimées depuis le tampon.

h- **Msg_ET :** Un compteur qui montre le nombre des trames envoyé avant l'écoulement de leur CàR.

i- **Msg_RT :** Un compteur qui montre le nombre des trames réceptionné avant l'écoulement de leur CàR.



N°	IDs	Cycle	TTL	Type	CàR	N_Valid	Msg_E	Msg_R	Msg_Er	Msg_Ge	Msg_Del	Msg_ET	Msg_RT
Noeud : 0	00000000110	00000	00482	D	xxxxx	00000	00112	00131	00000	00112	00000	00112	00000
	00000000011	02792	00698	R	xxxxx	00000	00058	00077	00000	00058	00000	00058	00058
	00000000100	02568	00428	R	xxxxx	00000	00063	00190	00000	00063	00000	00063	00063
N°	IDs	Cycle	TTL	Type	CàR	N_Valid	Msg_E	Msg_R	Msg_Er	Msg_Ge	Msg_Del	Msg_ET	Msg_RT
Noeud : 1	00000000011	00000	00698	D	xxxxx	00000	00077	00090	00000	00077	00000	00077	00000
	00000000100	00856	00428	R	xxxxx	00000	00190	00190	00000	00190	00000	00190	00190
	00000000111	01496	00374	R	xxxxx	00000	00109	00109	00000	00109	00000	00109	00109
N°	IDs	Cycle	TTL	Type	CàR	N_Valid	Msg_E	Msg_R	Msg_Er	Msg_Ge	Msg_Del	Msg_ET	Msg_RT
Noeud : 2	00000000100	00000	00428	D	xxxxx	00000	00190	00231	00000	00190	00000	00190	00000
	00000000111	01496	00374	R	xxxxx	00000	00109	00109	00000	00109	00000	00109	00109
	00000000010	01072	00536	R	xxxxx	00000	00152	00152	00000	00152	00000	00152	00152
N°	IDs	Cycle	TTL	Type	CàR	N_Valid	Msg_E	Msg_R	Msg_Er	Msg_Ge	Msg_Del	Msg_ET	Msg_RT
Noeud : 3	00000000111	00000	00374	D	xxxxx	00000	00109	00193	00000	00109	00000	00109	00000
	00000000010	01072	00536	R	xxxxx	00000	00152	00152	00000	00152	00000	00152	00152
	00000000101	02360	00590	R	xxxxx	00000	00069	00138	00000	00069	00000	00069	00069

Figure 4.5 : Etat Des Nœuds

4-4-Détails des Nœuds :

Ces panneaux interprètent le fonctionnement interne de contrôleur est de transciever de chaque nœud; il fournit aux utilisateurs un rapport détaillé sur les activités instantanée (figure 4.6) ; chaque panneau repartie sur 04 groupes d'information comme suit :

4-4-1-Groupe Surveillance :

Il est constitué par les informations concernant la surveillance sur les états des composants internes de nœud :

a- **N° :** c'est le numéro d'identification de nœud.

b- **Etat de bus :** signifier l'occupation de chacune des bus (True : Libre et disponible, False : Occupé ou indisponible).

c- **Compteur :** c'est le nombre des bits récessif successif transmis dans chaque bus.

d- **L Msg_Rx :** c'est la taille estimé de message en cour de réception de chaque bus, elle est calculée depuis les bits [16..19] après la réception de 19 bits traités par la méthode de Stuffing.

e- **Crc Calculé :** c'est le résultat de la méthode qui calcul le Crc après la réception de la plus part du trame traitée par la méthode de Stuffing dans chaque bus ; il prouve l'intégrité du message (Crc = 0 : Msg correct, Crc <>0 : Msg Erroné).

4-4-2-Groupe de réception :

La réception de trame se fait de manière série c.-à-d. bit par bit comme l'émission; le Transciever regroupe ces bit selon l'ordre d'émission et puis appliquer la méthode Stuffing pour assembler la trame transmise; ce groupe est constitué par 04 buffer de réception comme suit : Msg_Rx01, Msg_Rx02, Msg_Rx03et Msg_Rx04.

4-4-3-Groupe d'émission :

L'émission d'une trame se fait de manière série dans un seul canal, chaque instant P il existe un seul signal d'un bit dans ce canal; le Transciever diffuse cette trame bit par bit après l'application de la méthode Stuffing; ce groupe est constitué par 04 buffer d'émission comme suit : Msg_Tx01, Msg_Tx02, Msg_Tx03et Msg_Tx04 ; le symbole '>' signifie le bit actuel émis.

4-4-4-Groupe Tampon :

Les trames de requête ou de donnée se génèrent par le contrôleur de nœud, avant la transmission de ces trames il faut les mémorisés dans le tampon en conséquence d'occupation des canaux.

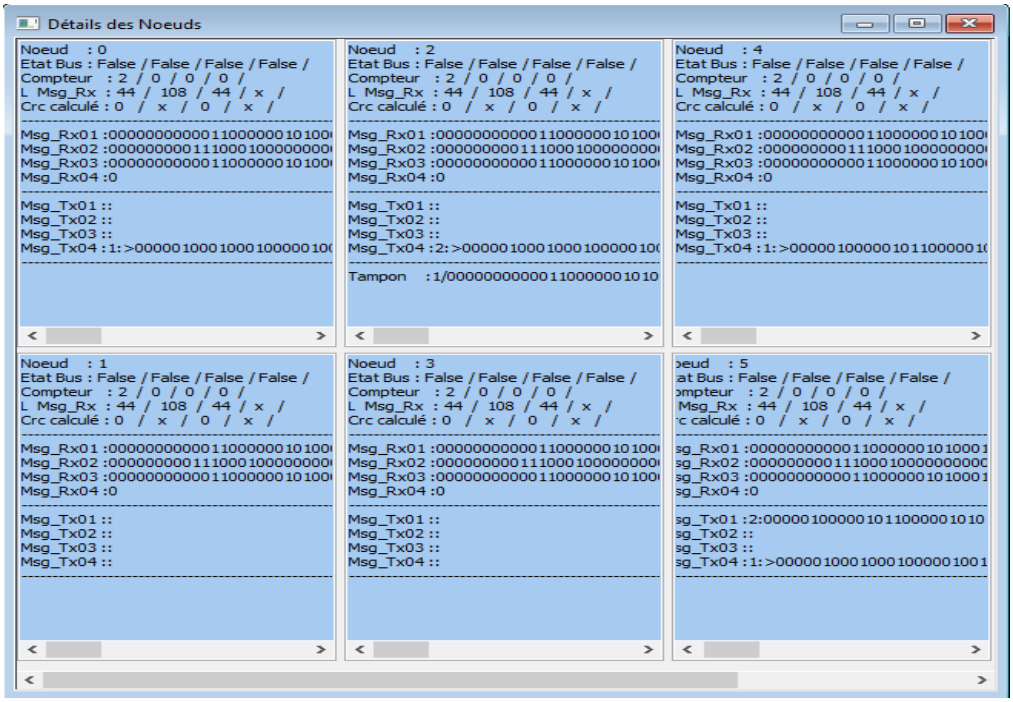


Figure 4.6 : Détails des Nœuds.

5-Résultat et discussion:

Après l'exécution d'une simulation en utilisant le modèle prédéfinie dans notre application, on récupère le résultat sauvegarder dans le Cloud; on a déterminé 02 fonctions de calcul qui illustrent le taux de la performance et de la fiabilité de la transmission des messages entre les nœuds à travers un nombre varie de bus. Tout d'abord on va décrire le modèle prédéfinie.

5-1-Description du modèle prédéfinie :

L'exécution de la simulation de modèle prédéfinie repose sur 05 étapes après le lancement de l'application; il faut d'abord cochés la case du modèle prédéfinie, sélectionner le sauvegarde dans le Cloud avec la période souhaitée, créer le mappe des nœuds, sélectionner les bus désirés puis démarrée la simulation; ce modèle est généré avec 8 nœud chacun composé par un ID pour Data et un autre ID pour Requête ; chaque ID possède des champs comme les montre le tableau.

Nœuds	IDs	Cycle	TTL	Type
Noeud0	00000010 ⇔ ID2	0	320	D
	00000100 ⇔ ID4	745	320	R
Noeud1	00000100 ⇔ ID4	0	320	D
	00000011 ⇔ ID3	540	320	R
Noeud2	00000011 ⇔ ID3	0	320	D
	00000001 ⇔ ID1	335	320	R
Noeud3	00000001 ⇔ ID1	0	320	D
	00000101 ⇔ ID5	745	320	R
Noeud4	00000101 ⇔ ID5	0	320	D
	00000111 ⇔ ID7	955	320	R
Noeud5	00000111 ⇔ ID7	0	320	D
	00000110 ⇔ ID6	955	320	R
Noeud6	00000110 ⇔ ID6	0	320	D
	00000000 ⇔ ID0	335	320	R
Noeud7	00000000 ⇔ ID0	0	320	D
	00000010 ⇔ ID2	540	320	R

Tableau 4.1 : Modèle prédéfinie à 8 nœuds

5-2- Statistique brute de simulation récupérée depuis Cloud :

L'exécution de simulation après la suivie des 05 étapes mentionner dans la description du modèle prédéfinie sauvegarde périodiquement chaque 100000 Pulses l'état des nœuds dans un répertoire nommé 'Cloud'+ date et heure actuelles ; le sauvegarde se fait en 02 fichiers (texte et CSV) nommés 'Etat'+ date et heure + Pulses actuelles; le fichier texte sauvegardé est plus détaillé que le fichier CSV. On répète l'exécution de simulation de ce modèle 04 fois en variant le nombre des bus de 01 à 04 ; après la fin de la quatrième simulation, on a représenté le résultat des fichiers CSV comme le montre le tableau ce dessous :

	Modèle				01 Canal				02 Canaux				03 Canaux				04 Canaux			
Nœuds	**IDs**	**Cycle**	**TTL**	**Type**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**
Noeud7	**ID0**	**0**	**320**	**D**	0	298	298	0	0	298	298	0	0	298	298	0	0	298	298	0
Noeud3	**ID1**	**0**	**320**	**D**	1	206	206	0	0	296	296	0	0	298	298	0	0	298	298	0
Noeud0	**ID2**	**0**	**320**	**D**	97	119	171	52	1	184	184	0	0	184	184	0	0	184	184	0
Noeud2	**ID3**	**0**	**320**	**D**	23	11	26	15	1	174	174	0	0	184	184	0	0	184	184	0
Noeud1	**ID4**	**0**	**320**	**D**	52	2	54	52	6	132	133	1	0	134	134	0	0	134	134	0
Noeud4	**ID5**	**0**	**320**	**D**	0	0	0	0	22	100	115	14	0	133	133	0	0	134	134	0
Noeud6	**ID6**	**0**	**320**	**D**	16	6	22	16	19	79	92	13	3	104	104	0	0	104	104	0
Noeud5	**ID7**	**0**	**320**	**D**	0	0	0	0	26	44	66	22	3	104	104	0	1	104	104	0
Nœuds	**IDs**	**Cycle**	**TTL**	**Type**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**	**N_Valid**	**Msg_E**	**Msg_R**	**Msg_Dl**
Noeud6	**ID0**	**335**	**320**	**R**	0	298	298	0	0	298	298	0	0	298	298	0	0	298	298	0
Noeud2	**ID1**	**335**	**320**	**R**	118	206	206	91	2	297	296	1	0	298	298	0	0	298	298	0
Noeud7	**ID2**	**540**	**320**	**R**	137	171	119	13	9	184	184	0	0	184	184	0	0	184	184	0
Noeud1	**ID3**	**540**	**320**	**R**	177	26	11	158	64	174	174	10	3	184	184	0	0	184	184	0
Noeud0	**ID4**	**745**	**320**	**R**	132	54	2	79	27	133	132	0	1	134	134	0	0	134	134	0
Noeud3	**ID5**	**745**	**320**	**R**	133	0	0	133	85	115	100	19	6	133	133	0	0	134	134	0
Noeud5	**ID6**	**955**	**320**	**R**	99	22	6	82	48	92	79	12	4	104	104	0	0	104	104	0
Noeud4	**ID7**	**955**	**320**	**R**	104	0	0	104	83	66	44	38	14	104	104	0	1	104	104	0

Tableau 4.2 : Tableau de résultat de la simulation en utilisant le modèle prédéfinie.

En peut facilement remarquer la différence entre le nombre des messages émis est les messages reçus ainsi que la démunissions des messages invalides et les messages supprimés par rapport à l'augmentation de nombre des canaux chaque simulation.

On remarque aussi une proportion direct entre le rapprochement des valeurs des messages émis et les messages reçus avec l'augmentation de nombre des canaux jusqu'à qu'il soit semblables.

5-3 Calcul de performance et fiabilité du système :

Nous admettrons par la suite que la performance et la fiabilité de notre système est liée à l'aptitude d'accomplir la transmission des messages dans les différentes modalités pendant une période spécifique et identique (100000 Pulses); elles sont déterminées par deux fonctions, Performance P(x) et Fiabilité F(x) :

5-3-1 La performance :

Le système embarqué en temps réel consiste à envoyer des trames de requête, lors la réception de ces dernières les nœuds qui les identifier répondent par des trames de données; la performance de système possède une relation entre ces trames; pour calculer la performance de notre système avec la fonction P(x) on a basé sur 02 paramètres (D : trames de donnée) et (R : trames de requête) de chaque modalité de simulation (V1 : utiliser 01 canal, V2 : utiliser 02 canaux, V3 : utiliser 03 canaux, V4 : utiliser 04 canaux) tel que :

a- La fonction de performance concernant les trames de donnée P(D):

Cette fonction calcul le taux de performance dans le contexte de trames de données émis par rapport aux trames de requêtes reçus comme suit :

$$P(D) = Msg\_E/Msg\_R \qquad (4.1)$$

L'application de cette fonction sur les résultats de la table TAB03 dont le type des trames est 'D' donne les taux de performance du système comme le montre le tableau ce dessous :

IDS	P(D) V1	P(D) V2	P(D) V3	P(D) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	100,0%	100,0%	100,0%	100,0%
ID2	69,6%	100,0%	100,0%	100,0%
ID3	42,3%	100,0%	100,0%	100,0%
ID4	3,7%	99,2%	100,0%	100,0%
ID5	0,0%	87,0%	100,0%	100,0%
ID6	27,3%	85,9%	100,0%	100,0%
ID7	0,0%	66,7%	100,0%	100,0%

Tableau 4.3 : Tableau des taux de performance P(D).

Les résultats du tableau 4.3 sont schématisés dans la figure 4.7.

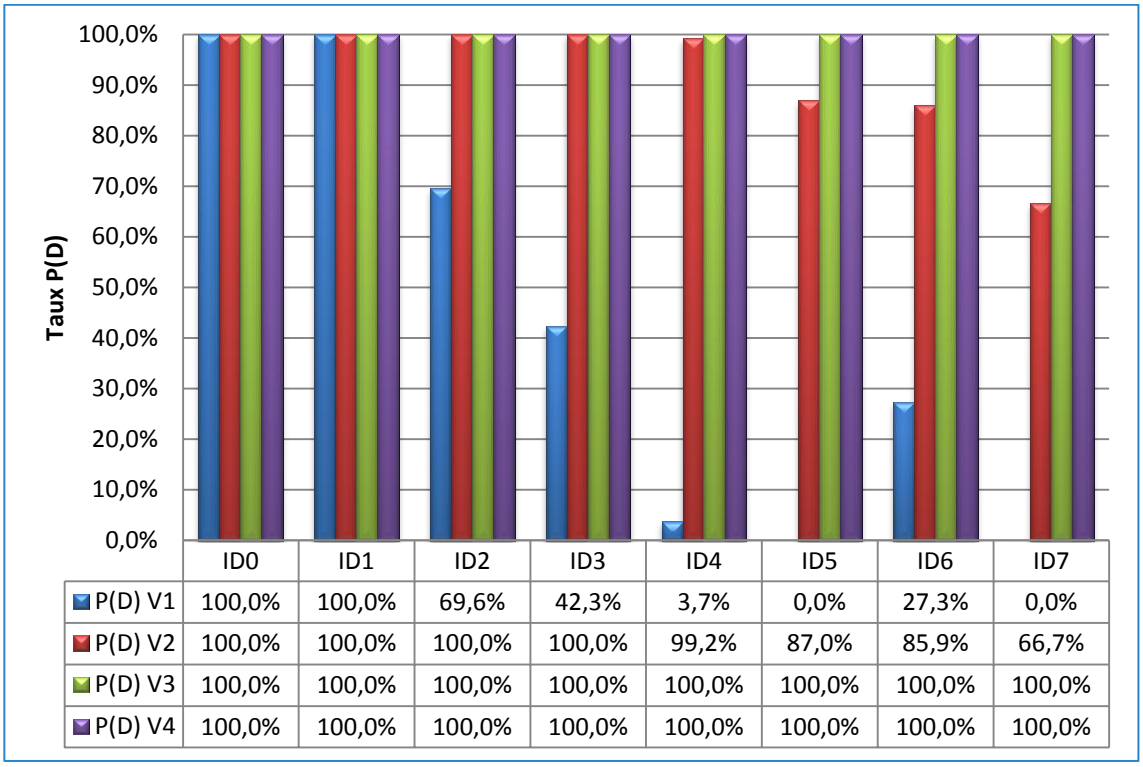


Figure 4.7 : Graphe des taux de performance P(D).

b- La fonction de performance concernant les trames de Requête P(R):

Cette fonction calcul le taux de performance dans le contexte de trames de données reçus par rapport aux trames de requêtes générés comme suit :

$$P(R) = Msg\_R/Msg\_Ge \qquad (4.2)$$

Tel que le nombre des trames de requête générés est calculé comme suit :

$$Msg\_Ge = Msg\_E + Msg\_Dl \qquad (4.3)$$

L'application de cette fonction sur les résultats de la table TAB03 dont le type des trames est 'R' donne les taux de performance du système comme le montre le tableau ce dessous :

IDS	P(R) V1	P(R) V2	P(R) V3	P(R) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	69,4%	99,3%	100,0%	100,0%
ID2	64,7%	100,0%	100,0%	100,0%
ID3	6,0%	94,6%	100,0%	100,0%
ID4	1,5%	99,2%	100,0%	100,0%
ID5	0,0%	74,6%	100,0%	100,0%
ID6	5,8%	76,0%	100,0%	100,0%
ID7	0,0%	42,3%	100,0%	100,0%

Tableau 4.4 : tableau des taux de performance P(R).

Les résultats du tableau 4.4 sont schématisés dans la figure 4.8.

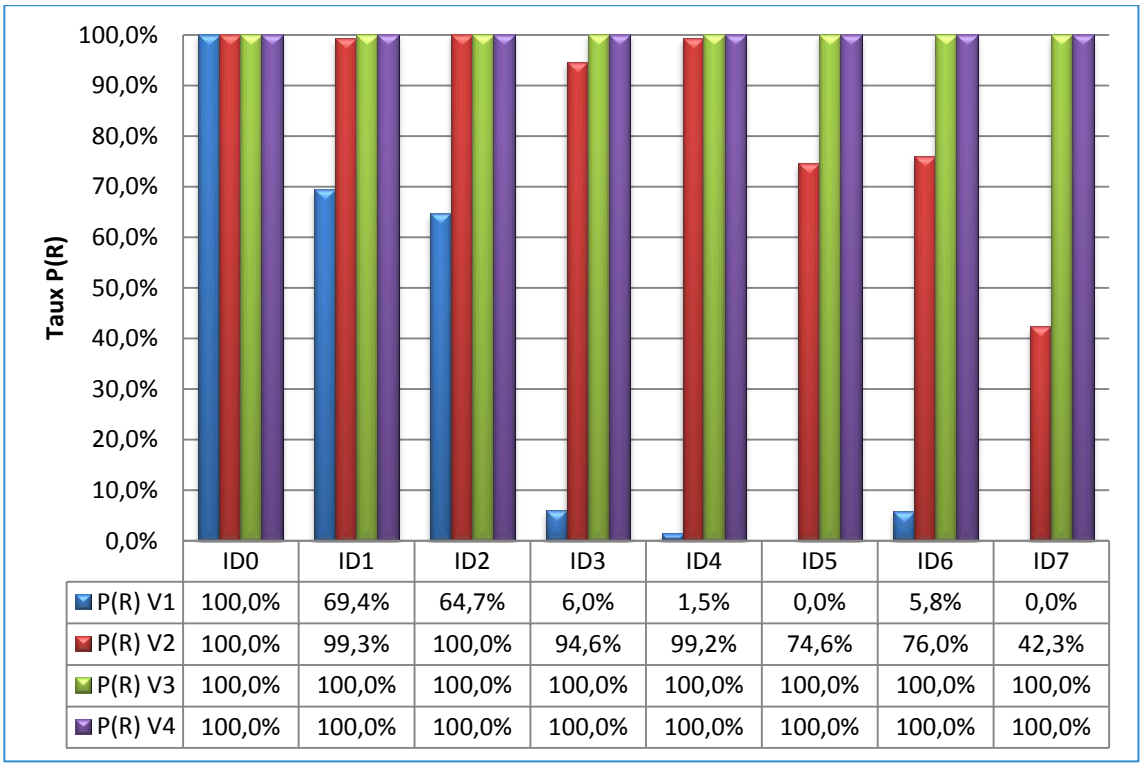


Figure 4.8 : Graphe des taux de performance P(R).

5-3-2 La fiabilité :

La relation entres les trames de requetés est les trames de données dépasse la performance dans le fait d'une requête émis doit avoir une réponse, donc une trame de requête il faut l'émis et il doit avoir une réponse dans un temps considéré; la fiabilité de système possède une relation entre ces trames dans ce temps considéré. Pour calculer la fiabilité de notre système avec la fonction F(x) on utilise le même principe de paramètre que la performance tel que :

a- La fonction de fiabilité concernant les trames de données :

Cette fonction calcul le taux de la fiabilité dans le contexte de trames de données émis avant l'écoulement de CàR par rapport aux trames de requêtes reçus comme suit :

$$F(D) = Msg\_ET / Msg\_R \qquad (4.4)$$

Tel que le nombre des trames de donnée émis en temps est calculé comme suit :

$$Msg\_ET = Msg\_R - N\_Valid \qquad (4.5)$$

L'application de cette fonction sur les résultats de la table TAB03 dont le type des trames est 'D' donne les taux de fiabilité comme le montre le tableau ce dessous :

IDS	F(D) V1	F(D) V2	F(D) V3	F(D) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	99,5%	100,0%	100,0%	100,0%
ID2	43,3%	99,5%	100,0%	100,0%
ID3	11,5%	99,4%	100,0%	100,0%
ID4	3,7%	95,5%	100,0%	100,0%
ID5	0,0%	80,9%	100,0%	100,0%
ID6	27,3%	79,3%	97,1%	100,0%
ID7	0,0%	60,6%	97,1%	99,0%

Tableau 4.5 : Tableau des taux de fiabilité F(D).

Les résultats du tableau 4.5 sont schématisés dans la figure 4.9.

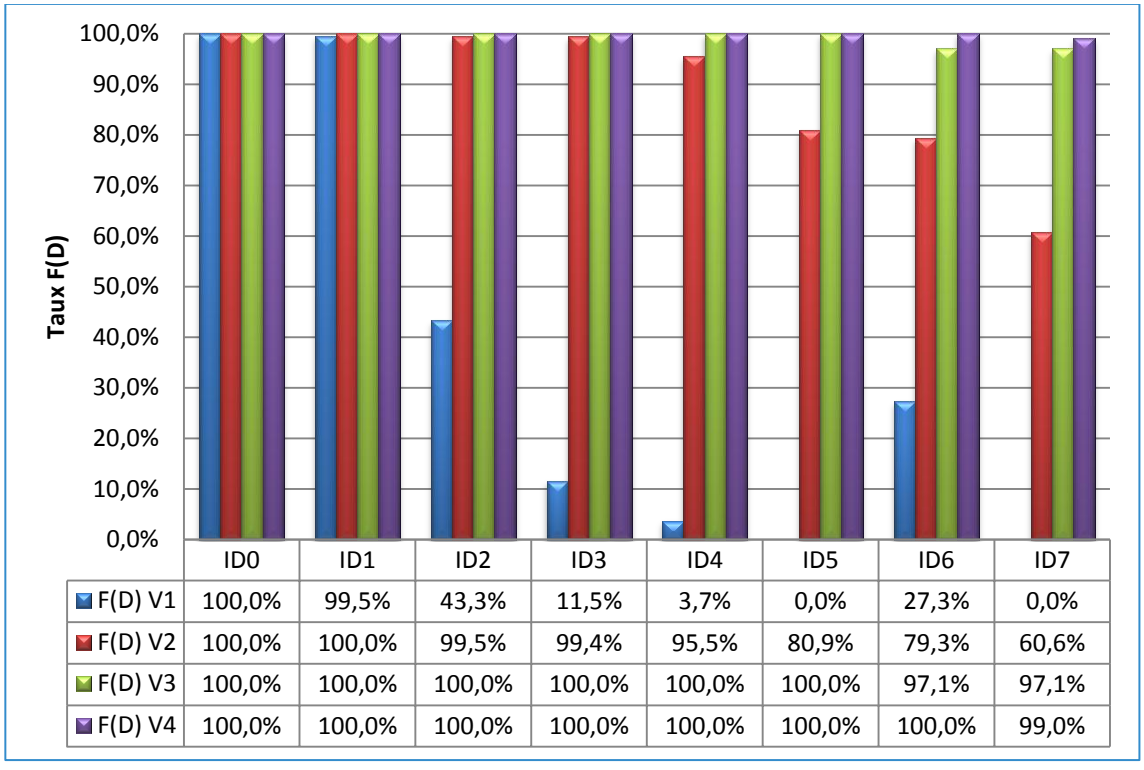


Figure 4.9 : Graphe des taux de fiabilité F(D).

b- La fonction de fiabilité concernant les trames de requête :

Cette fonction calcul le taux de la fiabilité dans le contexte de trames de données reçu avant l'écoulement de CàR par rapport aux trames de requêtes générées comme suit :

$$F(R) = Msg\_RT / Msg\_Ge \qquad (4.6)$$

Tel que le nombre des trames de donnée reçus en temps est calculé comme suit :

$$Msg\_RT = (Msg\_E + Msg\_Dl) - N\_Valid \qquad (4.7)$$

Et le nombre des trames de requête générés est calculé comme suit :

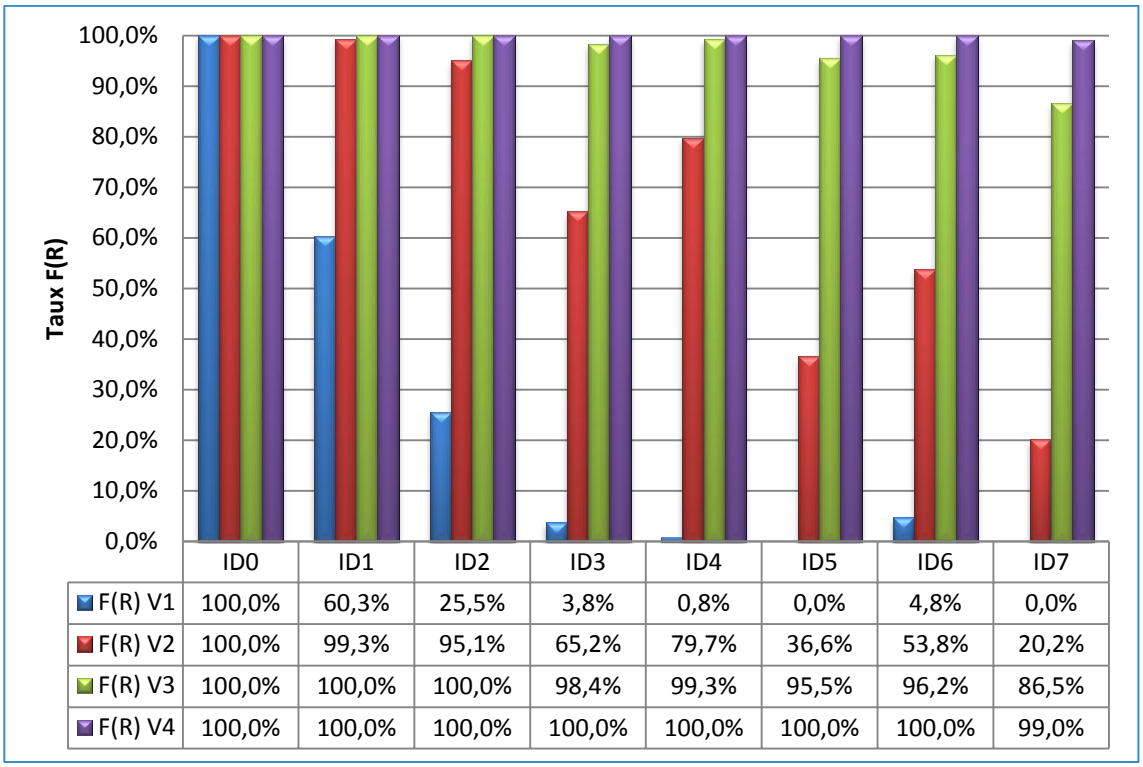
$$Msg\_Ge = Msg\_E + Msg\_Dl \qquad (4.8)$$

L'application de cette fonction sur les résultats de la table TAB03 dont le type des trames est 'R' donne les taux de fiabilité comme le montre le tableau ce dessous :

IDS	F(R) V1	F(R) V2	F(R) V3	F(R) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	60,3%	99,3%	100,0%	100,0%
ID2	25,5%	95,1%	100,0%	100,0%
ID3	3,8%	65,2%	98,4%	100,0%
ID4	0,8%	79,7%	99,3%	100,0%
ID5	0,0%	36,6%	95,5%	100,0%
ID6	4,8%	53,8%	96,2%	100,0%
ID7	0,0%	20,2%	86,5%	99,0%

Tableau 4.6 : Tableau des taux de fiabilité F(R).

Les résultats du tableau 4.6 sont schématisés dans la figure 4.10.



	ID0	ID1	ID2	ID3	ID4	ID5	ID6	ID7
F(R) V1	100,0%	60,3%	25,5%	3,8%	0,8%	0,0%	4,8%	0,0%
F(R) V2	100,0%	99,3%	95,1%	65,2%	79,7%	36,6%	53,8%	20,2%
F(R) V3	100,0%	100,0%	100,0%	98,4%	99,3%	95,5%	96,2%	86,5%
F(R) V4	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	99,0%

Figure 4.10 : Graphe des taux de fiabilité F(R).

6-Discutions :

Le modèle prédéfinie décrit ce dessus est déterminé d'une façon d'augmenter la concurrence entres les nœuds dans le contexte de transmission des trames pour qu'il nous aide à tester notre approche proposée qui consiste à tolérer les pannes et les fautes de transmission déterminés par la fonction de performance et la fonction de fiabilité.

Après avoir récupérer les résultats sauvegardés depuis le Cloud et les évaluer, on remarque que :

- ✓ Un changement radical significatif total de taux de performance du système au niveau de transmission des messages entre le taux de P(D) V1 et de P(D) V4 et entre le taux de P(R) V1 et de P(R) V4 d'après le tableau 04 et le tableau 05, dont il existe des taux nulles et autres faibles dans le système mono bus (V1), par contre dans le système à 04 bus (V4) sont totalement parfaits comme le montre le tableau 08.

IDS	P(D) V1	P(D) V4	P(R) V1	P(R) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	100,0%	100,0%	69,4%	100,0%
ID2	69,6%	100,0%	64,7%	100,0%
ID3	42,3%	100,0%	6,0%	100,0%
ID4	3,7%	100,0%	1,5%	100,0%
ID5	0,0%	100,0%	0,0%	100,0%
ID6	27,3%	100,0%	5,8%	100,0%
ID7	0,0%	100,0%	0,0%	100,0%

Tableau 4.7 : Tableau de comparaison de performance entre Système V1 et V4.

- ✓ Un changement radical significatif de taux de fiabilité du système au niveau de transmission des messages entre le taux de F(D) V1 et de F(D) V4 et entre le taux de F(R) V1 et de F(R) V4 d'après le tableau 06 et le tableau 07 dont il existe des taux nulles et autres faibles dans le système mono bus (V1), par contre dans le système à 04 bus (V4) les taux sont parfaits comme le montre le tableau 09.

IDS	F(D) V1	F(D) V4	F(R) V1	F(R) V4
ID0	100,0%	100,0%	100,0%	100,0%
ID1	99,5%	100,0%	60,3%	100,0%
ID2	43,3%	100,0%	25,5%	100,0%
ID3	11,5%	100,0%	3,8%	100,0%
ID4	3,7%	100,0%	0,8%	100,0%
ID5	0,0%	100,0%	0,0%	100,0%
ID6	27,3%	100,0%	4,8%	100,0%
ID7	0,0%	99,0%	0,0%	99,0%

Tableau 4.8 : Tableau de comparaison de fiabilité entre Système V1 et V4.

- ✓ Un changement croissant et important entre les taux de performance du système au niveau de transmission des messages lorsque on augmente à chaque fois le nombre des bus à partir de P(D) V1 jusqu'à P(D) V4 et à partir de P(R) V1 jusqu'à P(R) V4 d'après le tableau 04 et le tableau 05.
- ✓ Un changement croissant et important entre les taux de fiabilité du système au niveau de transmission des messages lorsque on augmente à chaque fois le nombre des bus à partir de F(D) V1 jusqu'à F(D) V4 et à partir de F(R) V1 jusqu'à F(R) V4 d'après le tableau 06 et le tableau 07.

Notre approche proposée permette la tolérance aux pannes qu'elle est représentée par l'augmentation de nombre des bus, tel qu'on a un bus cesse de fonctionner il existe d'autres bus assure la continuité de fonctionnement du système ; elle permette aussi la tolérance aux fautes qu'elle représenté par gestion de priorité de choisir un canal pour la retransmission du trame en cas d'une apparition d'une faute causée par un bruit de signal, une déficience d'un bus ou perdre l'arbitrage dans un bus et ce qui correspond à l'objectif principal de la recherche. L'augmentation de nombre de bus due à la disponibilité du câble sur marché, la facilité d'ajouter des bus, le cout moins chère du câble et le rendement parfait de cette augmentation.

7-Conclusion

Ce chapitre a été consacré à l'évaluation de la performance et de la fiabilité des systèmes embarque en temps réel, nous avons proposé une simulation d'un modèle basé sur le protocole CAN et qui consiste à faire connecter M nombre de nœud a N nombre de Canales, Nous commençons avec un seul canal puis ajoutons un par un jusqu'à ce que nous atteignons la quatrième canal, après avoir extrait les résultats du Cloud nous appliquons les fonctions de performance et de fiabilité pour calculer le taux de réussite et évaluer la capacité de notre méthode a toléré les fautes de transmission des messages et à les corriger.

Au final, nous avons mené une étude sur les résultats obtenus, qui sont représentés sous forme des tableaux et des graphiques, ces derniers nous a montrer des améliorations significatives des mécanismes de la tolérance aux fautes et panes, et d'après ces résultats, il a été constaté que le fonctionnent déroule comme prévu dans des états ordinaires. .

Conclusion Générale

Les systèmes embarques temps réel modernes tels que les systèmes d'un complexe nucléaire ou les systèmes des avions de guerre sont des systèmes de plus en plus critiques désormais à logiciels prépondérants, et souvent plonges dans un milieu physique perturbateur voire hostile.

La conception des tels systèmes nécessite la collaboration de plusieurs équipes spécialisées dans des domaines déférents comme la sureté de fonctionnement, la sureté informatique et la communication…, cette dernière qu'elle soit internes (câblage torsadé) ou externes (Cloud Computing) entre ces systèmes posent d'autres challenges pour les développeurs.

Dans ce mémoire, nous avons considéré une propriété cruciale qui doit être prise en compte à la conception des systèmes embarqués, qui est la tolérance aux fautes celle-là est la technique adoptée dans ce travail pour réaliser un système sûr de fonctionnement.

Ce système doit en plus respecter des contraintes temps réel en absence et en présence des fautes, pour atteindre cet objectif nous avons présenté une proposition pour tolérer un problème majeur (perte d'arbitrage dans le protocole CAN) qui dirige à des dégâts fatals dans le déroulement de notre système temps réel.

Notre proposition engage de faire augmenter le nombre de canaux de transmission allons de 1 bus jusqu'au 4 bus, pour faire donner une grande chance aux nœuds qui portent des identificateurs grands à transmettre ses données dans les délais; sachons déjà que la grandesse de l'identificateur implique l'augmentation de l'échec de transmission vis-à-vis aux identificateurs petits, à travers cette incrémentation de nombre des bus nous avons constaté une très grande amélioration dans les taux de performances et même la fiabilité touchons jusqu'à 100 % pour tous les nœuds de notre model prédéfinie.

Comme perspectives a ce travail, nous ambitionnons de faire :

- l'ajout des nouvelles techniques de tolérance aux fautes.
- l'amélioration de l'approche proposée dans le contexte d'extensibilité.
- essayer de rendre le système exploite les réseaux de capteurs.
- de bien réaliser un simulateur d'un système embarqué temps réel base sur notre approche et les technologies électroniques récentes.

Bibliographie

[FAB 12] Fabrice Jumel Livre Introduction aux systèmes embarqués INSA de Lyon pages 6-7-8 2012.

[TAM 05]Tammy NoergaardLivreEmbedded Systems Architecture A Comprehensive Guide for Engineers and Programmers page05-2005.

[LOU 17]L. LoukilMémoire de rechercheuniversité d'Oran https://m.20-bal.com/law/12648/index.html?page=6 (08h:047/03/2021).

[MEH 15] MEHALAINE Ridha Mémoire magister Optimisation de la consommation d'énergie sous contraintes temporelles au niveau des systèmes embarqués distribués temps réel, 2015pages 7.

[PER18] Perry Xiao Designing LiverEmbedded Systems and the Internet of Things (IoT) with the ARM Mbed page3.

[SAM 14] Samia BouzefraneMémoire de recherche Introduction aux systèmes temps réel CEDRIC –CNAM page03 .

[IUL 21] IulianOberSite web Introduction aux systèmes temps réel - https://www.irit.fr/~Iulian.Ober/str/STR-1.1.IntroSTR.pdf -09h25 05/03/2021 .

[MAR 12] mohamedmarouf THESE Doctorat Ordonnancement temps réel dur multiprocesseur tolérant aux fautes appliqué à la robotique mobile pages 23-24-25 juin 2012 .

[OMA 09] Omar kermia THESE Doctorat en science Ordonnancement temps réel multiprocesseur de tâches non-préemptives avec contraintes de précédence, de périodicité stricte et de latence 2009.

[ROH04] ROHÁRIK VÎLCU thèse doctorat l'Université Paris XII – Val de Marne systèmes temps réel embarqués ordonnancement optimal de tâches pour la consommation énergétique du processeur Mihaela 2004.

[GRE 09] Grevet N Mémoire de rechercheLecloudcomputing évolution ou révolution. Aout 2009.

[AMA 21] SITE WEB aws.amazon https://aws.amazon.com/fr/what-is-cloud-computing/ mars 2021 22h00.

[WAR 11] Warin S Livre Le Cloud Computing. Réelle révolution ou simple évolution 2011.

[HOL 21] SITE WEB https://www.holded.com/fr/blog/histoire-du-cloud mars 2021 .

[AND 11] Anderson J. LIVRE Choisissez votre cloud Août 2011.

[ORA 21] ORACLE LIVRE . Architectural Strategies for Cloud Computing.

[LEF10] LEFORTPROJETA.CloudComputing.Projettutoré en licence professionnelle asrall, 2010 .

[WAR 15] Livre blanc IaaS, PaaS, SaaS : quel Cloud choisir ?2015. https://www.globalsp.com/wp-content/uploads/2015/11/livre-blanc-iaas-paas-saas-quel-cloud-choisir-pour-votre-entreprise-global-sp.pdf

[MSE16]Sehit, M., Bracquemond, A., Soubra, H., &Ramdan Cherif, A. (2016). Méthode d'évaluation des exigences de fiabilité pour des modules logiciel embarqués à partir de spécifications. Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement, 11-13 Octobre 2016, Saint Malo, France.

[MME10] Measurement Modelling and Evaluation of Computing Systems and Dependability and Fault Tolerance: 15th International GI/ITG Conference, MMB&DFT 2010, Essen, Germany, March 15-17, 2010. Proceedings

[FTC87]Pelegrin, M. J. (1987). Fault Tolerant Considerations and Methods for Guidance and Control Systems. ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE).

[MEF07]Mihalache, A. G. (2007). Modélisation et évaluation de la fiabilité des systèmes mécatroniques: application sur système embarqué (Doctoral dissertation, Université d'Angers).

[DFT09]De Florio, V. (2009). Dependability and fault-tolerance: Basic concepts and terminology. In Application-Layer Fault-Tolerance Protocols (pp. 1-20).IGI Global.

[AFT85]Avižienis, A. (1985, December). The N-Version Approach to Fault-Tolerant Software. IEEE Trans. Software Eng., 11, 1491–1501.

[STR94]Rushby, J. (1994). Critical Systems Properties: Survey and Taxonomy. Reliability Engineering and System Safety, 43(2), 189–219.

[PDC95]Laprie, J.-C.(1995). Dependability—Its Attributes, Impairments and Means.InB.Randell, J.-C.Laprie, H. Kopetz, & B. Littlewood (Eds.), Predictably Dependable Computing Systems (pp. 3–18). Berlin: Springer Verlag.

[FTP81]Anderson, T., & Lee, P. (1981).Fault Tolerance—Principles and Practice.Pren- tice-Hall.

[SDT92] levkov v etat de l'art sur les systemes distribues temps reel tolerant les fautes ; Juin 1992

[RCS99] Dominique Paret ; HassinaRebaine « Réseaux de communication pour systèmes embarqués

CAN, CAN FD, LIN, FlexRay, Ethernet… 2 e édition1999

[MEE16]Sehit, M., Bracquemond, A., Soubra, H., &Ramdan Cherif, A. (2016). Méthode d'évaluation des exigences de fiabilité pour des modules logiciels embarqués à partir de

spécifications. Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement, 11-13 Octobre 2016, Saint Malo, France.

[FTC87]Pelegrin, M. J. (1987). FaultTolerantConsiderations and Methods for Guidance and Control Systems.ADVISORY GROUP FOR AEROSPACE RESEARCH AND DEVELOPMENT NEUILLY-SUR-SEINE (FRANCE).

[MEF07]Mihalache, A. G. (2007). Modélisation et évaluation de la fiabilité des systèmes mécatroniques: application sur système embarqué (Doctoral dissertation, Université d'Angers).

[DFT09]De Florio, V. (2009). Dependability and fault-tolerance: Basic concepts and terminology. In Application-Layer Fault-ToleranceProtocols (pp. 1-20).IGI Global.

[AFT85]Avižienis, A. (1985, December). The N-Version Approach to Fault-Tolerant Software. IEEE Trans. Software Eng., 11, 1491–1501.

[STR94]Rushby, J. (1994). CriticalSystemsProperties: Survey and Taxonomy. Reliability Engineering and System Safety, 43(2), 189–219.

[PDC95]Laprie, J.-C.(1995). Dependability—ItsAttributes, Impairments and Means.InB.Randell, J.-C.Laprie, H. Kopetz, & B. Littlewood (Eds.), PredictablyDependableComputingSystems (pp. 3–18). Berlin: Springer Verlag.

[FTP81]Anderson, T., & Lee, P. (1981).FaultTolerance—Principles and Practice.Pren- tice-Hall.

[SDT92] LEVKOV V ETAT DE L'ART SUR LES SYSTEMES DISTRIBUES TEMPS REEL TOLERANT LES FAUTES ; Juin 1992.

[DPR98]Multiplexed Networks for Embedded Systems CAN, LIN, FlexRay, Safe-by-Wire..by Dominique Paret, RoderickRiesco MA (z-lib.org)1998 .