FTF

Tema 2: Fases en la Tolerancia a Fallos

1.- Introducción

2. Patología de los fallos

3.- Detección del error

4.- Estimación y confinamiento de daños

5.- Recuperación del estado correcto

6.- Localización del fallo y reconfiguración del sistema

FTF

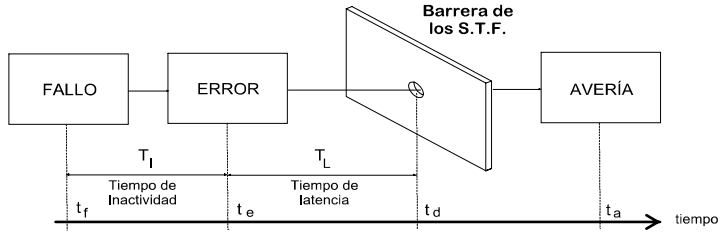
1. Introducción

- La implementación de la tolerancia a fallos en cualquier sistema vendrá siempre ligada a sus características, diseño e implementación. Las técnicas dependerán de las necesidades y funcionalidad del sistema.
- Existen sin embargo algunos principios generales que se pueden utilizar.
- Para conseguir la tolerancia a fallos se identifican 4 fases:
  - Detección del error
  - Estimación de daños
  - Recuperación del error
  - Tratamiento del fallo y continuación del servicio del sistema
- En algunos sistemas, alguna fase se realiza de forma implícita, o es una tarea trivial.

FTF

2. Patología de los fallos

- Podemos considerar la relación temporal en el proceso de creación de los fallos, errores y averías



- T. de inactividad (Fallo dormido): $T_I = t_e - t_f$
- T. de latencia (Fallo latente): $T_L = t_d - t_e$
- Cobertura es la probabilidad de detectar el fallo (Cobertura en la detección, en la localización, en el aislamiento, en la reconfiguración y en la recuperación

FTF

3. Detección del error

- Los fallos y las averías no pueden ser observados de forma directa, sino que deben ser deducidos a través de la presencia de errores.
- Como un error se define como el estado de un sistema, se pueden realizar comprobaciones para ver si existe un error o no.
- La efectividad de un esquema de tolerancia a fallos dependerá de la efectividad del mecanismo de detección de errores empleado.
- Idealmente, el mecanismo de detección de errores debe detectar cualquier posible error causado por aquellos fallos que el esquema de tolerancia a fallos intenta tratar. Sin embargo esto no es posible.
- **Comprobadores ideales**:
  - Se diseñan a partir de las especificaciones del sistema, sin tener en cuenta su diseño
  - Debe ser completo (todos los errores) y correcto (nunca avisa de un error inexistente)
  - Al comprobador no le deben afectar los fallos del sistema (es independiente del sistema)
- En los sistemas reales se utilizan los **comprobadores de aceptación**
  - **Objetivo:** Minimizar el coste y aumentar la cobertura de detección de errores

FTF

Tipos de comprobadores

- Comprobadores replicados
  - Se implementan sin necesidad de conocer la estructura interna del sistema replicado
  - Los resultados de los componentes diferentes se comparan o votan
  - Es uno de los métodos más caros
  - Si se supone que el diseño es correcto, que los fallos ocurren debido a causas físicas y que además son independientes se pueden replicar los componentes (TMR)
  - *Diversificación funcional* para evitar fallos de diseño
  - La replicación también se utiliza con propósitos de continuar con el servicio y de aislamiento del fallo en los sistemas distribuidos
- Temporizadores
  - Se utilizan para comprobar las restricciones temporales de los componentes
  - El desbordamiento del temporizador implica que las salidas del componente pueden ser incorrectas
  - Se utilizan en el hardware (*watchdog timers*) y en el software
  - En los sistemas distribuidos se utilizan para detectar el error en los nodos

FTF

Tipos de comprobadores (II)

- Comprobadores estructurales y de código
  - Para los datos, siempre existen los **comprobadores semánticos** (aseguran que el valor del dato es consistente con el resto del sistema) y los **comprobadores estructurales** (aseguran que la estructura interna del dato es correcta).
  - Si se utiliza redundancia en la información, entonces se utilizan los comprobadores estructurales.
  - El más común es la codificación (hw): Se utilizan bits extra cuyo valor depende de los datos.
- Comprobadores de la razonabilidad
  - Determinan si el estado de algún objeto en el sistema es "razonable".
  - La más habitual es la comprobación del rango de una variable, o la tasa de cambios de algún valor
  - El sistema también realiza comprobaciones de rangos en tiempo de ejecución (*run-time checks*)

    Por ejemplo en el acceso a las estructuras de datos se comprueba el rango del valor y el tipo
  - También se utilizan **aserciones** sobre el estado del sistema.

    Es una expresión lógica sobre el valor de diferentes variables en el sistema que será cierta si el estado del sistema es consistente

FTF

Tipos de comprobadores (III)

- Diagnosis
  - El sistema realiza comprobaciones en sus componentes para ver si funcionan correctamente.
  - Pueden ser valores especiales de entrada cuyas salidas son conocidas
  - El valor producido en la salida se compara con el correcto
  - Se llaman **autocomprobaciones** (*self-checks*) si las realiza el sistema al arrancar
  - Habitualmente requieren que el sistema esté parado, por eso su uso es limitado

FTF

4. Estimación y confinamiento de daños

- Siempre transcurre un tiempo, llamado latencia de detección del error, entre que se produce el fallo del componente y se detecta el error. Esto se debe a que el sistema no se monitoriza continuamente para detectar la existencia de errores.
- Debido a la interacción entre los componentes, el error se puede propagar
  - –OBJETIVO: Evaluar el alcance de la corrupción antes de pasar a corregir el sistema
- Para determinar el alcance de los daños se examina el flujo de información entre los componentes del sistema, para identificar aquellos en los que no ha habido comunicación y que por tanto seguro que están libres de errores.
- Método dinámico: Se almacena y examina el flujo de información (complejo)
- Método estático: Se utilizan cortafuegos para evitar que la información circule
- En muchos casos no se aplica esta fase, sino que se hacen estimaciones en la fase de recuperación sobre los posibles daños teniendo en cuenta la estructura del sistema

FTF

5. Recuperación del estado correcto

- OBJETIVO: Eliminar el estado erróneo del sistema para que no produzca una nueva avería del sistema en el futuro
- En caso de avería, el sistema se debe devolver a un estado consistente
- Técnicas:
  - Recuperación hacia atrás (*backward recovery)*

    El sistema se lleva a un estado anterior, que se supone que es correcto
    El estado del sistema debe almacenarse periódicamente (*checkpoint*)
    Es una técnica muy general y no depende en demasía de la naturaleza del fallo
    Muy útil en fallos transitorios
    Problemas: Supone un gasto de recursos importante
  - Recuperación hacia adelante (*fordward recovery)*

    Como no existe un estado anterior, se realizan las acciones correctoras pertinentes para devolver al sistema a un estado correcto
    Se necesitan conocer tanto el alcance de los daños como la naturaleza de los mismos para poder eliminar el error del sistema.
    Es imprescindible realizar una buena diagnosis de la razón de la avería
    Como la diagnosis depende del sistema y de la aplicación no se suele utilizar

FTF

6. Tratamiento del fallo y continuación del servicio

- En las fases anteriores se elimina el error del sistema (se detecta, se dimensiona y se elimina). Si este error es debido a un fallo transitorio ya se puede reinicializar.
- OBJETIVO: Identificar el componente averiado para que no sea utilizado en las computaciones posteriores
- Fases:
  - Localización del fallo: Se identifica al componente averiado
  - Reparación del sistema: El componente averiado ya no se utilizará o se hará de forma diferente

    La reparación se hace en línea y sin intervención manual

    Se realiza una *reconfiguración dinámica* del sistema de modo que la redundancia presente en el sistema se utiliza para realizar la tarea del componente averiado (ejem: sistema con repuesto)
- Una vez que se ha reparado el sistema continua el servicio. Los efectos de la TaF han sido únicamente una degradación de las prestaciones o una interrupción temporal del servicio.