Programa

1. Redes de dados e programação da comunicação distribuída (revisão)
2. RPC (*Remote Procedure Call*),
   RMI (*Remote Method Invocation*),
   Web Services
3. Gestão de Nomes
4. **Segurança**
   Canais seguros
   Autenticação
   Autorização
5. Tolerância a Faltas
   Replicação
   Transações

Modelos fundamentais

- Explicitam quais são as entidades e características essenciais de um sistema
- Permitem-nos:
  1. Generalizar o que é possível e impossível resolver nesse modelo
     - Por provas matemáticas
  2. Desenhar soluções mais facilmente
     - Pois não pensamos nos detalhes de hardware, etc

Modelos fundamentais

- Explicitam quais são as entidades e características essenciais de um sistema
- Permitem-nos:
  3. Provar matematicamente propriedades das nossas soluções
     - Fiabilidade, desempenho, escalabilidade, segurança
  4. Determinar facilmente se determinada solução funciona num sistema em particular
     - Basta verificar se os pressupostos do modelo usado para a solução se verificam no sistema em particular!

Modelos fundamentais

- Logo, antes de desenhar qualquer solução, é muito boa prática definir os modelos fundamentais!

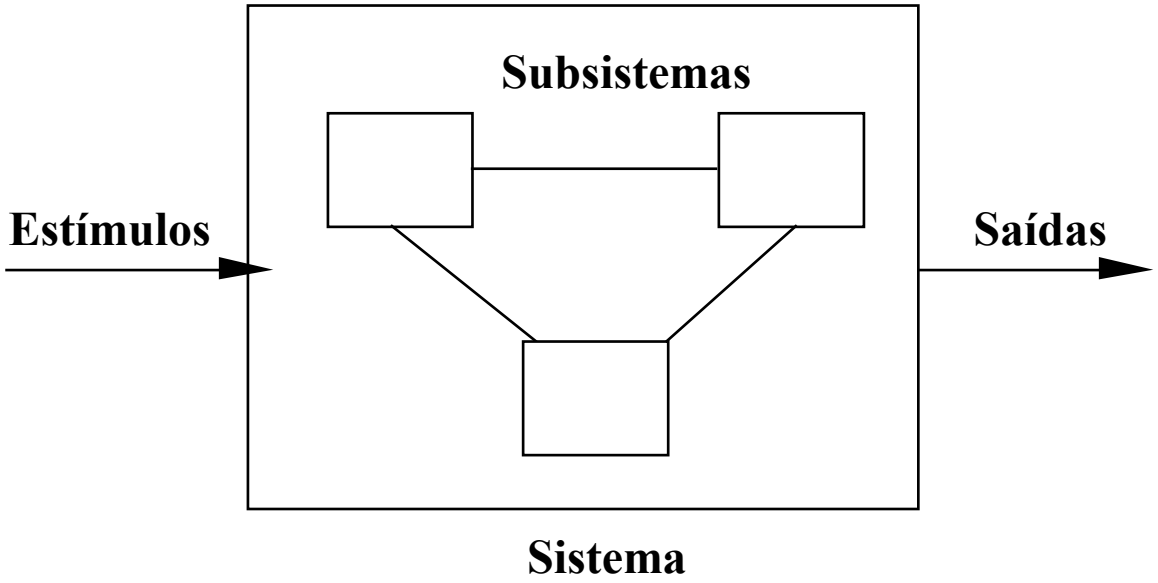
- Três modelos fundamentais:
  - Modelo de Interação
  - Modelo de Segurança
  - Modelo de Faltas

Tolerância a Faltas

Tolerância a Faltas

Terminologia de base

Sistema Computacional



Um **sistema** tem uma especificação funcional do seu comportamento que define, em função de determinadas entradas e do seu estado, quais são as saídas.

Sistema Computacional

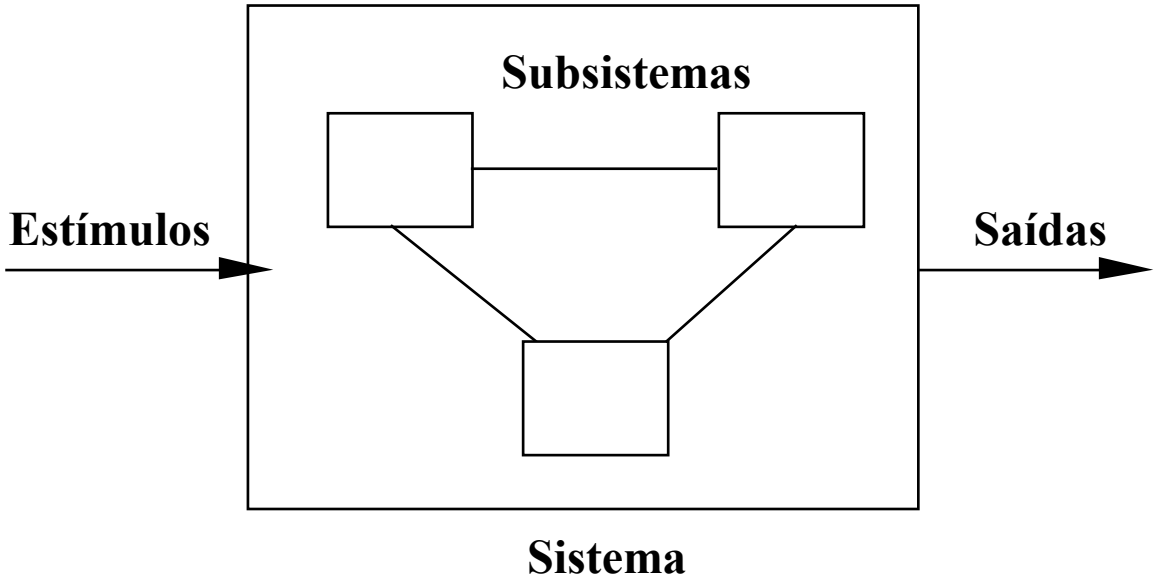
Sistema computacional:

- Formado por um conjunto de componentes internas
- Tem um estado interno
- Sujeito a um conjunto de entradas, ou estímulos externos
- Tem um determinado comportamento
  - Produz resultados em função das entradas e do seu estado interno

Comportamento:

- Especificado
- Observado
  - Serviço cumprido
  - Serviço interrompido

Sistema Computacional



Sistema determinístico
se as saídas e o estado seguinte forem uma função (determinística) dos estímulos e do estado actual

Falta, Erro, Falha

Falta (*fault*):

- Acontecimento que altera o padrão normal de funcionamento de uma dada componente do sistema

Erro (*error*):

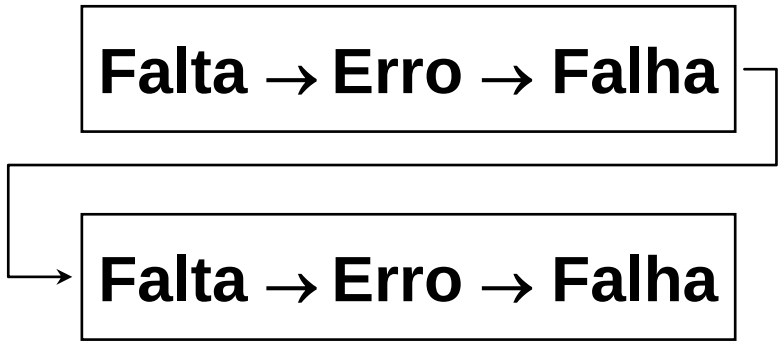
- Transição do sistema, provocada por uma falta, para um estado interno incorreto
- Estado interno inadmissível
- Estado interno admissível, mas não o especificado para estas entradas

Falha (*failure*):

- Quando se desvia da sua especificação de funcionamento
- Num determinado estado, o resultado produzido por uma dada entrada não corresponde ao esperado

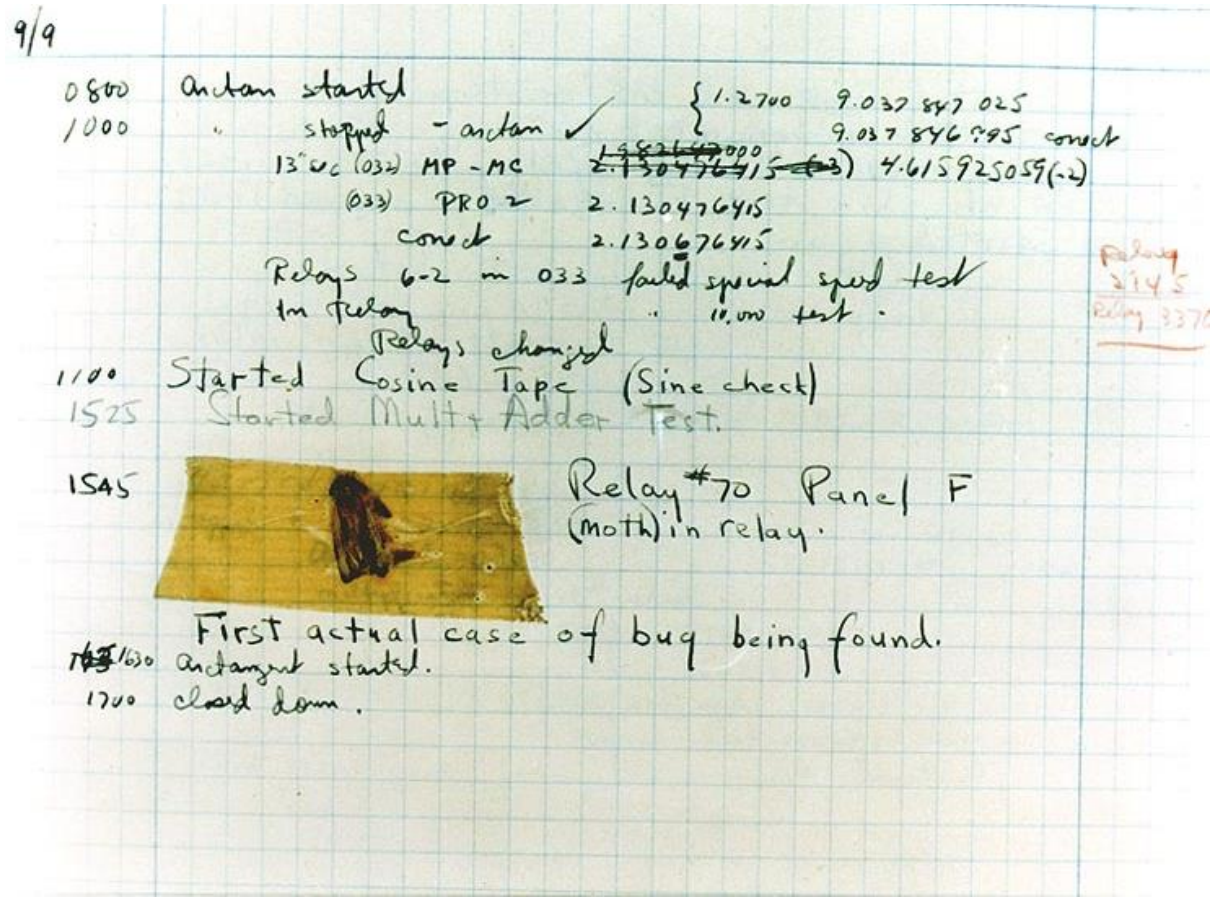
Falta->Erro->Falha

- Exemplo:
  - Falta: cabo de alimentação desligado
  - Erro: o processador (e restantes componentes) não funcionam
  - Falha: o computador não arranca
- Falta: a causa de um erro é uma falta
- Erro: uma falha ocorre devido a um erro
- Falha: desvio do comportamento especificado



Exemplo de Falta: o primeiro bug

Mark II, general view of calculator frontpiece, 1948.



Exemplo: *bug software*

Falta:
- Engano de um programador ao definir a lógica do programa colocando uma instrução errada

Erro:
- Execução da instrução errada
- Erro fica latente até se manifestar uma falha do programa (ex.: dado incorreto na base de dados, variável com o valor errado, etc.)

Falha:
- O erro torna-se efectivo e o programa falha

Exemplo: bit de memória preso (*stuck to one*)

Falta:

- Posição de memória, em que um bit fica sempre com o valor 1
- Falta latente pois não dá origem a erro se:
  - Esta posição de memória não for utilizada
  - Se não for escrito um 0 naquele bit

Erro:

- Escrita de um octeto com o bit a 0
- Detectado pelo bit de paridade erro
- Possível evolução :
  - Erro processado (ex: código corrector de erros da memória) => Falta foi tolerada
  - Erro não processado => Erro fica latente até esta posição de memória ser lida

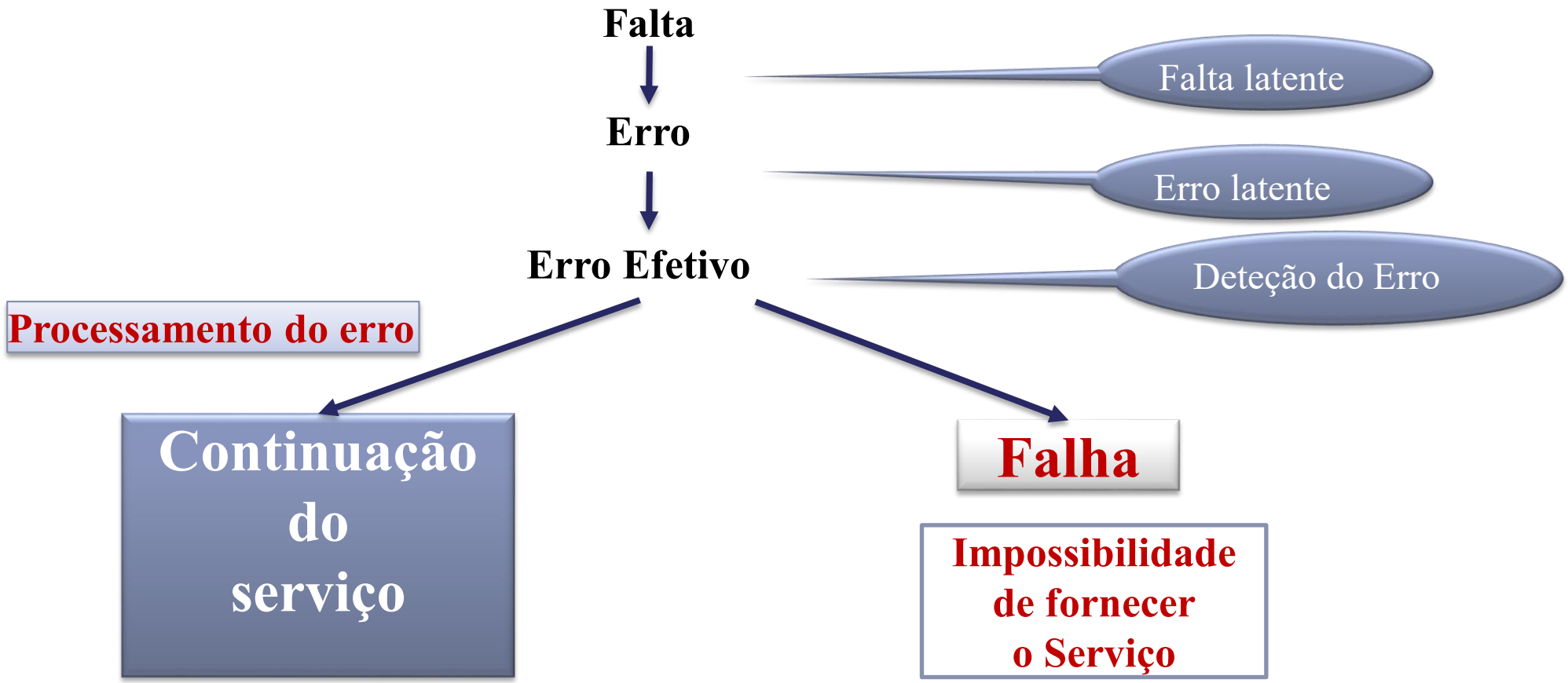
Falha:

- Leitura de um valor incorreto da posição de memória
- O erro torna-se efetivo e o sistema de memória falha, não funciona de acordo com o especificado

Políticas de Tolerância a Faltas

- Qualquer política de tolerância a faltas baseia-se na existência de um **mecanismo redundante** que possibilite que a função da componente comprometida seja obtida de outra forma
- A **redundância** pode assumir diversas formas:
  - **Física** ou **espacial**, com duplicação de componentes
  - **Temporal**, com repetição da mesma ação
  - **Informação** com algoritmos que calculam um estado correto

Modelo de base da Tolerância a Faltas



Erros Latentes e Efetivos

- Latência de um erro
  - Tempo que decorre entre a ocorrência de um erro e a falha correspondente
- Consequência:
  - Latente: ainda não provocou a falha, ainda não foi detetado
  - Efetivo: foi detetado e se não for tratado pode causar a falha

Modelos fundamentais

Interação e Faltas

Modelo de Interação

O que pressupomos sobre o canal de comunicação?

- Latência, que inclui:
  - Tempo de espera até ter acesso à rede +
  - Tempo de transmissão da mensagem pela rede +
  - Tempo de processamento gasto em processamento local para enviar e receber a mensagem
- Largura de banda
  - Quantidade de informação que pode ser transmitida simultaneamente pela rede

Modelo de Interação

O que pressupomos sobre o canal de comunicação? (cont.)

- Canal assegura ordem de mensagens?
- Mensagem pode chegar repetida?
- *Jitter*
  - Que variação no tempo de entrega de uma mensagem é possível?

E sobre os relógios locais?

- Taxa com que cada relógio local se desvia do tempo absoluto

Modelo de Interação: sistemas síncronos vs. assíncronos

- Sistema **síncrono** é aquele em que são garantidos os seguintes limites:
  - Cada mensagem enviada chega ao destino dentro de um tempo limite conhecido
  - O tempo para executar cada passo de um processo está entre limites mínimo e máximo conhecidos
  - A taxa com que cada relógio local se desvia do tempo absoluto tem um limite conhecido
- Caso algum destes limites não seja conhecido, o sistema é **assíncrono**

Modelo de Interação: Sistemas síncronos vs. assíncronos

- Estimar valores **prováveis** para os limites anteriores é normalmente fácil…
- Mas conhecer limites **garantidos** nem sempre é possível!
  - Exemplos:
    - Quanto tempo pode demorar até um *email* chegar?
    - Quanto tempo pode demorar até concluir transferência de ficheiro por FTP?
- Qualquer solução desenhada para sistema assíncrono é correta em sistema síncrono
  - E o inverso, é verdade?

Exemplo/Desafio

Exemplo/Desafio

- 2 divisões de um exército no topo de 2 colinas
- Objetivo: atacarem em simultâneo o inimigo
  - Ordem de ataque é dada por uma das divisões
    - Por exemplo, a mais numerosa
  - Comunicação entre ambas é por mensageiro
- Solução?
  - Caso os mensageiros demorem entre *min* e *max* a ir de uma colina a outra (sistema síncrono)
  - Caso não seja possível conhecer os limites de tempo dos mensageiros (sistema assíncrono)

Deteção de faltas de omissão de processos

- O que pode cada divisão de exército saber sobre a outra divisão na outra colina?
  - Num sistema assíncrono
  - Num sistema síncrono

- É possível detetar que a outra divisão já foi derrotada?
- É possível saber se a outra divisão está viva neste momento?

Modelo de Faltas

- Que componentes podem falhar?
- De que forma podem falhar?

Modelo de Faltas num Sistema Distribuído

- Num sistema distribuído o modelo de faltas é muito mais complexo que num sistema centralizado.
  Várias componentes do sistema podem falhar:
  - Faltas na comunicação
  - Faltas nos nós
    - Processadores/Sistema
    - Processos servidores ou clientes
    - Meios de Armazenamento Persistente

Vamos considerar que todas estas faltas provocam a falha do nó

Tipos de Faltas

- Classificadas por:
  - Causa
  - Origem
  - Duração
  - Independência
  - Determinismo

Tipos de Faltas

Causa

- Falta Física: fenómenos elétricos, mecânicos, …
- Falta Humana
  - Acidental: conceção, operação, …
  - Intencional: ataque premeditado (consideradas no capítulo de Segurança)
  - *Estudo de 2003 sobre falhas em serviços na Internet aponta os erros humanos (operação) como a principal causa de faltas*

Origem

- Falta Interna: componentes internos, programa, …
- Falta Externa: temperatura, falta de energia, …

Tipos de Faltas

Duração

- **Faltas Permanentes**: mantêm-se enquanto não forem reparadas
  (ex.: cabo de alimentação desligado)
  - Fáceis de detetar
  - Difíceis de reparar
- **Faltas Temporárias ou Transientes**: ocorrem apenas durante um determinado período, geralmente por influência externa
  - Difíceis de reproduzir, detetar
  - Fáceis de reparar
  - As faltas transientes ficam reparadas imediatamente após terem ocorrido
    (ex.: perda de mensagem)

Exemplo: Faltas na Comunicação

Falta temporária ou transiente

- Normalmente resolvida por
  - Protocolos de transporte com tratamento de erros - TCP
  - RPC com semânticas: pelo-menos-uma-vez, no-máximo-uma-vez

Falta permanente

- Impossível de recuperar sem redundância física – redes malhadas, cablagens duplas
- O protocolo IP procura resolver este problema se a rede tiver redundância física

Tipos de Faltas

Independência

- **Faltas independentes:**
  - Probabilidade de falta de uma componente é independente das outras componentes
  - Em geral, boa aproximação no *hardware*
- **Faltas dependentes:**
  - Probabilidades de falta correlacionadas
  - Exemplos:
    - Faltas no *software*, se for idêntico em várias máquinas
    - Múltiplos componentes *hardware* a correr no mesmo local, sujeitos à mesmas faltas externas
      - Incêndios, falhas de energia, roubos, etc.

Tipos de Faltas

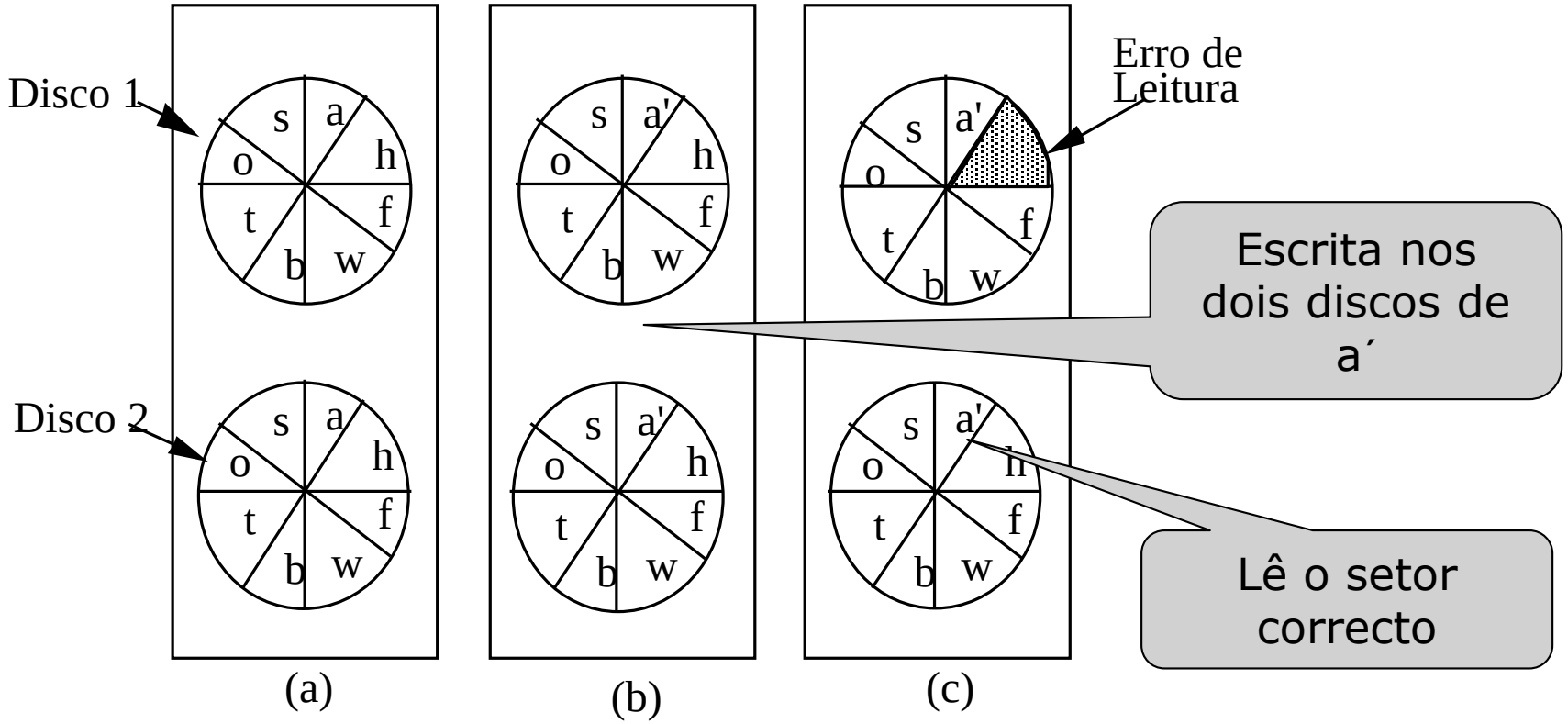
Determinismo

- **Faltas Determinísticas**:
  - Dependem apenas da sequência de entradas (*inputs*)
  - Repetindo essa sequência, reproduzimos a falta
- **Faltas Não-Determinísticas (“*Heisenbugs*”)**:
  - Dependem de outros fatores
    (e.g., escalonamento de *threads*,
    leituras do relógio, ordem de entrega de mensagens)
  - Difíceis de reproduzir, depurar

Tipos de Faltas

- **Faltas** por omissão ou **silenciosas**
  - Quando componente pára e não responde a nenhum estímulo externo
- **Faltas** arbitrárias ou **bizantinas**
  - Pior caso possível,
    em que qualquer comportamento do componente é possível

Falta silenciosa

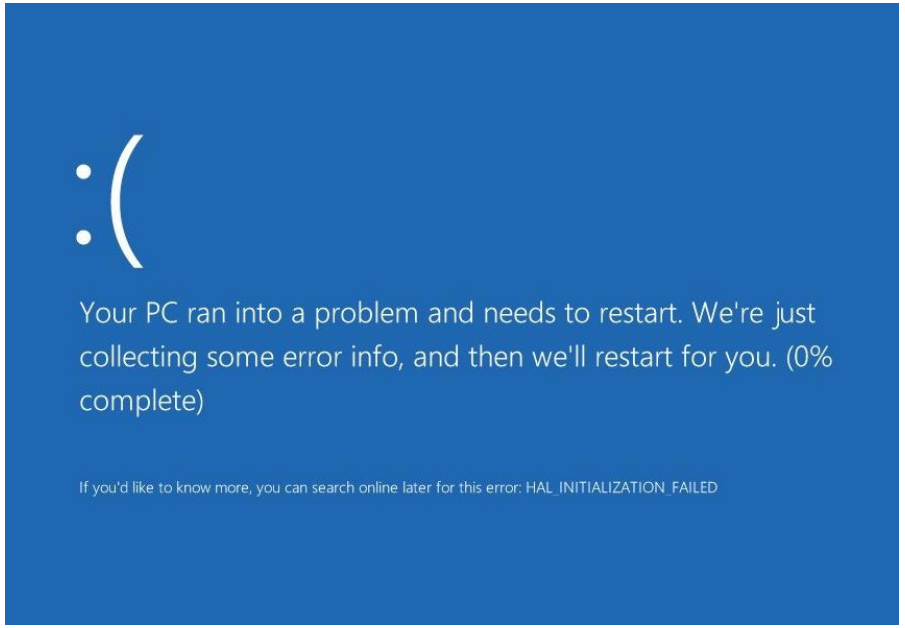


A falta de um setor é evidenciada por um erro de leitura. Se o modelo considera 1 falta silenciosa o setor correspondente no 2º disco está correto.

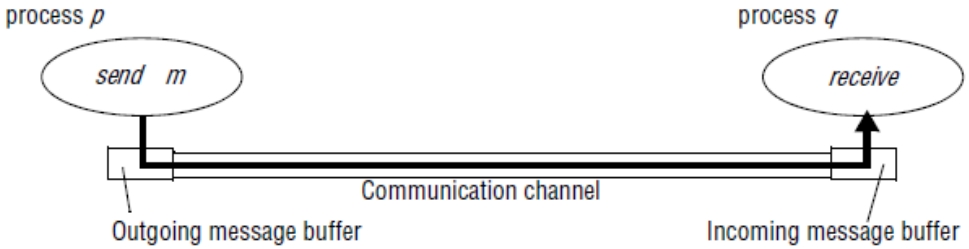
Faltas silenciosas dos processos

- Quando um processo em falha deixa de responder a estímulos do exterior

- Pode ser detetável
  - *Fail-stop failure*
- Ou não detetável
  - *Crash failure*



Faltas silenciosas do canal



- Quando o canal não entrega uma mensagem
- Podemos distinguir entre:
  - *Send-omission failure*
    - Mensagem perdeu-se entre o processo emissor e o *buffer* de saída para a rede
  - *Channel-omission failures*
    - Mensagem perdeu-se no caminho entre ambos os *buffers*
  - *Receive-omission failure*
    - Mensagem chegou ao *buffer* de entrada do recetor mas perdeu-se depois

Faltas arbitrárias (bizantinas)

- De um processo
  - Processo responde incorretamente a estímulos
  - Processo responde mesmo quando não há estímulos
  - (Processo não responde a estímulos)
- Do canal
  - Mensagem chega com conteúdo corrompido
  - Entrega mensagem inexistente ou em duplicado
  - (Não entrega mensagem)
- Faltas arbitrárias do canal são raras
  pois os protocolos sabem detetá-las
  - Como? O que fazem quando as detetam?

Faltas frequentemente não consideradas no modelo

Falta densa

- Acumulação de tantas faltas toleráveis que deixa de ser tolerável

Falta arbitrária (bizantina)

- Faltas que fogem ao padrão de comportamento especificado para a componente, por exemplo, um nó da rede que envia mensagens corretas a um interlocutor e erradas a outro

Faltas de temporização

- Algum dos pressupostos de tempo de um sistema síncrono deixa de ser garantido
  - Ex.: mensagem tem um atraso superior a Tmax,
    relógios desfasados,
    servidor sobrecarregado responde depois de Tmax
- Não fazem sentido em sistemas assíncronos

Modelo de Faltas: Cobertura e catástrofes

- Depois de definido o modelo de faltas, podemos decidir:
  - Quais as faltas que serão toleradas
  - Quais as que não serão toleradas
- **Taxa de cobertura:** relação entre as faltas que serão toleradas e o conjunto de faltas previsíveis
- As faltas que originam erros sem possibilidade de tratamento dão origem a **catástrofes**

Modelo de Faltas: Cobertura e catástrofes

Faltas que é vulgar considerar

- Pela simplificação que introduzem nos algoritmos é muitas vezes assumido que a falta é **silenciosa** sem que haja real demonstração que é assim de facto

Tipos de faltas que é vulgar **não** considerar no modelo:

- **Faltas densas** - resultam da acumulação de faltas, não permitindo o seu tratamento porque são superiores à redundância do sistema ou à sua capacidade de manutenção
- **Faltas arbitrárias (bizantinas)**

Políticas de tolerância a faltas

Políticas de Tolerância a Faltas

Recuperação do erro

- Substitui um estado errado por um estado correto, podendo tornar sem efeito algumas etapas do processamento já efetuado.
- Esta política implica:
  - Deteção do erro
  - Cálculo de um estado anterior ou posterior correto
- Durante o tempo de recuperação o sistema fica indisponível, afetando a disponibilidade

Políticas de Tolerância a Faltas

Compensação do erro

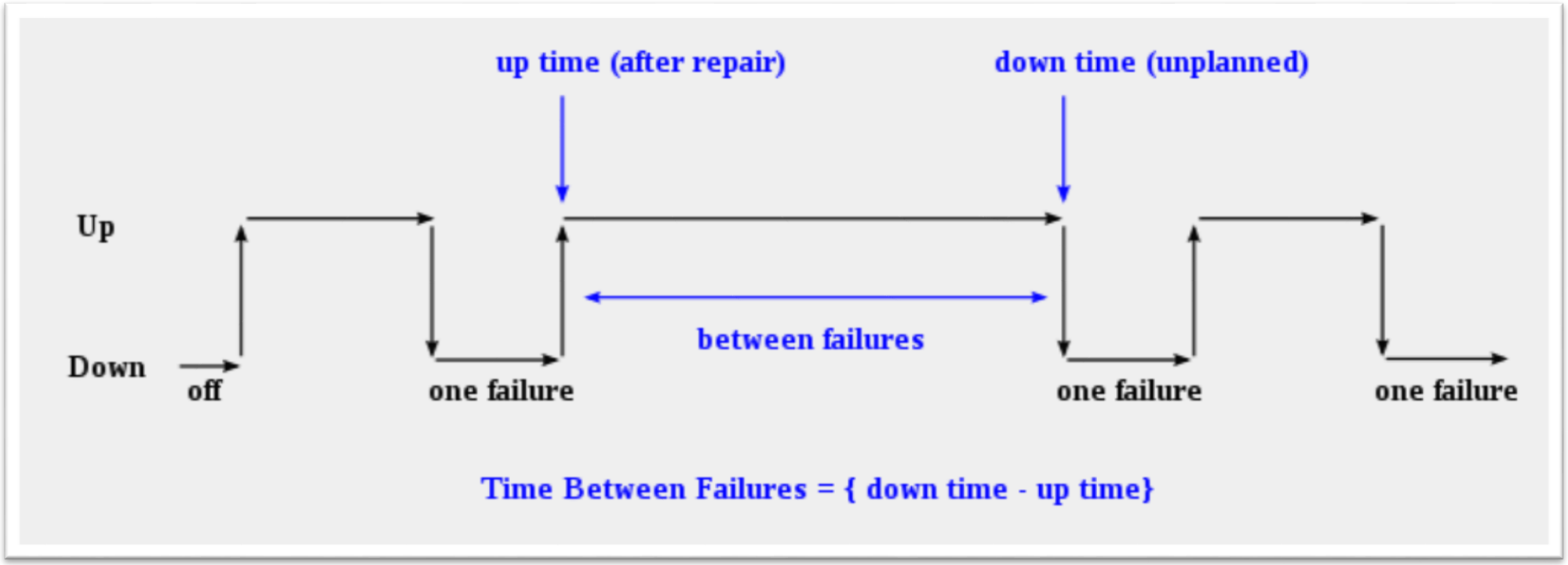
- Calcula um estado correto a partir de componentes redundantes
  - A arquitetura do sistema tem de possuir redundância suficiente para ser capaz de computacionalmente definir o estado correto, apesar de um estado interno errado
  - Esta abordagem procura limitar ou eliminar o período de recuperação, ou seja, maximizar a disponibilidade do sistema
- As duas políticas podem ser usadas em conjunto

Medidas usadas em tolerância a faltas

Fiabilidade, Disponibilidade

Fiabilidade (*reliability*)

- Mede o tempo médio desde o instante inicial até à próxima falha
- **MTTF** (*Mean Time To Failure*): medida estatística da fiabilidade
  - É o critério fundamental se o sistema não for reparável
- **MTBF** (*Mean Time Between Failures*): define a fiabilidade para sistemas reparáveis

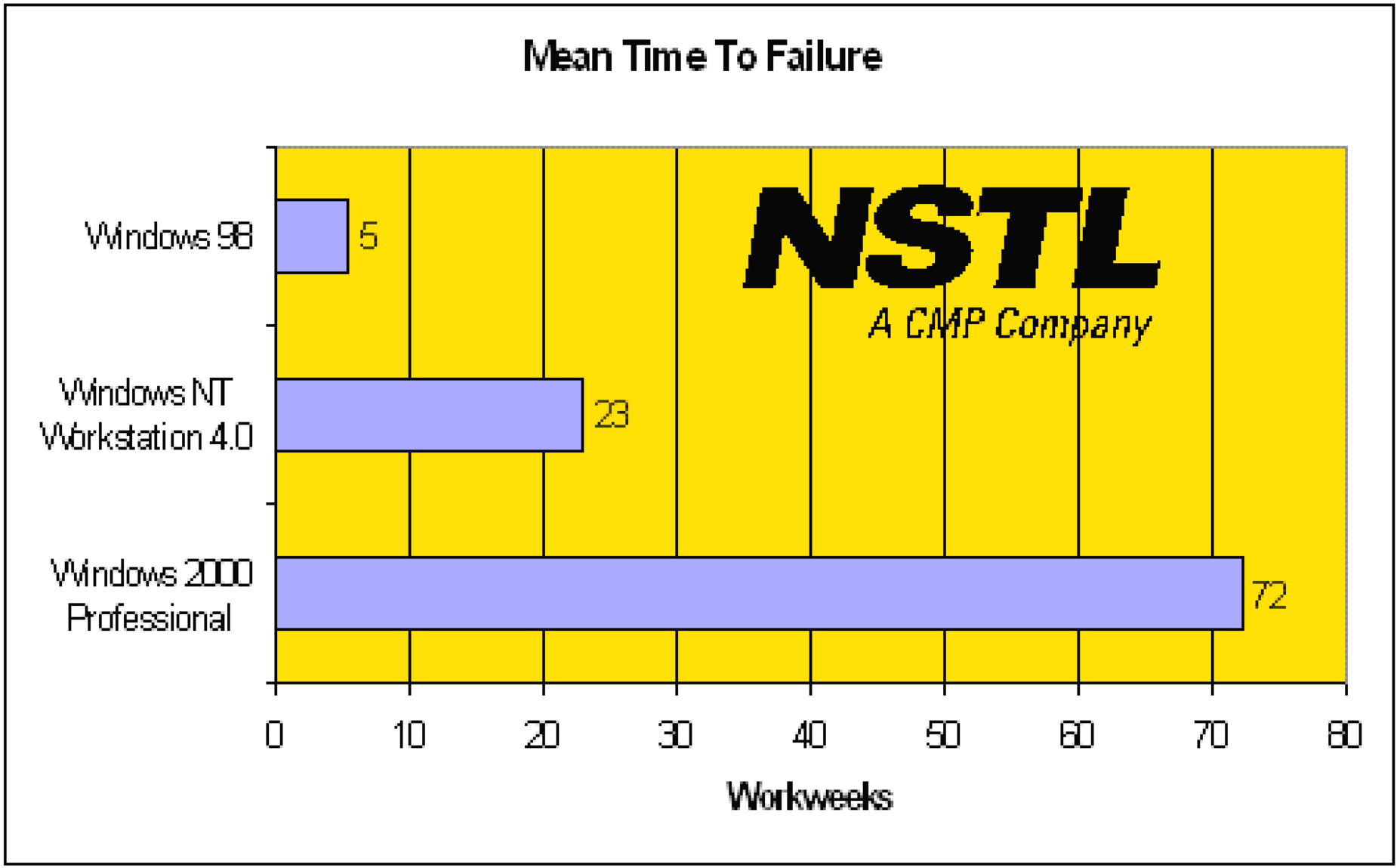

up time (after repair)
down time (unplanned)
Up
Down
off
one failure
between failures
one failure
one failure
Time Between Failures = { down time - up time}

Fiabilidade, Disponibilidade

Disponibilidade (*availability*)

- Mede a relação entre o tempo em que um serviço é fornecido e o tempo decorrido
  - **MTTR** (*Mean Time to Repair*): medida estatística da interrupção de serviço
- **Disponibilidade** = MTBF / (MTBF + MTTR)

Level of availability	Availability (percent)	Downtime per year
Commercial or standard	99.5	43.8 hours
Highly available	99.9	8.75 hours
Fault resilient	99.99	53 minutes
Fault tolerant	99.999	5 minutes
Continuous	100	0 minutes


Mean Time To Failure
NSTL
A CMP Company
Windows 98
5
Windows NT Workstation 4.0
23
Windows 2000 Professional
72
0
10
20
30
40
50
60
70
80
Workweeks

Classes de Disponibilidade

Tipo	Indisponibilidade (min/ano)	Disponibilidade	Classe
Não gerido	52 560	90%	1
Gerido	5 256	99%	2
Bem gerido	526	99.9%	3
Tolerante a faltas	53	99.99%	4
Alta disponibilidade	5	99.999%	5
Muito alta disponibilidade	0.5	99.9999%	6
Ultra disponibilidade	0.05	99.99999%	7

D: Disponibilidade
(também chamado “número de noves de disponibilidade”)

Classe de Disponibilidade = $\log_{10} [1 / (1 - D)]$

Exemplos de Classes de Disponibilidade

- Especificações existentes:
  - Classe 5: equipamento de monitorização de reatores nucleares
  - Classe 6: centrais telefónicas
  - Classe 9: computadores de voo