El Problema de los Generales Bizantinos

Leslie LAMPORT, Robert SHOSTAK, Marshall PEASE

Ing. Pablo A. DEYMONNAZ - 9 de febrero de 2023

Objetivo

Implementar sistemas informáticos confiables.

Motivación Personal

Resumen

- Manejar componentes con mal funcionamiento que den información contradictoria.
- Se modela como Generales Bizantinos, frente a una ciudad enemiga.
- Generales se comunican a través de un mensajero, para acordar un plan de ataque.
- Problema: encontrar un algoritmo que asegure que los generales leales llegan a un acuerdo.

Introducción

- Parte defectuosa de un sistema: comportamiento incoherente
- Cada división del ejército comandada por un General, se comunican por mensajes y existen traidores.
- Los generales deben tener un algoritmo que asegure que:
  - **A)** Todos los Generales (leales) acuerdan un plan de acción, razonable.
  - **B)** Un número bajo de traidores no debe causar que los leales tomen un mal plan.

Introducción (cont.)

- Cada General comunica su observación *v(i)*, usa su método para combinar *v(1)*,...,*v(n)* y obtener el valor.
- A) Se logra con que todos apliquen el mismo método.
    - Todos los Generales deben recibir la misma información, los leales usan el mismo valor de *v(i)*. Si es leal, se usa su valor *v(i)*
- B) Usando un método robusto: ej: opinión de la mayoría (los traidores alteran la decisión si estaba dividida)

Problema de Generales Bizantinos

- El problema se re-escribe como un general que le da órdenes a sus n-1 tenientes. Con las **Condiciones de consistencia interactivas**:

  IC1) todos los tenientes leales obedecen la misma orden

  IC2) si el General es leal, cada teniente leal obedece la orden que él envía.

  El problema de los generales bizantinos parece engañosamente simple.

Resultados de imposibilidad

- La dificultad es que si los mensajes son orales, ninguna solución funciona a menos que haya más de ⅔ de Generales leales.
  => para 3 generales, no funciona ninguna solución si hay 1 traidor.
- **Mensaje oral:** está en control completo del emisor (los traidores pueden generar mensajes).

Consideramos los mensajes como dos posibilidades: "**atacar**" y "**retirarse**"

Comandante leal

El comandante leal envía la orden "atacar" a 2 tenientes. Teniente 2 es traidor y le dice al 1 que recibió retirarse. Para satisfacer IC2, teniente 1 debe obedecer atacar.

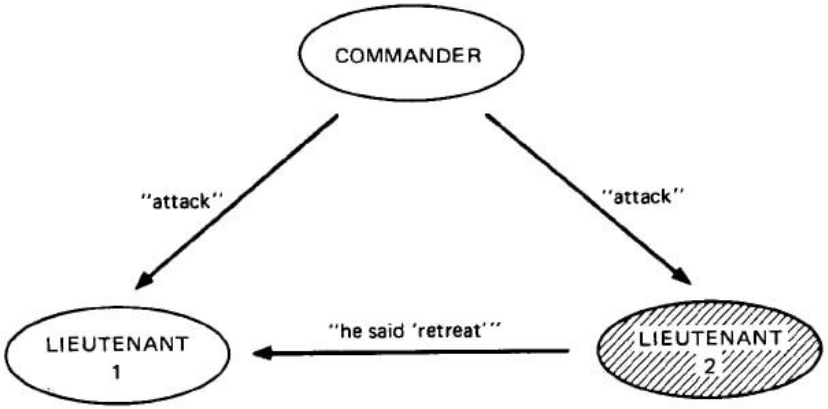


Fig. 1. Lieutenant 2 a traitor.

Comandante traidor

El comandante traidor envía la orden atacar a un "teniente" y "retirarse" al otro. El teniente 1 no sabe quién es el traidor y no puede saber qué mensaje le envió el general al teniente 2. Para el teniente 1, la situación es igual en las dos figuras. Debe obedecer “atacar”.
=> viola **IC1**

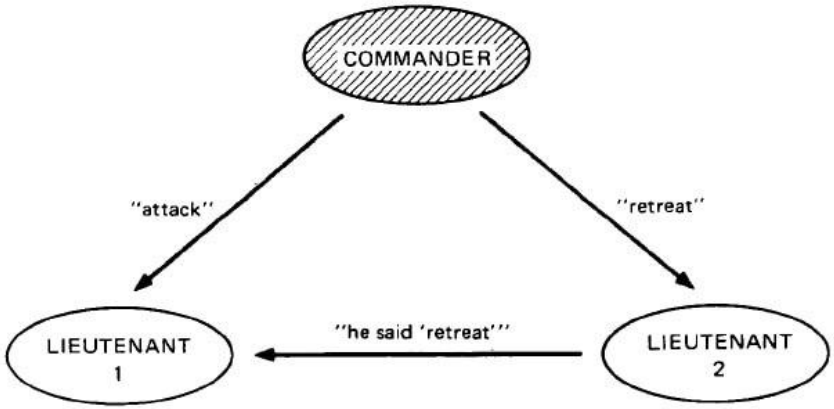


Fig. 2. The commander a traitor.

Resultados de imposibilidad

- No hay solución con menos de *3m+1* generales que pueda hacer frente a *m* traidores.

  Se prueba por contradicción.

Acuerdo aproximado

- Un acuerdo aproximado es tan difícil de alcanzar como el exacto.
- Se plantea acordar la hora del ataque.

- IC1': todos los tenientes leales atacan dentro de los 10 minutos unos de otros.
- IC2': si el comandante general es leal, entonces cada teniente leal ataca dentro de los 10 minutos de la hora dada por el comandante.

  Se demuestra con la decisión binaria y la restricción de los horarios.

Solución con Mensajes Orales

- Se presenta la solución para más de 3m+1 Generales.
- Cada general ejecuta un algoritmo que incluye el envío otros generales

  Asumimos que los leales ejecutan correctamente el algoritmo.

  A1) Cada mensaje enviado se transmite correctamente.

  A2) El receptor del mensaje conoce quién lo envió.

  A3) La ausencia de un mensaje puede ser detectada.

Solución con Mensajes Orales

- A1 y A2 evitan que un traidor interfiera en la comunicación entre dos generales:
  - A1 implica que no se pueden modificar mensajes y
  - A2 que un traidor no puede inyectar mensajes espurios
- A3 desincentiva a un traidor fuerce una decisión con la ausencia del envío de mensajes.
- Un general traidor puede decidir NO enviar mensajes. Dado que los tenientes leales deben obedecer, necesitan un comportamiento default. Por ejemplo: RETIRARSE.

Algoritmo OM(m) / m>=0

- Resuelve el problema con al menos *3m+1* Generales / a lo sumo *m* traidores.
- Lo describimos como tenientes "obteniendo un valor" (en lugar de “una orden”)
- Función "**mayoría**": si la mayoría de los valores *v_i* iguales a *v*, entonces *mayoría(v1,...,v_n-1)=v*
- Existen dos opciones naturales para el valor **mayoría**:

  1- el valor mayoría entre los *v_i*, si existe, sino el valor "retirarse".

  2- la mediana de *v_i*, asumiendo que vienen de un conjunto ordenado.

OM(o)

1. El general envía su valor a cada teniente.
2. Cada teniente usa el valor recibido del general o usa el valor "retirarse" si no recibe valor.

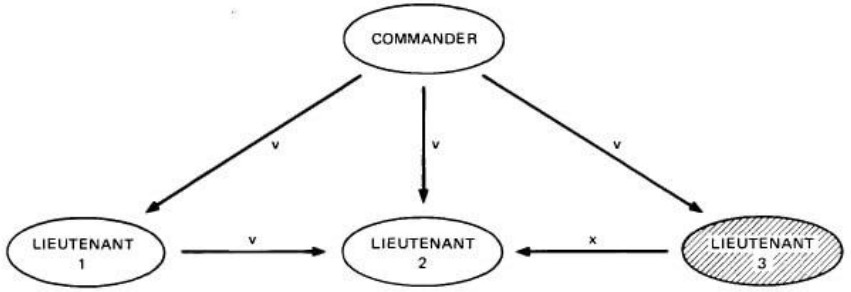
OM(m) m>0

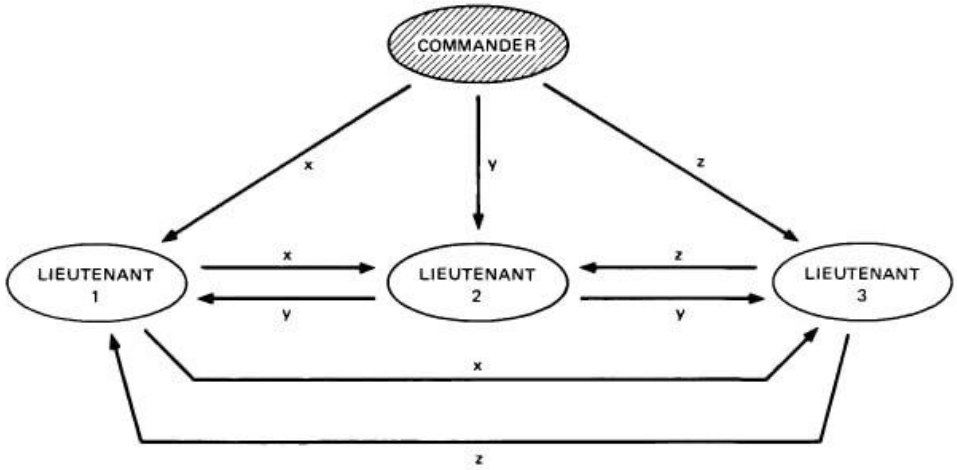
1. El general envía su valor a cada teniente.
2. Para cada *i*, sea *v_i* el valor que recibe el teniente *i* del General, o sino sea "retirarse" si no recibe valor. El teniente i actúa como el general en el algoritmo *OM(m-1)* enviándole el valor *v_i* a los *n-2* otros tenientes.
3. Para cada i, y cada j!=i, sea *v_j* el valor que el teniente i recibe del teniente j en el paso 2. o "retirarse" si no recibe valor. El teniente i usa el valor *mayoría(v1,...,vn-1)*.

OM(m)

Ejemplos con Comandante General leal y traidor.

El paper lo prueba por inducción.





Solución con Mensajes Firmados

- Evitamos que los traidores puedan mentir, impedimos que los Generales envíen mensajes falsificables.
- Agregamos la premisa:

  A4) a) La firma de un general leal no puede ser falsificada. Cualquier alteración al contenido de su mensaje firmado puede ser detectada.

  b) Cualquiera puede verificar la autenticidad de la firma del general.
- No se hacen premisas sobre la firma del general traidor => puede haber falsificaciones. Una firma puede ser falsificada por otro traidor.

Solución con Mensajes Firmados (cont.)

- Existe una solución para 3 Generales. Se plantea un algoritmo que hace frente al problema para *m* traidores y cualquier número de Generales.

- El general envía una orden firmada a cada uno de los tenientes. Cada teniente agrega su firma a la orden y la envía a los otros tenientes, que agrega su firma y la envía a los otros y así sucesivamente => cada teniente debe recibir efectivamente un mensaje firmado, hacer varias copias y enviar esas copias firmadas.

Solución con Mensajes Firmados (cont.)

- Función "***choice***" que es aplicada a un conjunto de órdenes para obtener una.
- Los requerimientos son:

    1. Si el conjunto V consiste de un solo elemento v => *choice(V) = v*.

    2. *choice(vacío) = "retirarse"*

- Una posible definición es permitir que *choice(V)* sea la *mediana* de V.

Algoritmo SM(m)

- *x:i* representa el mensaje firmado por el general *i*
- *v:j:i* representa el valor *v* firmado por *j*, luego el valor *v:j* firmado por *i*
- Cada teniente mantiene un conjunto *V_i*, conteniendo el conjunto de las órdenes firmadas correctamente recibidas hasta el momento (si el comandante es leal, el conjunto debe tener un sólo elemento).

Puede haber varios mensajes con la misma orden

Algoritmo SM(m) (cont.)

0) V_i = vacío.

1) el general firma y envía su valor a cada teniente.

2) para cada *i*:

A) si el teniente i recibe un mensaje de la forma *v:0* del comandante y él no había recibido ninguna orden, entonces:

i) asigna *V_i = {v}*

ii) envía el mensaje *v:0:i* a cada otro teniente

Algoritmo SM(m) (cont.)

B) Si el teniente recibe un mensaje de la forma: *v:0:j_i...:j_k* y *v* no está en el conjunto:

i) agrega *v* a *V_i*

ii) si *k* < *m*, envía el mensaje *v:0:j_i...:j_k:i* a cada uno de los otros tenientes que no están en el listado.

3) Para cada *i*: cuando el teniente i no recibe más mensajes (se puede usar timeout o contar los mensajes recibidos), obedece la orden ***choice(V_i)***

Algoritmo SM(m)

En este caso los tenientes pueden detectar que el general es traidor, porque su firma aparece en dos lugares.

Ambos ejecutan:

choice({“atacar”, “retirarse”})

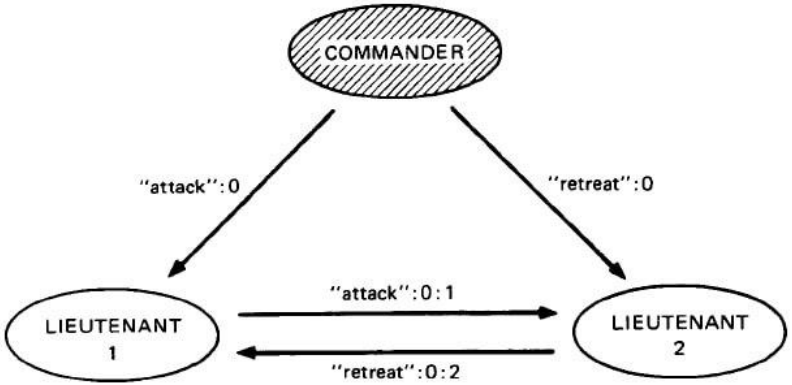


Fig. 5. Algorithm SM(1); the commander a traitor.

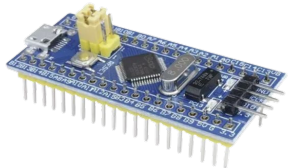
Sistemas confiables

La única forma que conocemos para construir un sistema informático confiable es usando diferentes "procesadores", para computar el mismo resultado y luego realizar una votación de mayoría para obtener un único valor.

Generales → Dispositivos de input

Tenientes → Procesadores

Ser leales → No fallan



Sistemas confiables - Premisas

Todos los procesadores *non-faulty* deben producir el mismo output => deben usar el mismo input

- El input puede provenir de un único componente que falla
- Pueden leer el valor mientras está cambiando

Condiciones

1) todos los procesadores correctos deben usar el mismo input

2) si la unidad de entrada no falla, todos los procesadores que no fallan deben usar el valor que provee como input (para generar el mismo output).

Sistemas confiables - Hardware

No se puede garantizar que dos procesadores (non-faulty) lean el mismo valor de input del mismo dispositivo que falla.

Excepto teniendo los procesadores comunicado entre sí resolviendo el problema de los Generales Bizantinos => Asegura que todos los procesadores usen el mismo valor de input, pero no asegura que tenga sentido.

Para valores importantes: se debe colocar dispositivos de entrada redundantes y que todos los procesadores usen los valores para producir el mismo output.

Las funciones *mayoría* y *choice* con la mediana generan valores razonables para un input non-faulty, aun cuando la lectura cambia.

Sistemas confiables - Pasaje de mensajes

A1) Cada mensaje enviado se transmite correctamente.

Las comunicaciones pueden fallar: no se puede distinguir si falla la comunicación o el procesador:

- OM(m) funciona hasta m fallas.
- SM(m) funciona con fallas de comunicación (una falla no falsifica la firma del mensaje)

Sistemas confiables - Pasaje de mensajes

A2) El receptor del mensaje conoce quién lo envió.

Un procesador que falla no pueda impersonar a uno que no falla.

En la práctica, esto significa que la comunicación IPC sea sobre líneas fijas en lugar que una línea switching.

Sistemas confiables - Pasaje de mensajes

A3) La ausencia de un mensaje puede ser detectada.

Se detecta con la falta de recepción de un mensaje.

Requiere:

- Tener relojes sincronizados, con una diferencia mayor a *T*.
- El procesamiento de mensajes está acotado en *u*.

El delay máximo para SM(m) es: $T_0 + k\,(u + T)$
(k cantidad de firmas del mensaje)

Sistemas confiables - Pasaje de mensajes

A4) Firma del mensaje. Función firma debe cumplir:

- Si el procesador *i* es *non-faulty*, un procesador faulty no puede generar *S_i(M)*. Esto no se puede asegurar, pero se puede asegurar con una probabilidad de confianza.
- Dado *M* y *X*, cualquier proceso puede determinar si *X = S_i(M)*

Casos de interés:

- Mal funcionamiento random: faulty genera una firma válida
- Inteligencia maliciosa: (e.g. procesador con malware). Problema de falsificar *S_i* es criptográfico. Evitar ataque por repetición con un *nonce*.

Conclusiones

- Presentó soluciones al problema de los Generales Bizantinos que se aplican a construir sistemas confiables.
- Son “costosas” en cantidad de mensajes y en tiempo.
- En OM(m) y SM(m) cada teniente debe esperar a que el mensaje generado por el General sea reenviado por otros m tenientes.

Conclusiones

- Alcanzar confiabilidad frente a un mal funcionamiento arbitrario es un problema difícil y su solución parece ser inherentemente cara.
- La forma de reducir el costo es hacer premisas acerca del tipo de fallas (ej: una computadora puede fallar en responder, pero nunca responder mal).
- Cuando se requiere extrema confiabilidad NO se pueden hacer las presunciones y se debe pagar el costo del Problema de los Generales Bizantinos.

Preguntas?

Muchas gracias

https://lamport.azurewebsites.net/pubs/byz.pdf