Tolérance aux pannes

Module 10

INF8480 Systèmes répartis et infonuagique

Michel Dagenais

École Polytechnique de Montréal
Département de génie informatique et génie logiciel

Sommaire

Tolérance aux pannes

Un peu de terminologie

- Sûreté de fonctionnement (*Dependability*)
  - Composantes (*Attributes*)
    - Disponibilité (*Availability*)
    - Fiabilité (*Reliability*)
    - Sécurité (*Security*)
    - Confidentialité (*Confidentiality*)
    - Intégrité (*Integrity*)
    - Maintenabilité (*Maintainability*)
  - Mécanismes (*Means*)
    - Prévention des fautes (*Fault prevention*)
    - Tolérance aux fautes (*Fault tolerance*)
    - Eradication des fautes (*Fault removal*)
    - Prévision des fautes (*Fault forecasting*)
  - Problèmes (*Impairments*)
    - Faute (*fault*)
    - Erreur (*Error*)
    - Panne ou défaillance (*Failure*)

Systèmes fiables

Disponibilité: être prêt à l'utilisation.

Fiabilité: continuité de service.

Sûreté: pas de conséquences catastrophiques pour l'environnement.

Sécurité: prévention d'accès non autorisés et/ou de la manipulation de l'information (confidentialité, intégrité, disponibilité).

Maintenabilité: réfère à la facilité avec laquelle un système défaillant peut être réparé.

Définitions

- **Défaillance (panne):** lorsque le comportement d'un système viole sa spécification de service
  - Les défaillances résultent de problèmes inattendus internes au système qui se manifestent éventuellement dans le comportement externe du système.
- Ces problèmes sont appelés erreurs et leurs causes mécaniques ou algorithmiques sont dites fautes.
- Quatre sources de fautes peuvent causer la défaillance d'un système:
  - Spécification inadéquate.
  - Erreurs de conception dans le logiciel.
  - Défaillance matérielle.
  - Interférence sur le sous-système de communication

Classification des fautes

Transitoire: se produit de manière isolée (une fois et disparaît); rayonnement alpha qui perturbe un bit de mémoire.

Intermittente: se reproduit sporadiquement (transitoire et survient à plusieurs reprises); problème de bruit, de mauvaise synchronisation, de couplage entre signaux, de chaleur...

Permanente: persiste indéfiniment (jusqu'à réparation) après son occurrence; disque brisé, circuit brûlé...

Classification des défaillances

détectée: le serveur répond avec un message d'erreur (disque avec parité incorrecte sur un bloc).

plantage: serveur s'arrête, mais il fonctionnait correctement jusqu'alors.

panne d'omission: serveur ne répond pas à une requête; bonne réponse ou pas de réponse (contrôleur de disque brûlé).

panne de temporisation: la réponse survient en dehors de l'intervalle de temps réel spécifié; délai excessif (commande de centrale nucléaire, commande de vol, transactions à haute fréquence).

panne de réponse: réponse est simplement incorrecte; capteur bruité, mémoire corrompue.

panne arbitraire (Byzantine): réponses arbitraires voire malicieuses (testeur qui essaie de déjouer le système, attaque de sécurité).

Tolérance aux pannes

Approches pour atteindre la Fiabilité

- Eviter les fautes
  - Prévention de fautes: comment prévenir l'occurrence ou l'introduction de fautes
  - Élimination de fautes: comment réduire la présence (nombre et sévérité) de fautes
- Accepter les fautes
  - Tolérance aux fautes: comment fournir un service en dépit des fautes
  - Prévision de fautes: comment estimer la présence, la création et les conséquences des fautes

Prévention de Fautes par évitement

- Utilisation des composants les plus fiables en respectant les coûts et contraintes de performance.
- Choix des meilleures techniques d'assemblage et d'interconnexion des composants.
- Langages de programmation avec abstraction de donnée, modularité, sécurité.
- Environnement de développement et méthodologie structurés aidant à gérer la complexité et ne rien oublier.
- Examiner les formes d'interférences.
- Spécification rigoureuse des besoins, si pas formelle.

Elimination de Fautes

- La prévention ne peut éliminer la possibilité de fautes. Il faut détecter et éliminer les fautes. Prévoir un jeu de test le plus complet possible, dans des conditions de charge réalistes.
- Un test peut seulement être utilisé pour démontrer la présence de fautes, pas leur absence complète.
- Il est parfois impossible de tester sous des conditions réelles.
- La plupart des tests sont faits dans un mode de simulation, et il est difficile de garantir que la simulation est exacte.
- Les erreurs introduites durant la spécification des besoins peuvent ne pas se manifester jusqu'à ce que le système tombe en panne.

Tolérance aux fautes

- **Récupération de faute:** si un délai de récupération est acceptable.
- Il suffit de pouvoir reprendre où on avait laissé, après avoir redémarré (et possiblement réparé) le système.
- Requiert un stockage permanent fiable pour ne perdre aucune information. Copies de sauvegarde, cliché à intervalle régulier...
- Faire attention de bien tout sauver avant d'accepter une transaction.

Tolérance aux fautes

- Masquer la présence de fautes en utilisant la redondance
- **Redondance matérielle:** composants matériels ajoutés pour supporter la tolérance aux fautes à tous les niveaux (alimentation électrique, processeurs, disques, réseau...).
- **Redondance logicielle:** inclut tous les programmes et instructions utilisés pour tolérer les fautes.
- **Redondance temporelle:** temps extra pour exécuter les tâches (exécuter les instructions plusieurs fois) pour la tolérance aux fautes.

Niveaux de tolérance aux fautes

Tolérance aux fautes complète: le système continue à fonctionner en présence de fautes, sans perte significative de fonctionnalité ou de performance.

Dégradation graduelle: le système continue à fonctionner en présence de fautes, acceptant une dégradation partielle des fonctionnalités ou de performance durant le recouvrement ou la réparation.

Arrêt sécuritaire: le système maintient son intégrité tout en acceptant un arrêt temporaire de son fonctionnement.

Notes:

- Niveau de tolérance aux fautes nécessaire dépend de l'application.
- (La plupart des systèmes critiques nécessitent une tolérance complète, mais plusieurs se contentent d'une dégradation graduelle.)

Phases de la tolérance aux fautes

Détection d'erreurs: la présence des fautes est déduite en détectant une erreur.

Recouvrement d'erreurs: élimine les erreurs de telle façon qu'elles ne se propagent pas par des actions futures.

Phases de la tolérance aux fautes (suite)

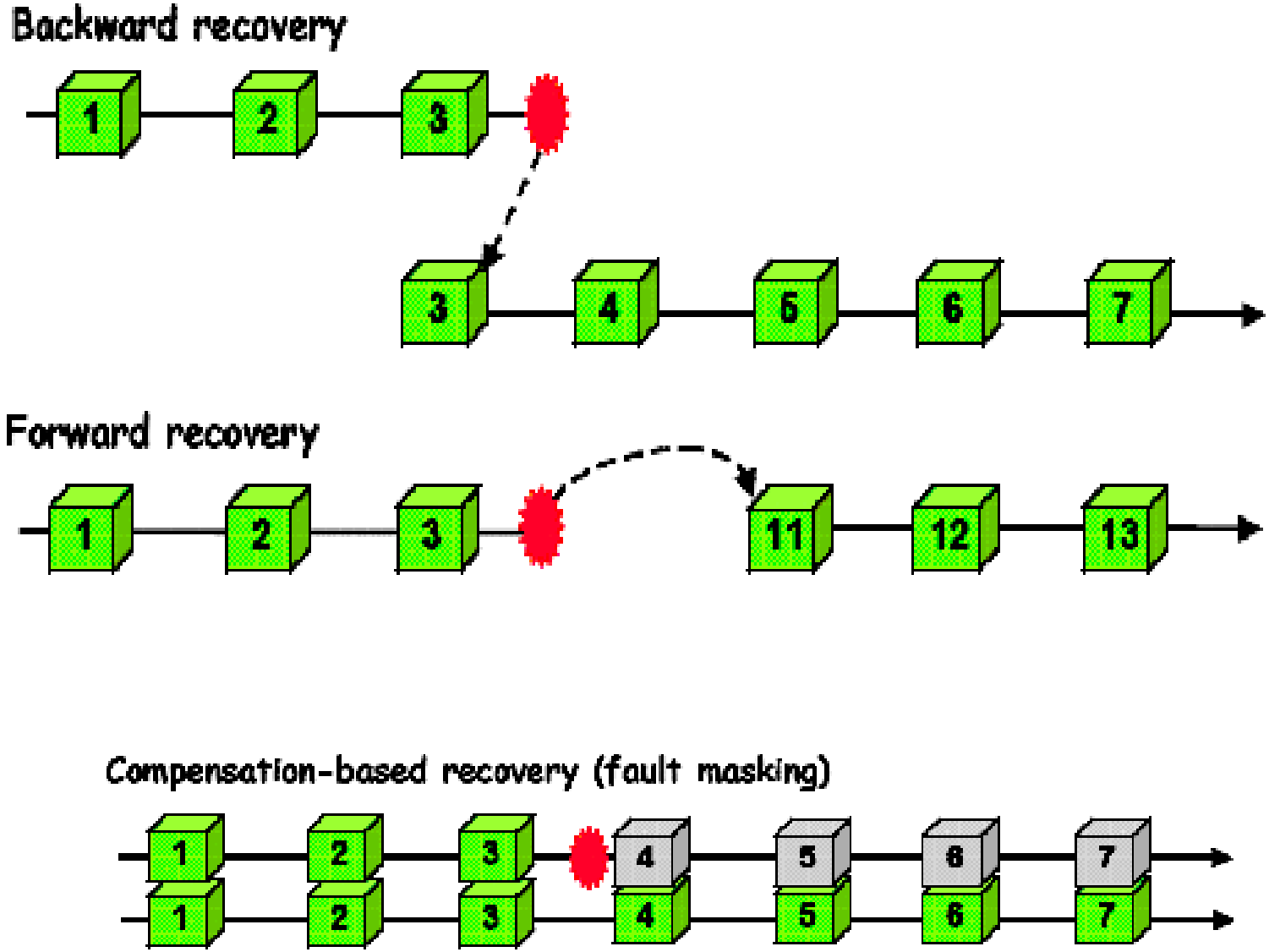
Recouvrement d'erreurs en aval: continuer à partir de l'état erroné en faisant des corrections sélectives à l'état du système. (Remplacer un disque défectueux dans une unité RAID)

Recouvrement d'erreurs en amont: consiste à restaurer le système à un état précédent sûr et en exécutant une section alternative du programme. (Remplacer un disque défectueux et l'initialiser avec une copie de sauvegarde)

Point de recouvrement: point auquel un processus est restauré

Point de contrôle (*checkpoint*): établissement d'un point de recouvrement (en sauvegardant l'état approprié du système)

Méthodes de recouvrement d'erreurs



Traitement de fautes et continuité de service

- Le recouvrement d'erreur retourne le système à un état exempt d'erreur; l'erreur peut survenir à nouveau. La phase finale de la tolérance aux fautes est d'éradiquer la faute du système.
- Une erreur est un symptôme de faute; corriger l'erreur ne répare pas la faute.
- Le composant défaillant (ou la panne) doit être identifié.
- Le traitement automatique des pannes est difficile et spécifique au système.
- Localisation de la panne et réparation du système
- Les techniques de détection d'erreur peuvent aider à retracer la panne
  - Faute matérielle: le composant peut être remplacé
  - Faute logicielle: peut être éliminée dans une nouvelle version du code
  - Dans les applications qui ne doivent pas s'arrêter il est nécessaire de modifier le programme alors qu'il s'exécute!

Masquage hiérarchique des fautes

- Un niveau détecte, reprend et masque les erreurs au niveau plus bas:
  - Le client demande l'adresse IP d'un ordinateur.
  - Le serveur de nom fait une requête récursive à d'autres serveurs et prend les serveurs alternatifs lorsqu'un serveur ne répond pas.
  - Un serveur interrogé découvre une erreur sur un disque et lit plutôt de son second disque en miroir.
  - Le client a la réponse et ne sait rien des défaillances.

Masquage par groupe des fautes

- Le client demande à tous les serveurs, prend le premier qui répond. Peut tolérer $\frac{n-1}{n}$ pannes par omission.
- Le client demande à tous les serveurs et prend un vote sur les réponses. Peut tolérer $\frac{n-2}{n}$ pannes si les chances d'avoir deux mauvaises réponses identiques sont presque nulles, ou $\frac{n}{2n+1}$ pannes même si tous les ordinateurs défaillants tentent vicieusement de s'entendre sur une mauvaise réponse.
- **Groupe synchronisé:** tous les serveurs doivent recevoir les mêmes mises à jour dans le même ordre.
- **Groupe non synchronisé:** les serveurs de secours enregistrent les mises à jour dans un journal mais sans les traiter. Au besoin, ils utilisent ce journal pour se mettre à jour et prendre le relais avec un certain délai.

Tolérance aux fautes matérielle

Statique: composants redondants utilisés pour cacher les effets des fautes

- **Exemple:** Triple Modular Redundancy (TMR): 3 composants identiques et un circuit de vote majoritaire; les résultats sont comparés et si un diffère des deux autres il sera masqué.
- Assume que la faute n'est pas commune (telle qu'une erreur de conception) mais elle peut être transitoire ou causée par la détérioration du composant.
- Pour masquer les fautes de plus d'un composant il faut NMR (N-tuple Modular Redundancy).

Dynamique: redondance utilisée à l'intérieur d'un composant indiquant si le résultat est erroné.

- Fournit une technique de détection d'erreurs; le recouvrement doit être fait par un autre composant

Tolérance aux fautes logicielle

- Utilisée pour détecter les erreurs de conception
- **Statique:** programmation N-Versions
- **Dynamique:**
  - Détection et recouvrement
  - Blocs de recouvrement: recouvrement d'erreurs en amont
  - Exceptions: recouvrement d'erreurs en aval

Tolérance aux pannes

Réplication de données

- Maintien de copies d'un objet sur plusieurs systèmes.
- Peut améliorer les performances du système (2x plus de bande passante de lecture pour deux disques en miroir, même performance qu'un seul disque en écriture).
- Augmente la disponibilité en masquant les fautes et donc minimisant l'impact des pannes.

Effets positifs de la réplication

Performance:

- Mise en cache des données accédées fréquemment, ou la réplication des données près du point d'accès pour réduire le temps d'accès.
- Répartition des données sur plusieurs serveurs pour répartir la charge de travail.
- La réplication d'objets statiques est triviale tandis que la réplication d'objets dynamique requiert des mécanismes de contrôle de la concurrence et de maintient de la cohérence (ajoute de la surcharge de traitement et de communication).

Disponibilité:

- Offrir une disponibilité du service le plus près possible de 100%
- 2 types de pannes:
  - Serveur: réplication de données sur des serveurs qui tombent en panne indépendamment des autres.
  - Partition du réseau ou opérations déconnectées: les utilisateurs dans leurs déplacements se connectent/déconnectent aléatoirement au réseau.

Requis généraux pour la réplication de données

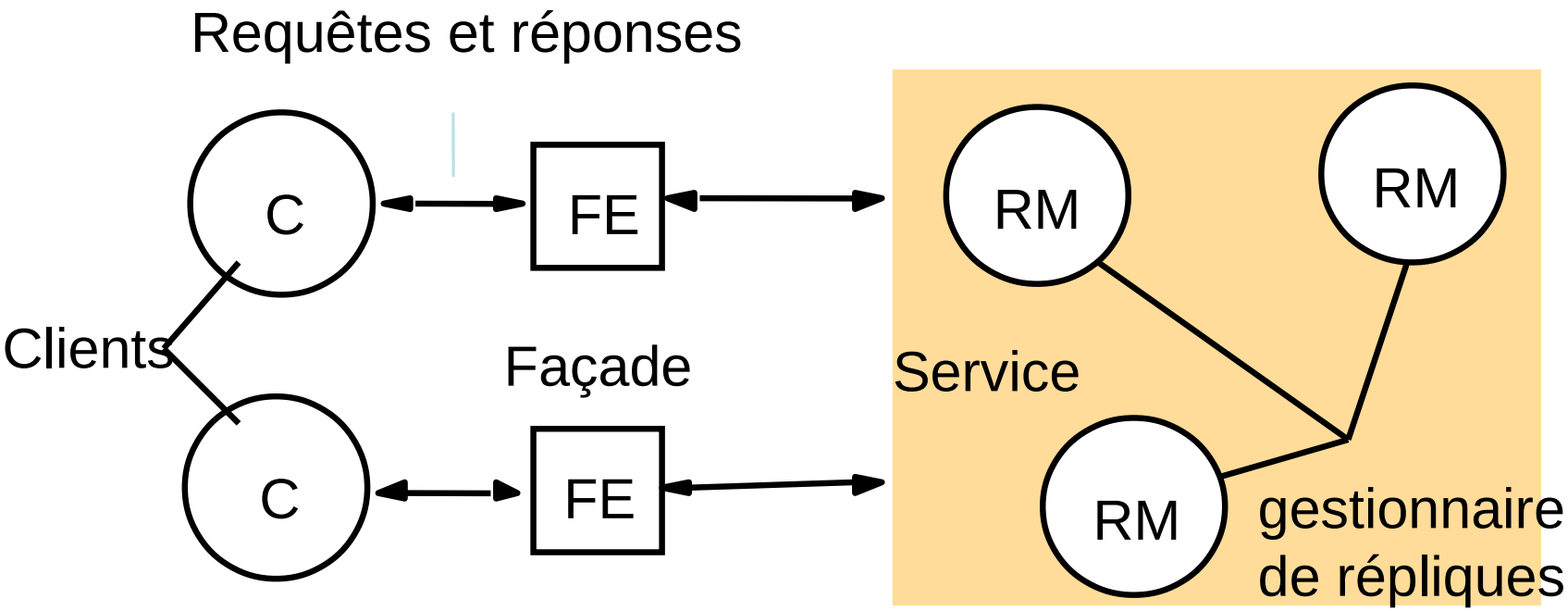
- Transparence de la réplication:
  - Les utilisateurs voient des objets logiques, ils n'ont aucune connaissance des multiples copies physiques qui existent
  - Ils accèdent une instance de l'objet, peu importe laquelle, et reçoivent un résultat unique

- Cohérence:
  - Étant donné plusieurs copies physiques de l'objet et des accès concurrents, les copies doivent être maintenues dans un état cohérent entre elles

Modèle du système

- Les données dans le système sont considérées comme un ensemble d'objets logiques.
- Chaque objet logique est implémenté par un ensemble d'objets physiques appelés répliques.
- La cohérence de chacune des répliques dépend des opérations appliquées localement par le système sur lequel elles résident.
- Les répliques peuvent être incohérentes entre elles à un point donné dans le temps.
- Le système en soi est asynchrone et peut tomber en panne.

Modèle architectural pour la réplication



Modèle architectural de base pour la réplication

- Le système se compose de trois éléments :
  - Les clients effectuent les requêtes sur des objets répliqués à travers les FE (*Front End*).
  - Les FE agissent comme intermédiaire pour les clients et s'occupent de contacter les RMs (*Replica Manager*). Ils assurent la transparence pour les utilisateurs
  - Les RMs maintiennent les répliques d'objets qui sont sous sa responsabilité
    - Les objets peuvent se trouver répliqués sur tout ou sur un sous-ensemble des RMs
    - Les RMs offrent un service d'accès aux objets répliqués au client (accès en lecture ou en mise à jour)

Les étapes de base pour l'accès aux objets répliqués

1. Requête
   - La requête d'un client est prise en charge par le FE qui s'occupe de la relayer à un ou plusieurs RM (diffusion sélective)
2. Coordination
   - Les RMs décident d'accepter ou non la requête, lequel RM va servir la requête et dans quel ordre par rapport aux autres requêtes (FIFO, causal, total)
3. Exécution
   - Le ou les RMs concernés réalisent la requête (peut-être partiellement pour permettre le recouvrement de l'ancien état de l'objet)
4. Accord
   - Les RMs se mettent d'accord sur les effets de la requête sur les différentes instances de l'objet répliqué (approche immédiate ou paresseuse)
5. Réponse
   - Un ou plusieurs RMs renvoient la réponse au FE ayant effectué la requête, qui lui-même la donne au client concerné

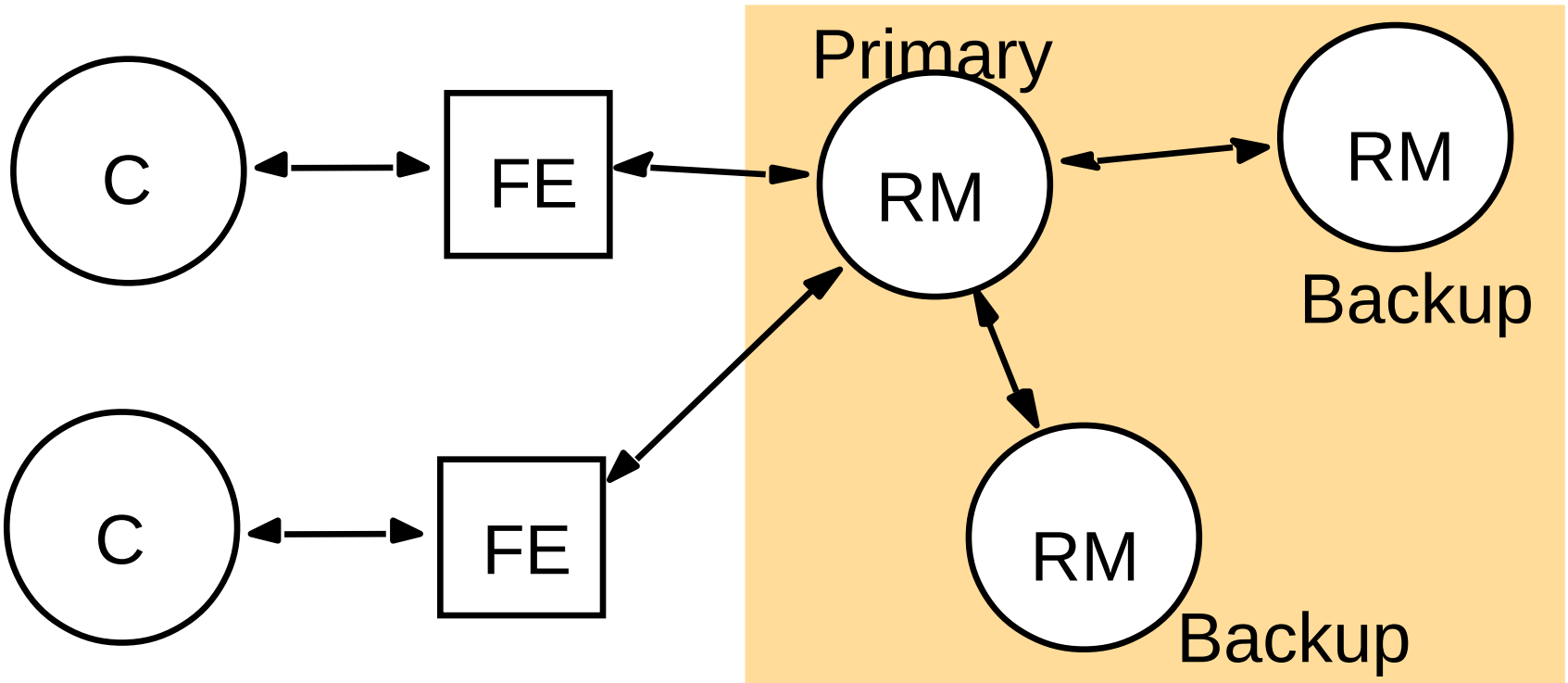
Service tolérant aux fautes

- **Un service a un comportement dit correct s'il fonctionne selon les spécifications même en présence de pannes**
- Les clients ne peuvent voir la différence entre un service obtenu à partir de données répliquées et celui obtenu à partir d'une copie unique
- On doit s'assurer que l'ensemble des répliques produit le même comportement qu'une seule copie

Modèle de réplication passive

- Dans ce modèle, à tout moment il y a un RM actif (primaire) qui répond aux requêtes des clients et des RMs passifs (secondaires) qui servent de RMs de secours en cas de panne.
- Le résultat d'une opération effectuée sur le RM primaire est communiqué à tous les secondaires (ce qui permet de maintenir la cohérence des données).
- Quand le RM primaire tombe en panne, un RM secondaire prend la relève (après une élection, par exemple).
- Le système est linéarisable puisque la séquence des opérations est programmée et contrôlée par le RM primaire.
- Si le RM primaire tombe en panne, le système demeure linéarisable puisqu'un des RMs secondaires reprend exactement là où le RM primaire est tombé en panne.

Modèle de réplication passive



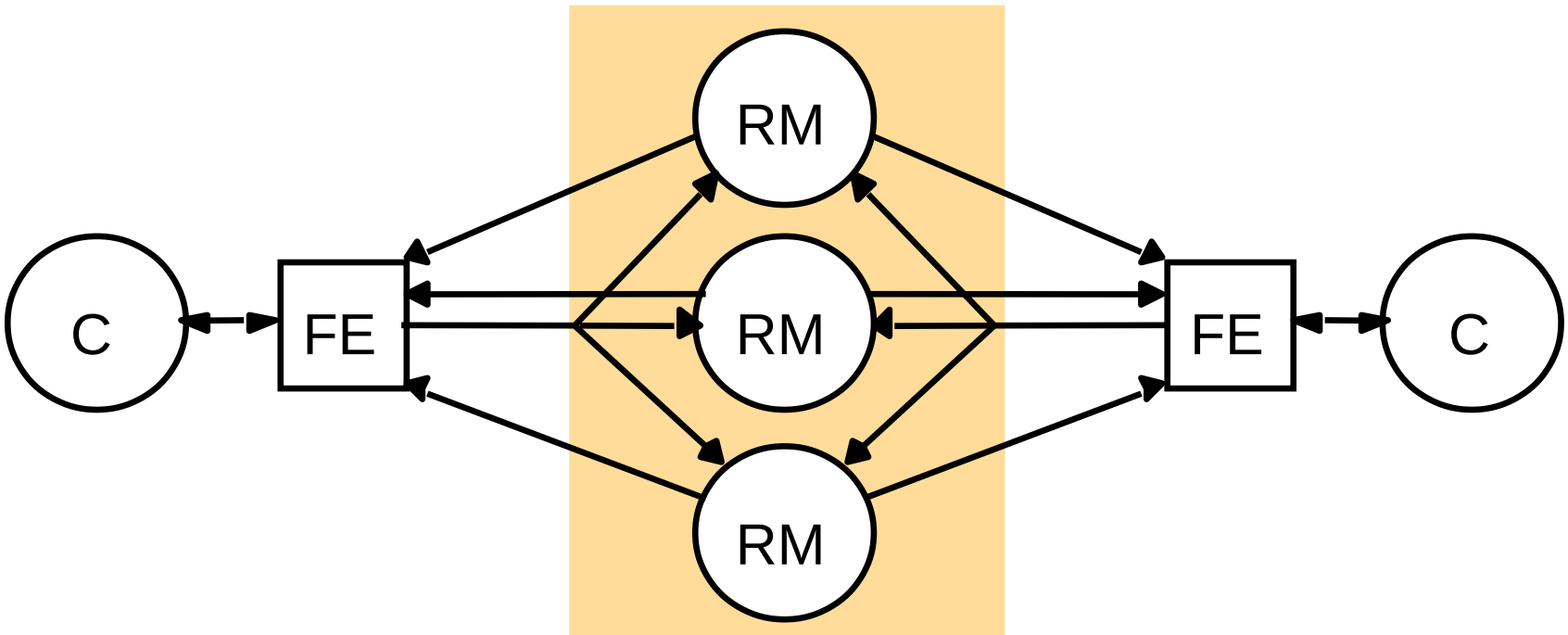
Exemples de réplication passive

- Serveur NIS: un serveur primaire qui propage les modifications à plusieurs serveurs secondaires.
- Linux virtual server: un serveur primaire a deux adresses IP, une pour la gestion et une pour le service. Un serveur de rechange reçoit les mises à jour et interroge souvent le serveur primaire. Si le serveur primaire ne répond plus, il le déconnecte, prend son adresse IP de service et commence à s'annoncer sous cette adresse et à servir les requêtes.
- Deux serveurs DHCP peuvent coexister s'ils offrent des blocs d'adresses différents. Le second serveur répond s'il voit que le premier ne répond pas aux requêtes à tous pour une adresse dynamique.

Modèle de réplication active

- Les RMs sont des machines à état qui jouent exactement le même rôle et sont organisés en groupe.
- Les RMs débutent dans le même état, et changent d'état concurremment, de telle manière que leurs états restent identiques.
- Si un RM tombe en panne, cela n'a aucun effet sur les performances du système parce que les autres RMs peuvent prendre la relève de manière transparente.
- Ce modèle requiert un mécanisme de communication à diffusion sélective (*multicast*) fiable et totalement ordonné (pour que les RMs effectuent les mêmes opérations dans le même ordre).
- Supporte la cohérence séquentielle mais pas la linéarisation (car l'ordre total ne correspond pas nécessairement à l'ordre d'exécution temporel des opérations).

Modèle de réplication active



Exemple de réplication active

- Serveurs NFS avec automount: la requête est envoyée à tous les serveurs et le premier à répondre est pris.

- CODA: plusieurs serveurs offrent les mêmes fichiers et se propagent les modifications. Très flexible mais ne peut garantir l'absence de conflits (mises à jour concurrentes).

Tolérance aux pannes

Le consensus en réparti

- Plusieurs processus, corrects ou fautifs, échangent des messages.
- Chaque processus doit prendre une décision (e.g. qui est le serveur primaire).
- Terminaison: éventuellement tous les processus corrects arrivent à une décision.
- Consensus: tous les processus corrects terminent avec la même décision.
- Intégrité: si tous les processus corrects proposent la même décision, cette décision doit l'emporter.
- Chaque processus correct communique sa proposition et celle déjà connue d'autres processus au groupe. Chaque processus accumule les propositions des autres et se soumet à la proposition majoritaire.

Difficultés

- Le coordonnateur élu importe peu, l'important est d'avoir un consensus!
- L'élection hiérarchique produit un élu dans chaque partition du réseau, ce qui n'est pas acceptable si on doit avoir un coordonnateur unique (verrou, base de donnée).
- Exiger un vote à majorité? Ceci assure qu'un seul coordonnateur peut être élu.
- Chaque candidat va chercher des votes. Que faire en cas de résultat minoritaire? Peut-on changer son vote?

Algorithme de Paxos

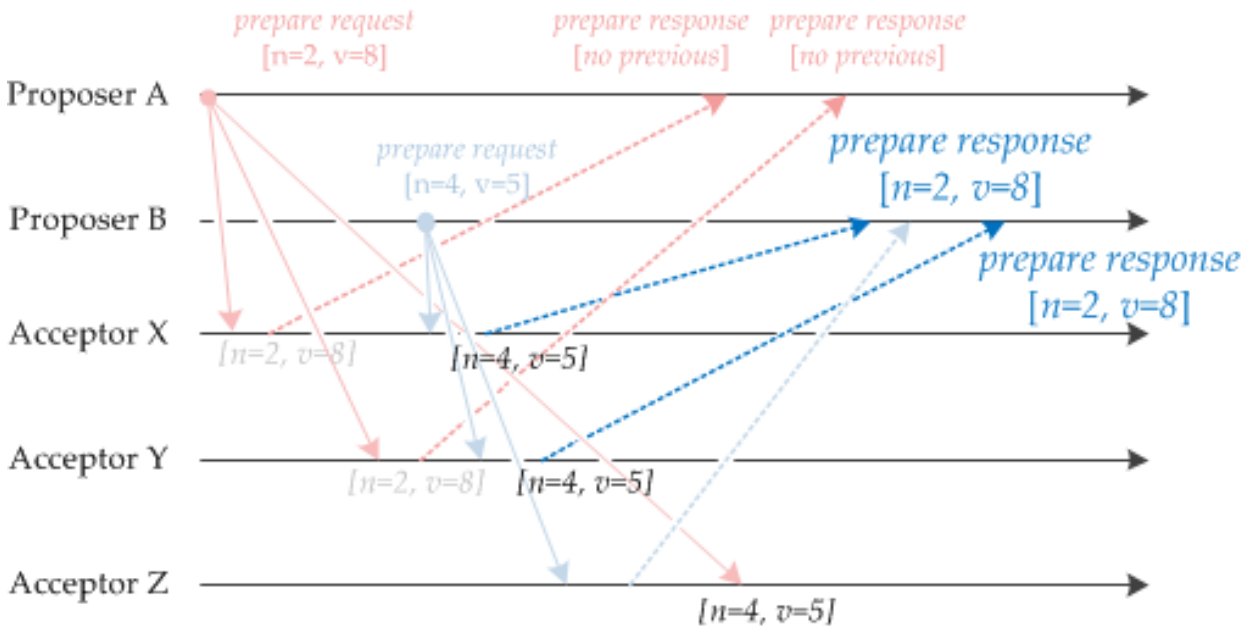
- Le consensus en réparti est très difficile à assurer en présence de défaillances, messages asynchrones, partitionnement de réseau...
- Proposé par Lamport en 1989 et publié dans une revue, après avoir été entièrement revisé, seulement en 1998.
- Complexe, utilisé surtout pour les questions fondamentales (e.g., élire un serveur primaire de manière sécuritaire, ensuite le serveur primaire peut coordonner le reste).

Algorithme de Paxos

- Par exemple, si on veut élire (établir un consensus) un serveur primaire, il faut obtenir une majorité parmi les accepteurs.
- Chaque proposition a un numéro de séquence et une valeur.
- Première proposition reçue par un accepteur, il promet de ne pas accepter une proposition antérieure et mémorise cette proposition.
- Proposition antérieures reçues par un accepteur, il promet mais retourne la plus récente proposition reçue.
- Le proposeur choisit la proposition la plus récente déjà reçue par un accepteur et demande à chaque accepteur de l'accepter.
- Si une majorité d'accepteurs acceptent, le consensus est obtenu.

Exemple de l'algorithme de Paxos

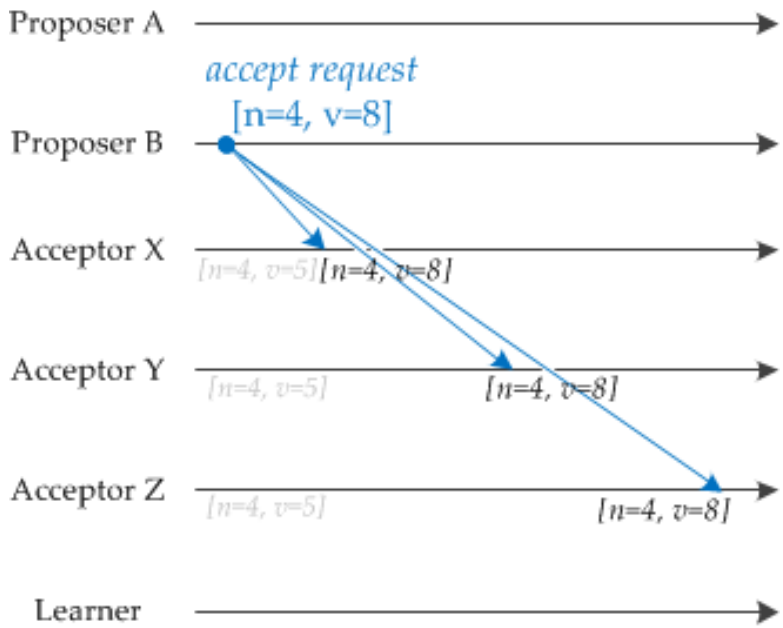
- A propose "n=2, v=8" qui est accepté par X et Y, et B propose "n=4, v=5" qui est accepté par X et Y (en remplacement de n=2) et par Z. Ensuite, Z ignore la proposition de A car elle est antérieure (n=2).



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

Exemple de l'algorithme de Paxos (suite)

- A, avec 2 votes sur 3, envoie "accept n=2, v=8" qui est ignoré (désuet) par X, Y et Z; il échoue. B envoie "accept n=4, v=8" (v=8 de proposition antérieure de A) accepté par tous. Consensus atteint!



https://medium.com/@angusmacdonald/paxos-by-example-66d934e18522

Algorithme Raft

- Reliable/Replicated/Redundant And Fault-Tolerant: RAFT (un radeau pour fuir l'ile de Paxos).
- Proposé par Diego Ongaro et John Ousterhout en 2013: "In Search of an Understandable Consensus Algorithm".
- Preuve formelle de sécurité.
- Utilisé par Etcd, MongoDB et plusieurs autres systèmes.

Algorithme Raft

- Lors de son initialisation ou lorsqu'il perd le contact avec le chef, un membre incrémente le numéro de terme (époque) et se propose comme candidat. Il envoie un message à tous les autres membres pour avoir leur appui.
- Chaque membre ne vote qu'une seule fois pour un terme.
- Si un numéro de terme plus élevé apparaît, l'élection en cours devient caduque et est perdue.
- Si un candidat obtient une majorité absolue de votes, il est élu.
- Les délais avant de demander un vote sont choisis aléatoirement pour minimiser les chances de collisions.
- Le nouveau chef élu reste en poste aussi longtemps qu'il est opérationnel et que le contact n'est pas perdu.

Le consensus avec Raft

- Le chef est responsable de coordonner un état répliqué entre tous les membres.
- Toutes les nouvelles entrées sont envoyées au chef.
- Le chef met l'entrée dans son journal et l'envoie aux membres.
- Chaque membre ajoute l'entrée dans son journal et répond que c'est accepté.
- Si une majorité de membres acceptent l'entrée, elle est commise par le chef.
- Le chef envoie un message pour confirmer que l'entrée est commise.
- Si un nouveau chef est élu, il vérifie avec chaque membre les dernières entrées commises et les propage à tous.

Tolérance aux pannes

Transactions réparties avec réplication

- Le protocole à deux phases pour compléter une transaction doit s'étendre aux données répliquées qui doivent elles aussi changer de manière atomique avec le reste.

- Le coordonnateur de transaction parle au gestionnaire de copies qui agit à son niveau comme coordonnateur.
  - Les messages "*prêt*", et dans une seconde phase "*compléter*" ou "*annuler*", sont relayés à toutes les copies avant de produire une réponse au coordonnateur.
  - C'est un peu comme si le coordonnateur de transaction devait contacter toutes les copies.

Transactions réparties avec réplication (suite)

- Lecture sur un, écriture sur tous: verrou en lecture sur un, ou verrou en écriture sur tous (bloqué si un verrou en lecture existe).
- Copie primaire: tous les verrous sont pris sur la copie primaire, au moment d'accepter la transaction, toutes les mises à jour sont propagées aux copies.
- Il faut faire attention si les copies disponibles pendant la transaction changent, que le tout demeure cohérent.
- Lorsque le réseau est partitionné, on peut continuer dans chaque partition et résoudre les conflits plus tard (optimiste), ou ne continuer que dans la partition qui a le quorum (si une partition a plus que la moitié et est donc la seule ainsi), ou en être réduit à des lectures seulement puisqu'il n'est plus possible d'obtenir des verrous d'écriture sur tous.

Tolérance aux pannes

Calcul de disponibilité

- Probabilité qu'un composant soit fonctionnel au temps t: $\frac{MTTF}{MTTF+MTTR}$ (Mean Time to Fault / Repair).
- Probabilité p que deux événements indépendants, de probabilité p_1 et p_2, se produisent en même temps: $p = p_1 \times p_2$.
- Probabilité p que p_1 ou p_2 se produise: $p = p_1 + p_2 - p_1 \times p_2$
- Les probabilités de cas disjoints ($p_1 \times p_2 = 0$) peuvent s'additionner.
- Probabilité que k disques sur n soient fonctionnels ($n - k$ en panne), si la propabilité pour un disque est p_d.

$$p = \binom{n}{k} \times p_d^k \times (1 - p_d)^{n-k}$$

  - Sommation de k à n pour k disques ou plus fonctionnels.
  - Au moins 1 disque fonctionnel dans un miroir: $1 - (1 - p_d)^2$

Probabilités: peut-on sommer

- Probabilité que deux événements indépendants de probabilité p1 et p2 se produisent au même moment: $p1 \times p2$. Par exemple, un ordinateur est fonctionnel si la carte mère (pm) et le disque (pd) sont fonctionnels. Il y a une panne sauf si les deux sont fonctionnels.

$$1 - p_m \times p_d$$

- Peut-on plutôt prendre la probabilité de panne de carte + la probabilité de panne de disque?

$$(1 - p_m) + (1 - p_d) \neq 1 - p_m \times p_d$$

Probabilités: sommer si disjoint

- Quelle est la différence? La panne de disque se produit indépendamment de la carte qui peut être fonctionnelle ou non. On compte donc deux fois le cas où le disque et la carte sont défectueux. Les deux éléments sommés se recoupent alors qu'il faut bien tout séparer. Si p1 et p2 sont grands, la partie sommée en double est petite et le résultat est incorrect mais très proche. Si p1 et p2 sont petits, le résultat dépasse largement 1!

$$P(p_1 \cup p_2) = p_1 + p_2 - p_1 \times p2 =$$

$$(1 - p_m) + (1 - p_d) - ((1 - p_m) \times (1 - p_d)) =$$

$$(1 - p_m) + (1 - p_d) - (1 - p_d - p_m + p_m \times p_d) = 1 - p_m \times p_d$$

Systèmes parfaitement tolérants aux pannes?

- Avec le bon niveau de redondance, on peut diminuer les probabilités de panne pratiquement à 0, sous certaines hypothèses.
- Cinq 9 (0.99999) donne 5.26 minutes par an de panne, six 9 donne 31.5 secondes par an.
- En pratique, les pannes sont plus fréquentes car certaines hypothèses ne tiennent pas, surtout sur les nouveaux systèmes.
- Pour les systèmes plus matures, on analyse les accidents et on apprend des erreurs (e.g., en aviation).

Disques en miroir

- Un disque tombe en panne environ une fois par 4 ans (1h/4ans = 0.00003). Avec 2 disques en miroir les probabilités de pannes simultanées sont à peu près nulles ($.00003^2$ = 1h/150000ans). Plus besoin de copies de sauvegarde?!?
- Lors d'une panne, il faut être alerté et effectuer la réparation rapidement pour revenir à une configuration redondante.
- Le matériel redondant doit être en bon état de marche.
- En fin de vie, les probabilités de panne augmentent rapidement. Deux vieux disques deviennent problématiques.
- Il y a parfois des lots de disques non fiables avec fin de vie prématurée. Plusieurs vont acheter des disques de lots / modèles différents.
- Vol, feu, foudre, surtension...

Tolérance aux pannes

Copies de sauvegarde

- Un administrateur système a effectué des copies de sauvegarde, pendant plus d'une année, sans remarquer de problème, avant qu'un usager demande de récupérer un fichier à partir des copies. L'unité de ruban qui prenait les copies était défectueuse et les rubans illisibles.
- Un usager avisé, intégrant un nouveau groupe, feint de perdre un fichier afin de le faire relire et de s'assurer que la procédure de copies de sauvegarde fonctionnait bien. Quelques temps plus tard, son disque tombe en panne. Une partie des fichiers manquants étaient sur le ruban relu qui avait été oublié dans l'unité de ruban et son contenu écrasé.
- Un administrateur consciencieux faisait des copies de sauvegarde et effectuait régulièrement des tests. Un jour, l'unité de ruban est tombée en panne et il a découvert qu'elle était désalignée car aucune autre unité de ruban compatible ne pouvait relire les rubans.

Mémoire du Xerox Alto

- Système de mémoire du Maxc avec correction d'un bit en erreur et détection de deux bits en erreur. Aucune erreur rapportée dans les fichiers d'erreur.
- Le même système est repris pour le Alto mais avec détection pour 1 bit seulement. Aucune erreur pour 6 mois. Avec le logiciel d'édition plein écran Bravo, de nombreuses erreurs sont apparues subitement.
- Sur le Maxc, une mauvaise configuration faisait que les erreurs de parité n'étaient pas rapportées. De plus, certains patrons de bits causent beaucoup plus d'erreur dans les circuits de mémoire, ce qui est arrivé avec Bravo sur le Alto.
- Sur le Alto 2, avec des circuits de mémoire plus récents, les concepteurs ont remis la correction et la parité. Le système était très fiable mais ils ont découvert éventuellement que 25% de la mémoire n'était pas protégée. Meilleurs circuits? Erreurs non détectées?

Ordinateurs dans la navette spatiale

- Quatre ordinateurs redondants qui exécutent le même logiciel de manière synchrone, avec leurs sorties qui sont synchronisées et comparées 400 fois par seconde. Un cinquième ordinateur de relève qui exécute un autre logiciel programmé de manière indépendante.
- Lors du premier lancement, une erreur a été détectée laissant penser que plusieurs ordinateurs étaient défectueux et le lancement a été retardé.
- C'était un problème de synchronisation.
- Le second logiciel développé de manière indépendante ne protège pas contre les erreurs de spécification.

La fusée Ariane 5

- Le système informatique a été repris tel quel de la fusée Ariane 4. Rien de plus conservateur et sécuritaire!?!
- La valeur d'accélération pour Ariane 4 ne pouvait pas dépasser une certaine valeur maximale.
- Ariane 5 était plus rapide, lorsque la valeur maximale a été dépassée, l'ordinateur a été déclaré fautif et l'ordinateur de relève a été activé.
- L'ordinateur de relève a détecté la même erreur d'accélération trop grande et s'est désactivé.
- Le premier lancement n'a pas bien fonctionné...

Commutateurs 4ESS de AT&T, 15 janvier 1990

- Nouvelle version de logiciel installée le décembre précédent sans problème apparent.
- Un problème indépendant en janvier cause une faute qui est détectée sur un commutateur (e.g., A). La procédure corrective est de faire redémarrer le commutateur, ce qui prend 4 à 6 secondes.
- Le système A envoie un message aux autres commutateurs (e.g., B) qu'il n'acceptera plus d'appel car il redémarre.
- Après le redémarrage, l'ancien système envoyait un message pour reprendre les appels, le nouveau recommence directement à initier des appels.
- Le commutateur B, en recevant un appel de A, comprend que A redevient fonctionnel, il réinitialise sa logique interne pour en tenir compte.
- Si B reçoit un autre appel de A pendant qu'il réinitialise sa logique, il devient confus et initie un redémarrage comme A l'avait fait.
- En peu de temps, le réseau de 114 commutateurs n'était plus qu'une cascade sans fin de redémarrages.

L'appareil Therac 25

- Logiciel de commande pour un appareil à faisceau d'électron qui peut produire une lumière de positionnement, un faisceau d'électron ou un faisceau d'électron très puissant avec filtre spécial pour générer des rayons X.
- Lors d'une entrée du mode rayons X par erreur par l'opérateur, qu'il change tout de suite en mode faisceau d'électrons, la grande puissance était sélectionnée sans qu'il n'y ait de filtre pour rayons X en place.
- Le patient pouvait recevoir 100 fois la dose prévue.
- Le logiciel avait été pris tel quel d'une version antérieure. Le problème n'avait pas eu de conséquence avec l'ancien appareil car des mesures de protection par matériel étaient en place.

Les pannes sont-elles vraiment indépendantes?

- Ordinateurs redondants avec mêmes logiciel/matériel, vulnérabilités, employés, bâtiment...
- Les trois capteurs de vitesse du vol Air France 447 de Rio à Paris sont pris dans la glace et donnent une vitesse de 0. L'autopilote de désengage et les pilotes réagissent mal.
- Plusieurs moteurs des avions L-1011 ont fait défaut simultanément. La même équipe avait fait l'entretien de ces moteurs et oublié un joint d'étanchéité torique dans le circuit hydraulique.
- Les 4 moteurs du vol British Airways 9 de Londres à Auckland se sont arrêtés presque simultanément près de Jakarta. La poussière d'un nuage volcanique a causé le problème.