Navigators FCUL

Serviços Distribuídos Tolerantes a Intrusões

resultados recentes e problemas abertos

Miguel Pupo Correia
mpc@di.fc.ul.pt
www.di.fc.ul.pt/~mpc

LASIGE, Universidade de Lisboa

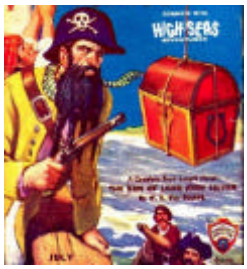
V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais
Florianópolis, Setembro de 2005

Tolerância a Intrusões?

- **Segurança**
  - ☞evitar que pessoas com intenções maliciosas prejudiquem o funcionamento de um sistema
- **Confiança no funcionamento (CnF)**
  - ☞garantir que o sistema se comporte de acordo com a sua especificação
  - ☞inclui Tolerância a Faltas
- **Tolerância a Intrusões (TI)**
  - ☞*aplicar o paradigma da tolerância a faltas no domínio de segurança*

Uma estória

Final (in)feliz



Segurança vs CnF

- Segurança:
  - ☞ênfase na prevenção: muralha, guardas, jacarés, ... – firewalls, controle de acesso, cripto...
- Confiança no Funcionamento:
  - ☞procura que o sistema continue operacional mesmo que alguns componentes falhem
  - ☞se o computador de bordo do avião falhar...
- Tolerância a Intrusões:
  - ☞*aplicar o paradigma da tolerância a faltas no domínio de segurança*

Aplicando TI à estória

redundância
diversidade

Navigators FCUL

A área da TI

- Conceito com 20 anos:
  - Joni Fraga e David Powell. A fault- and intrusion-tolerant file system. *Proc. Int'l Conf. on Computer Security*, 1985
- Por volta de 2000:
  - projecto europeu MAFTIA
  - programa americano OASIS (projectos ITUA, SITAR, PASIS, AgileStore...)

Navigators FCUL

O minicurso

- A TI é muito vasta
  - p.ex. podia-se dizer que a *detecção de intrusões* é um mecanismo de TI (mas a origem é diferente e tem mérito próprio!)
- Trabalho mais interessante dos últimos anos: **serviços distribuídos TI** (minha opinião claro...)
- *Objetivo:*
  garantir integridade, disponibilidade e confidencialidade de **serviços** constituídos por diversos servidores ligados por uma rede
  mesmo que alguns servidores sejam atacados e controlados por atacantes ou código nocivo

Navigators FCUL

Serviços distribuídos TI

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

Confiança no Funcionamento

Navigators FCUL

Conceito de sistema

- entidade que interage com outros sistemas (computacionais, mecânicos, seres humanos) através da sua fronteira
- tem diversos componentes
- caracterizado pela sua funcionalidade e por um conjunto de propriedades não funcionais: desempenho, segurança, confiabilidade,...
- fornece um serviço através da sua interface
- tem um estado

Nomenclatura em português: R. de Lemos e P. Veríssimo

Navigators FCUL

Processo de falha

- sistema oferece um serviço correto se obedece à sua especificação;
- caso contrário **falha** – o que queremos evitar!
- **falta**: causa remota de uma falha
  - interna – p.ex. defeito numa RAM
  - externa – p.ex. operador tropeça e desliga cabo
- **erro**: consequência de uma falta no estado
  - p.ex. registo corrompido devido a defeito na RAM
- falta ? erro ? falha

Navigators FCUL

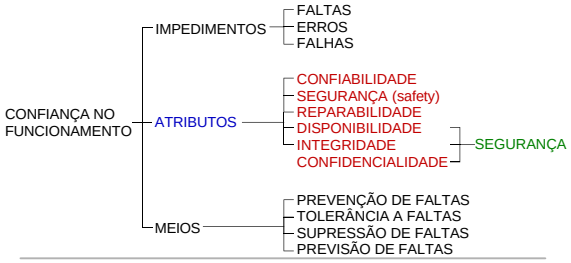
Processo de falha (cont)

- falta dormente: não gera (logo) um erro
- erro pode não causar (logo) falha
- Falta interna corresponde a falha de um componente do sistema:
  - falta ? erro ? falha ? falta ? erro ? falha ...

Navigators FCUL

Atributos de CnF

- Objetivo da CnF é garantir um conjunto de atributos de um sistema apesar da ocorrência de faltas:



Navigators FCUL

Meios/mecanismos de CnF

- **Prevenção de faltas** - meios para prevenir a introdução de faltas. Parte da engenharia de sistemas.
- **Tolerância a faltas** - meios para evitar a falha do serviço quando ocorrem faltas. Duas sub-categorias:
  - **mascaramento de faltas** - usar redundância para garantir que as faltas não causem a falha do sistema;
  - **detecção e processamento** - detectar a ocorrência de erros e processá-los de forma a os neutralizar.
- **Supressão de faltas** - meios usados durante o projeto para reduzir o número e severidade das faltas. Técnicas de verificação e validação de sistemas.
- **Previsão de faltas** - meios para estimar o número de faltas no sistema, e prever o número e consequências de faltas futuras. Modelagem e teste.

Navigators FCUL

Tolerância a Intrusões

Navigators FCUL

Tolerância a Intrusões

- O que é *aplicar o paradigma da tolerância a faltas no domínio de segurança?*
  - ☞assumir e aceitar que o sistema permanece sempre mais ou menos vulnerável;
  - ☞assumir e aceitar que os componentes do sistema podem ser atacados e que alguns desses ataques terão sucesso;
  - ☞garantir que o sistema como um todo permanece seguro e operacional, ou seja, que não falha.

Navigators FCUL

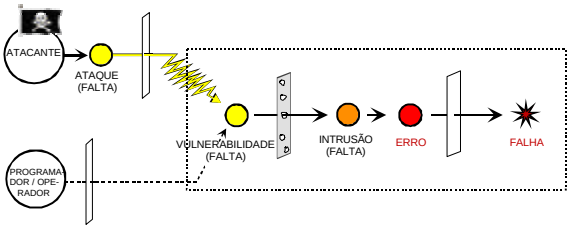
Conceitos de TI

- **vulnerabilidade**: falta de projeto ou de configuração, geralmente acidental, que pode ser explorada com fins maliciosos
- **ataque**: falta intencional, maliciosa, que visa explorar uma ou mais vulnerabilidades
- **intrusão**: resultado de um ataque que tem sucesso em explorar uma ou mais vulnerabilidades
- Estas faltas são englobadas na categoria mais geral: faltas arbitrárias ou bizantinas
  - ☞ L. Lamport et al., The Byzantine Generals Problem, 1982

Navigators FCUL

Modelo AVI

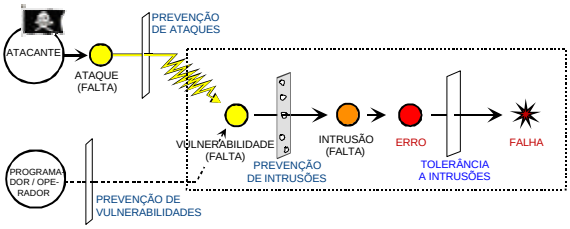
MAFTIA



processo de falha

Navigators FCUL

Modelo AVI (cont)



processo de falha; meios para a evitar

Navigators FCUL

Mais conceitos

- trust
  - ☞dependência de um componente em relação às propriedades de segurança de outro componente
  - ☞pode-se substituir "segurança" por confiança no funcionamento se considerarmos os atributos extra: confiabilidade, safety, reparabilidade
- trustworthiness = confiança no funcionamento
  - ☞se considerarmos também esses atributos

Navigators FCUL

Meios de CnF e serv.dist.TI

- O projecto de serviços distribuídos TI envolve os quatro meios de CnF referidos:
- Tolerância a faltas:
  - ☞a maior parte das soluções que veremos usam mascaramento de faltas: redundância de máquinas + protocolos tolerantes a faltas bizantinas
  - ☞processamento de erros: recuperação proativa
- Prevenção, supressão e previsão de faltas são importantes mas não são específicos da TI
  - ☞TI não substitui os meios de Segurança!!

Navigators FCUL

Sistemas Distribuídos

Navigators FCUL

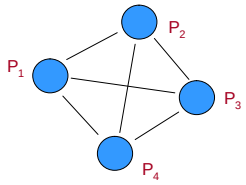
Um sistema distribuído é aquele que não o deixa trabalhar por causa da falha de um computador do qual nunca ouviu falar.

- Leslie Lamport

Navigators FCUL

Modelo topológico

- Processos + canais
  - ☞Comunicação ponto-a-ponto ou por difusão
  - ☞Processos completamente ligados



Navigators FCUL

Modelo de falhas

- Não há falhas, paragem de processos, omissões na comunicação...
- Geralmente vamos considerar:
  - ☞Processos podem falhar arbitrariamente
  - ☞Comunicação pode falhar arbitrariamente
    - – modificar, apagar, repetir mensagens
- Modelos híbridos:
  - ☞consideram alguns componentes simples seguros

Navigators FCUL

Modelos temporais

- Síncrono
  - ☞Há limites conhecidos p/ tempos de processamento
  - ☞Há limites conhecidos p/ tempos de comunicação
- Assíncrono
  - ☞Não há limites conhecidos p/ tempos de processam.
  - ☞Não há limites conhecidos p/ tempos de comunicaç.
  - ☞*Logo não deveria haver vulnerabilidades devido a premissas temporais...*
  - ☞*mas consenso impossível neste modelo... (FLP 85)*
- Modelos parcialmente síncronos e extensões

Navigators FCUL

Criptografia de Limiar

Navigators FCUL

Criptografia de Limiar

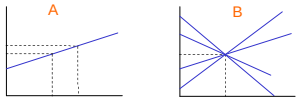
- Algoritmos tipicamente de TI que surgiram há vários âmbitos no âmbito da Segurança
  - ☞ são usados em várias das soluções que veremos
- Dados **N** processos cada um com uma *parte* secreta
- **Partilha de segredos**: permitir que **k** processos combinem as suas partes e revelem um segredo **s**; um conluio de **k-1** não pode revelar **s**
- **Partilha de funções:** permitir que **k** processos apliquem função **F** mas que **k-1** não o possam fazer
- Ex. de partilha de funções: assinatura de limiar (**k** processos conseguem fazer assinatura)

Navigators FCUL

Partilha de segredos (Shamir)

- Duas propriedades dos polinômios de grau **d**:
  - ☞A- dados quaisquer **d+1** pontos distintos da curva definida pelo polinômio, é possível determinar qualquer outro ponto do polinômio;
  - ☞B- se os índices do polinômio forem todos desconhecidos, o conhecimento de até **d** pontos não revela nenhuma informação sobre outros pontos

polinômio de grau d=1:



Navigators FCUL

Algoritmo de Shamir (cont)

- Polinômio:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_d x^d$$

- Faz-se:

$$d = k - 1$$

$$a_i \text{ aleatórios, exceto } a_0 = p(0) = s$$

- Cada processo conhece um ponto de *p(x)*
- **k** processos conseguem obter *p(0) = s*
- **k-1** processos não conseguem obter qualquer informação sobre *p(0)*
- Os cálculos são feitos *módulo* um primo grande

Navigators FCUL

Limitações do algoritmo

- um processo não pode saber se a sua parte está "boa"
  - ☞*partilha de segredos verificável*
- se uma parte estiver corrompida o segredo não é reconstruído (e isso pode nem ser detectado):
  - ☞*algoritmos robustos* (baseados em provas de conhecimento zero)
- *algoritmos proativos*

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

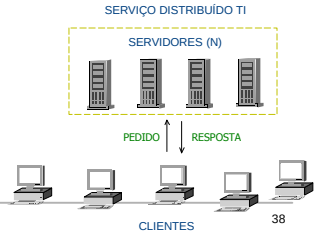
Replicação

- Idéia: código e dados replicados nos servidores
- objetivo: garantir a **disponibilidade** e a **integridade** do serviço e/ou dados (atributos)
- Duas aproximações: **replicação de máquinas de estados** e **sistemas de quoruns**

Replicação de Máquinas de Estados

Replicação de Máq.de Estados

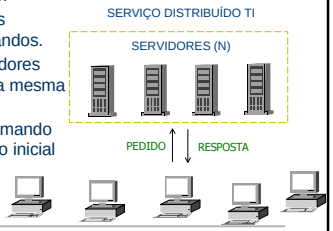
- solução genérica para a concretização de *serviços* tolerantes a faltas
- cada servidor é uma máquina de estados definida por variáveis de estado; comandos atômicos



Replicação de Máq.de Estados

- todos os servidores seguem a mesma sequência de estados sse:
- Estado inicial. Todos os servidores começam no mesmo estado.
- Acordo. Todos os servidores executam os mesmos comandos.
- Ordem total. Todos os servidores executam os comandos pela mesma ordem.
- Determinismo. O mesmo comando executado no mesmo estado inicial gera o mesmo estado final.

protocolo de difusão atômica



Resistência

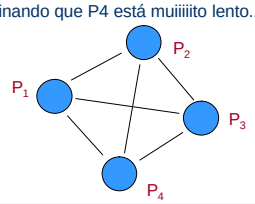
- número máximo **f** de **servidores** que podem falhar (ser corrompidos) para o serviço se manter correto
- em sistemas assíncronos TI baseados em replicação de máquinas de estado este limite é imposto pelo *protocolo de difusão atômica*:

$$f = \lfloor \frac{N-1}{3} \rfloor \qquad N \geq 3f+1$$

  - ex: N=4 servidores para tolerar f=1 corrompido; N=7 para tolerar f=2
- não há limite ao nº **clientes** que podem falhar
- cobertura das premissas – pode-se definir **f**?

Porquê 3f+1?

- Mínimo maioria de processos correctos: N = 2f+1
- N = 3f+1
  - Ex: N=4, f=1. Quantas mensagens pode esperar cada processo?
  - Imaginando que P4 está muiiiiito lento...



BFT
Byzantine Fault Tolerance

BFT

M. Castro e B. Liskov, MIT

- não é o primeiro serviço de RME TI mas é um dos mais citados trabalhos da área
- objetivos:
  - ☞sempre correto; progresso depende de premissa temporal fraca (FLP)
  - ☞bom desempenho (NFS TI com desempenho entre -24% a +2% de versões não replicadas)
- na discussão que se segue vamos considerar o caso mais apertado: $N = 3f + 1$
  - ☞ e não o caso geral...

Comunicação

- Cada par cliente-servidor partilha uma chave secreta
- Comunicação é protegida usando *message authentication codes*, MACs (ex: HMAC-MD5)
  - ☞cada mensagem leva um MAC
  - ☞o uso de MACs em vez de assinaturas baseadas em cripto de chave pública é uma das razões do bom desempenho
- Servidor-servidor é igual mas c/ 2 chaves

Cliente

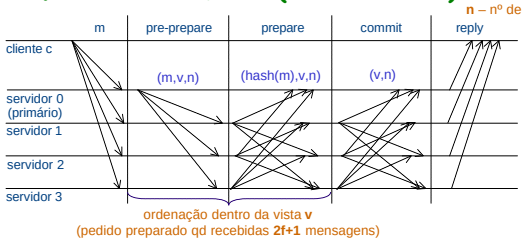
- quando pretende fazer um pedido difunde uma mensagem para todos os servidores com:
  - ☞o comando
  - ☞uma estampilha temporal (timestamp) para garantir que o comando é executado uma só vez
  - ☞um vetor de MACs, um MAC / servidor
    - – cada servidor só processa o comando se o seu MAC estiver correto)
- o cliente aceita a resposta se receber **f+1** cópias vindas de servidores diferentes

Replicação

- o algoritmo é baseado numa mistura de replicação passiva (ou primário-secundário) e **replicação ativa**
- as réplicas vão mudando de configuração (vista)
  - ☞ cada vista: um servidor primário; restantes secundários
- a ordem de execução dos pedidos é definida pelo primário
  - ☞ atribui próximo número de sequência a cada pedido recebido e reenviando-o para os secundários
- primário malicioso pode:
  - ☞ dar o mesmo número a dois pedidos
  - ☞ parar de atribuir números
  - ☞ deixar intervalos entre os números
- ⇒ os secundários verificam os números de sequência atribuídos pelo primário e marcam o tempo para ver se ele pára
- quando os secundários suspeitam que o primário falhou, mudam de vista e de primário.

Difusão atômica (acordo+ordem)

v – nº da vista
n – nº de ordem



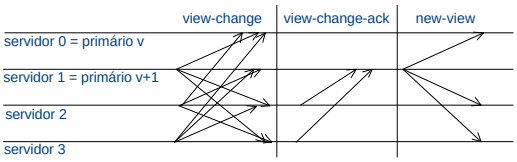
- Primário malicioso não pode criar um pedido do nada pois as réplicas corretas só processam pedidos que tenham recebido de um cliente.
- Cliente ou primário maliciosos não podem provocar a execução de dois comandos diferentes pela mesma ordem em duas réplicas corretas diferentes.

Fase commit

- as fases *pre-prepare* e *prepare* ordenam mensagens dentro de uma vista
- se o primário muda, pode ser atribuído um nº de ordem diferente à mensagem
- a fase *commit* resolve esse problema:
  - ☞quando um servidor recebe **2f+1** mensagens *commit* com o mesmo **(v,n)**, a mensagem fica confirmada
  - ☞quando não houver nenhuma mensagem não confirmada com **n´<n**, o comando é executado

Navigators FCUL

Mudança de vista



- Quando os secundários *suspeitam* primário...
- *Suspeitar ? detectar*
  - ☞ BFT pode ser atrasado forçando a mudança de vistas

Navigators FCUL

Otimização 1

- Dois tipos de comandos: leitura e escrita
  - ☞para garantir **ordem total** basta ordenar as escritas
- Otimização:
  - ☞servidores executam imediatamente os **comandos de leitura** vindos de clientes (sem as 3 fases...)
  - ☞retornam a resposta
  - ☞se o cliente recebe **f+1** respostas idênticas aceita;
  - ☞caso contrário reenvia e os servidores processam o comando da forma normal (3 fases)
  - ☞só não recebe **f+1** respostas iguais se houver concorrência com escritas (improvável)

Navigators FCUL

Otimização 2

- processamento em pacotes de pedidos *(batching)*
- o primário tenta juntar diversos pedidos antes de atribuir um número de ordem

Navigators FCUL

Outros sistemas

Navigators FCUL

Rampart

M. Reiter, ATT

- Anterior e mais genérico do que o BFT: **sistema de comunicação em grupo** (Isis, Horus, xAMP,...)
- Oferece primitivas de **comunicação** em difusão:
  - ☞ difusão confiável
  - ☞ difusão atômica
- Tem um protocolo de **filiação** que gere quem são os membros do grupo:
  - ☞ entradas, saídas, remoção de falhados (⇒ detecção de falhas)
- pior desempenho do que o BFT (usa assinaturas, etc)
- atacante pode atrasar processos corretos e causar a sua remoção (no BFT não); problemático para RME
- respostas podem ser assinadas com assinatura de limiar (mas só piora o desempenho)

Navigators FCUL

SINTRA

Cachin, Poritz, Kursawe, Shoup, IBM Zurich MAFTIA

- grupos estáticos
- difusão confiável, causal, atômica
- BFT e Rampart contornam FLP usando detectores de falhas; SINTRA usa um protocolo de acordo aleatório
- usa cripto chave pública, incluindo assinaturas de limiar
  - ☞desempenho fraco em LANs
  - ☞em WANs o tempo de processamento pode ser negligenciável face ao tempo de comunicação

Navigators FCUL

SecureRing e SecureGroup

- sistemas de comunicação em grupo
- vocacionados para LANs
- SecureRing:
  - ☞anel lógico; passagem de token (assinado)
  - ☞detecção de falhas
  - ☞melhor desempenho que Rampart
- SecureGroup:
  - ☞protocolo de difusão atômica aleatório
  - ☞a resistência é menor pois todas as mensagens de todos os processos que podem vir a entrar têm de ser ordenadas

L. Moser, M. Melliar Smith, K. Kihlstrom, UC Stª Barbara

Navigators FCUL

Worm-IT

M. Correia, N. Neves, P. Veríssimo, U. Lisboa
L. Lung, PUC-PR

- sistemas de comunicação em grupo
- *modelo de falhas híbrido*: os servidores são extendidos com um componente distribuído "simples" e seguro que faz "acordo"
  - ☞kernel seguro distribuído
- protocolo de difusão atômica é eficiente (não usa cripto de chave pública) e totalmente distribuído (não há primário)
- bom desempenho

Navigators FCUL

Otimização da resistência

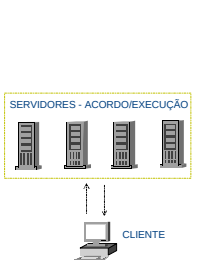
Navigators FCUL

Resistência

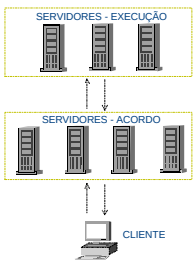
- Será que a resistência é indiferente?
- Cada réplica tem dois custos
  - ☞hardware e software
  - ☞de projeto (código feito à medida...)
- Diminuir o número de réplicas é importante!
- Mas em sistemas assíncronos já vimos que o mínimo é N = 3f+1 por causa da difusão atômica

Navigators FCUL

Separação acordo-execução



Arquitetura RME normal



2f+1 servidores "normais" que executam o serviço

3f+1 servidores "simples" que fazem acordo sobre ordenação

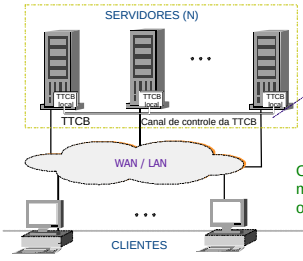
Proposta de Yin et al.

Navigators FCUL

Resistência 2f+1

M. Correia, N. Neves, P. Veríssimo, U. Lisboa

- Usando um **modelo de falhas híbrido arquitetural** consegue-se diminuir o nº de servidores de **3f+1** para **2f+1** (25% a 33%)



WORMHOLE: subsistema seguro e com funcionalidade limitada

O serviço é executado nos servidores mas chama a TTCB para ajudar a ordenar os pedidos.

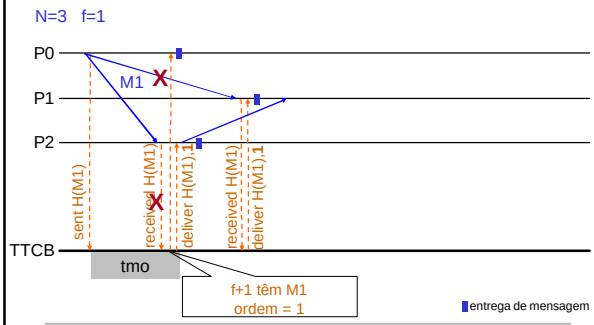
Navigators FCUL

Serviços da TTCB

- Trusted Multicast Ordering Service (TMO)
  - ☞O objetivo é suportar a execução de um protocolo de difusão atômica TI
  - ☞Não é afectado por faltas bizantinas (ataques, intrusões) pois é executado dentro da TTCB
- (existem outros mas não são necessários aqui)

Navigators FCUL

Difusão atômica com o TMO



Navigators FCUL

Serviço TMO

- Núcleo da solução
  - ☞Decide quando uma mensagem pode ser entregue
    - – se **f+1** servidores mostram que têm a mensagem, então pelo menos um correto tem
  - ☞Define uma ordem sequencial para as mensagens
  - ☞Resultados são confiáveis pois a TTCB é segura
- Concretização do serviço
  - ☞Quando há uma mensagem, TTCBs locais executam um *protocolo de acordo* para decidirem o nº de ordem
  - ☞Esse protocolo é executado num ambiente benigno, não tem de tolerar faltas bizantinas

Navigators FCUL

Envio de pedidos

- Os clientes têm relógios locais (não confiáveis) Protocolo:
  - ☞Enviar o pedido para um servidor
  - ☞Esperar por **f+1** respostas idênticas de servidores diferentes
  - ☞Se T_{resend} depois do pedido ter sido enviado não tiverem sido recebidas as respostas, reenviar para **f** servidores adicionais

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | **SISTEMAS DE QUORUNS** | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

Sistemas de quoruns

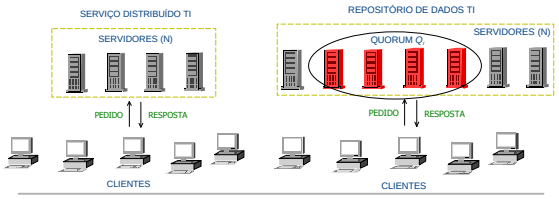
- **quorum**: um conjunto de servidores
- um **sistema de quoruns** $\mathcal{Q}$ é um conjunto de quoruns tal que:

$$\forall Q_1, Q_2 \in \mathcal{Q}, Q_1 \cap Q_2 \neq \emptyset$$

- as operações são feitas sobre um quorum, não sobre todos os servidores – escalabilidade

Quoruns vs RME

- **RME** – solução genérica para concretizar **serviços distribuídos TI**
- **Quoruns** – permitem concretizar **repositórios de dados**



RME QUORUNS

Mais características

- os sistemas de quoruns permitem *raciocinar* sobre os servidores e definir **objetos distribuídos** com diferentes semânticas:
  - ☞variáveis compartilhadas (VC) – um repositório de dados pode ser considerado um conjunto de VCs
  - ☞objetos de exclusão mútua
  - ☞objetos de consenso
- (geralmente) não precisam de fazer consenso, logo não ficam circunscritos pelo FLP – podem ser inteiramente assíncronos

Memória compartilhada vs Comunicação por mensagens

- Dois tipos de algoritmos distribuídos:
  - ☞baseados em comunicação por mensagens – *message passing*
    - – todos os que vimos e veremos...
  - ☞baseados em memória compartilhada – *shared memory*
    - – memória que pode ser lida e escrita por diversos processos
    - – mas em sistemas distribuídos a memória compartilhada tem de ser concretizada usando comunicação por mensagens...
    - – ... é o que veremos, com quoruns

Semânticas de vars compartilhadas

- Lamport, 1986
- Quantos processos escritores e leitores?
  - ☞múltiplos-escritores/múltiplos-leitores
  - ☞Lamport - apenas um-escritor/ múltiplos-leitores
- Semânticas de consistência
  - ☞a operação o_1 *acontece antes* de o_2 se termina antes de o_2 começar
  - ☞o_1 e o_2 são *concorrentes* se o_1 não acontece antes de o_2 e o_2 não acontece antes de o_1
  - ☞as definições das semânticas não consideram escritas concorrentes (seguindo Lamport)

Semânticas de consistência

- **segura**:
  - ☞ uma leitura que não seja concorrente com nenhuma escrita retorna o último valor escrito;
  - ☞ uma leitura concorrente com 1+ escritas retorna qualquer valor;
- **regular:**
  - ☞ garante a semântica segura e
  - ☞ leitura concorrente com escritas retorna um dos valores dessas escritas ou o último valor escrito antes da leitura;
- **atômica:**
  - ☞ garante a semântica regular e
  - ☞ as escritas e leituras retornam valores como se tivessem sido feitas de acordo com uma ordem definida;
  - ☞ uma variável com esta semântica também se diz **linearizável** (na realidade é uma generalização para objetos genéricos)
  - ☞ a semântica da RME é atômica para leituras e escritas
- *Há muitas outras, por exemplo:*
  Adve e Gharachorloo. Shared Memory Consistency Models: A Tutorial, 1995.

Variáveis compartilhas c/quoruns

- VCs são replicadas em todos os servidores
- cada VC **x** tem em cada servidor **u**:
  - ☞uma cópia $\mathbf{x_u}$
  - ☞uma estampilha temporal $\mathbf{t_{x,u}}$
    - – quando a VC foi escrita pela última vez
    - – as estampilhas de diferentes escritores não se intersectam
- existe um conjunto $\mathbf{W_x}$ com os identificadores dos servidores que podem escrever em **x**
  - ☞$\mathbf{W_x}$ é conhecido por todos os clientes e servidores (pode ser codificado no nome **x** ou guardado noutra VC)

VCs c/quoruns (cont)

- clientes comunicam com servidores usando uma chamada a procedimento remoto quorum **Q-RPC(m)**
  - envia **m** a um subconjunto de servidores
  - recolhe resposta de um quorum (pode exigir reenvios, etc.)
  - nem os clientes nem os servidores comunicam entre si!
- há várias classes de sistemas de quoruns; veremos duas:
  - quoruns de *disseminação-f*
  - quoruns de *mascaramento-f*

Quoruns de disseminação-f

- os valores são auto-verificáveis (são assinados pelo cliente que os escreve)
  - os servidores não podem forjar ou modificar valores
- os quoruns têm de obedecer a 2 propriedades:

$$\left.\begin{array}{l} \textit{Consistência. } \forall Q_1, Q_2 \in \mathcal{Q}, |Q_1 \cap Q_2| \geq f+1 \\ \textit{Disponibilidade. } \forall Q \in \mathcal{Q}, |Q| \leq n-f \end{array}\right\} \Rightarrow N \geq 3f+1$$

- vamos assumir o tamanho: $\forall Q \in \mathcal{Q}, |Q| = \lceil \frac{N+f+1}{2} \rceil$
- *f=1 Þ N=4 ; N=4 Þ |Q|=3*

Registo c/semântica regular

- Começamos por assumir escritores corretos
- **Escrever** o valor **v** na VC **x**:
  - fazer Q-RPC para obter o conjunto de estampilhas temporais $\{t_{x,u}\}_{u \in Q_1}$ do quorum Q_1
  - escolher uma estampilha **t** maior do que todas as obtidas e fazer Q-RPC para enviar o par $\langle v, t \rangle$ assinado para um quorum Q_2
- o que podem fazer de mal **f** servidores maliciosos?
- **Ler** o conteúdo de **x**:
  - fazer Q-RPC para obter o conjunto de pares assinados valor/estampilha $\{\langle v_u, t_u \rangle_{w_u}\}_{u \in Q_1}$ no quorum Q_1
  - retornar o valor com a maior estampilha

Registo c/semântica regular (cont)

- Escritores maliciosos
- Podem escrever valores diferentes em cópias diferentes ⇒ protocolo de *echo*
  - o escritor envia o par para os servidores e obtém ecos assinados vindos de um quorum inteiro Q_1
  - o escritor envia o par e as assinaturas para os servidores do mesmo quorum Q_1
  - servidores aceitam o valor se contiver as assinaturas de todos os servidores em Q_1
  - (como todos os quoruns se intersectam é impossível o escritor escrever em dois quoruns completos dois pares diferentes $\langle v, t \rangle$ e $\langle v', t \rangle$)

Quoruns de mascaramento-f

- dados não auto-verificáveis
  - pode não ser conveniente distribuir chaves públicas de cada escritor por todos os potenciais leitores

$$\textit{Consistência. } \forall Q_1, Q_2 \in \mathcal{Q}, |Q_1 \cap Q_2| \geq 2f+1$$
$$\Rightarrow N \geq 4f+1 \text{ e } \forall Q \in \mathcal{Q}, |Q| = \lceil \tfrac{N+2f+1}{2} \rceil$$

Registo c/semântica segura

- **Escrever** o valor **v** na VC **x**:
  - fazer Q-RPC para obter o conjunto de estampilhas $\{t_{x,u}\}_{u \in Q_1}$ do quorum Q_1
  - escolher estampilha **t** maior do que todas as recebidas
  - fazer Q-RPC para enviar $\langle v, t \rangle$ e receber ecos assinados de um quorum Q_2
  - fazer Q-RPC para enviar os ecos para Q_2, escrevendo $\langle v, t \rangle$ nesse quorum
- **Ler** o valor da VC **x**:
  - fazer Q-RPC para obter o conjunto de pares $\{\langle v_u, t_{x,u} \rangle_{w_k}\}_{u \in Q_1}$ no quorum Q_1, cada um assinado pelo servidor onde está
  - retornar o valor **v** do par $\langle v, t \rangle$ com maior **t** que apareça em pelo menos **f+1** respostas (ou $\perp$ se não houver nenhum)

semântica segura

Especificação de quoruns

- Nos algoritmos atrás os quoruns aparecem explicitamente (Q_1 , Q_2) logo podem ser quaisquer
- Forma mais simples mas menos genérica: especificar quoruns através do nº de servidores: f+1, 2f+1,...
- Alguns trabalhos também apresentam os servidores falhados de forma genérica – *fail-prone system:*
  - ☞ conjunto de subconjuntos de servidores;
  - ☞ um dos subconjuntos contém todos os servidores que podem falhar

Outros algoritmos

Comparação de semânticas e resistências:

referência	disseminação-f confirmável	mascaramento-f confirmável	disseminação-f não-confirmável	mascaramento-f não-confirmável
[Malkhi and Reiter, 1998a]	regular, 3f+1	segura, 4f+1		
[Malkhi and Reiter, 1998b]	atômica 3f+1	segura, 4f+1		
[Martin et al., 2002a]	atômica 3f+1	atômica 3f+1	regular, 2f+1	regular, 2f+1
[Martin et al., 2002b]	regular, 3f+1	segura, 4f+1	regular, 2f+1	segura, 3f+1

Algoritmo **confirmável**: é possível saber quando terminam as escritas

Outros trabalhos na área

- Outros objectos de memória compartilhada:
- **exclusão mútua**:
  - ☞ entrar seção crítica, sair seção crítica
  - ☞ algoritmo “no máximo 1” (Malkhi e Reiter)
  - ☞ algoritmo genérico (Neves, Fraga, Lung)
- **consenso**
  - ☞ algoritmo aleatório (Malkhi e Reiter)
- algoritmos para **quoruns dinâmicos** (Martin, Alvisi): Q-RPC ⇒ DQ-RPC

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | **FRAGMENTAÇÃO**

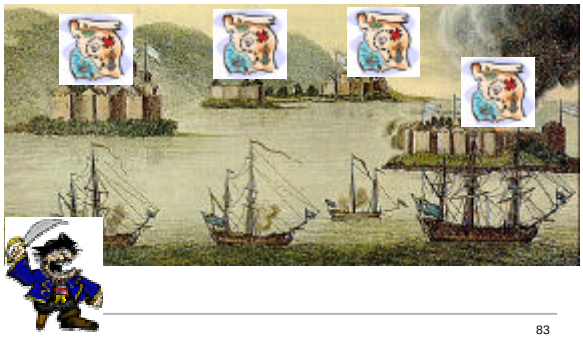
RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Replicação



Fragmentação

disponibilidade, integridade, confidencialidade
armazenamento eficiente

Navigators FCUL

Quoruns vs Fragmentação

- **Quoruns**: a força é colocada na concretização de variáveis com diversas semânticas e outros objectos – dados "pequenos"
- **Fragmentação**: a força é colocada no armazenamento de arquivos "grandes"
  - daí a importância de armazenar os dados *fragmentados* de forma eficiente

Navigators FCUL

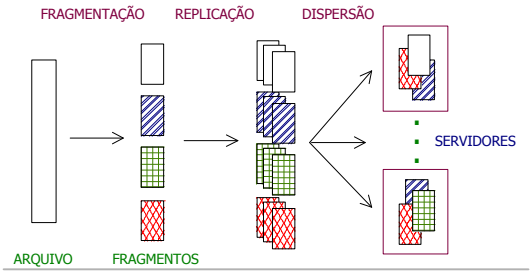
Fragmentação – FRS

- Trabalho seminal: Fraga e Powell 85 – FRS
  - *fragmentation-redundancy-scattering*
  - não fornece confidencialidade... *"reduz o significado da informação disponível a um intruso"*

Navigators FCUL

FRS

- armazenamento de um arquivo F em N servidores



Navigators FCUL

FRS (cont)

- Integridade – 2 soluções para detectar fragmentos corrompidos:
  - junta-se um MAC a cada fragmento
  - lê-se diversas cópias de cada fragmento; votação (reduz resistência)
- Servidores de segurança (TI)
  - localização dos arquivos
  - autorização de acesso aos arquivos
- Este trabalho trata de muitos dos problemas que foram depois estudados em TI

Navigators FCUL

Códigos de apagamento

Navigators FCUL

Otimização do espaço

- Como melhorar o espaço ocupado por um arquivo nos servidores?
  - **códigos de apagamento** (erasure codes)
    - usam informação redundante para permitir recuperar os dados se uma parte for *apagada*
  - **códigos de correção de erros** (error correcting codes)
    - usam informação redundante para permitir recuperar os dados se uma parte for *corrompida*
  - nesta área têm sido mais usados os códigos de apagamento

Navigators FCUL

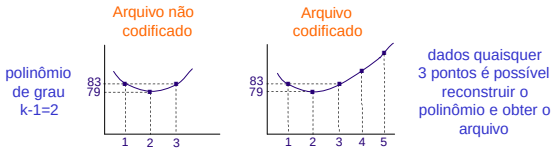
Códigos de apagamento

- M. Rabin, 1989
  - ☞armazenamento eficiente
  - ☞não aborda a distribuição nem a confidencialidade
  - ☞outros trabalhos a seguir abordaram a distribuição mas ainda não a confidencialidade
    - – Krawczyk, Alon et al., Garay et al., Goodson et al. (PASIS/OASIS)
- **código de apagamento-(k,N)**
  - ☞o arquivo é dividido em **N** fragmentos
  - ☞são necessários **k** fragmentos para o reconstruir
    - – (no máximo)

Navigators FCUL

Códigos de apagamento

- Exemplo baseado em interpolação de polinômios (Cachin e Tessaro):
  - ☞arquivo: SOS = 83-79-83
  - ☞k = 3, N=5



  - ☞para evitar usar valores muito grandes, o arquivo é dividido em pedaços mais pequenos (*striping*)

Navigators FCUL

Fragmentação c/confidencialidade

- Soluções para confidencialidade são recentes:
- Cachin e Tessaro 2004 (aliás, SRDS'05)
  - ☞não lida com acessos concorrentes
  - ☞cada arquivo só pode ser escrito uma vez (não tem versões)
- Cachin e Tessaro 2005
  - ☞retira essas limitações
  - ☞variável compartilhada com semântica atômica

Navigators FCUL

AVID

- *Asynchronous Verifiable Information Dispersal*
  - ☞não fornece confidencialidade, apenas integridade e disponibilidade
- **Arquivo F**
  - ☞codificado como um vetor $[F_1,...,F_N]$ usando um *código de apagamento-(k,N)*
  - ☞obtidas *impressões digitais*: vetor com sínteses criptográficas de cada F_i: $D = [D_1,...,D_N]$
  - ☞cada par (F_i,D) é enviado para um servidor usando o protocolo de dispersão

Navigators FCUL

AVID – protocolo de dispersão

- garante que todos os servidores armazenam um fragmento se o cliente for correto
  - ☞ou nada se for malicioso
- protocolo
  - ☞o cliente envia para cada servidor s_i o par (F_i,D)
  - ☞todos os servidores enviam para todos o seu par
  - ☞se o cliente for malicioso e faltarem alguns fragmentos, os servidores tentam reconstruí-los a partir dos que foram enviados
    - – se não for possível descartam tudo
  - ☞os servidores apagam todos os fragmentos que não lhe pertecem

Navigators FCUL

AVID – leitura do arquivo

- O cliente pede fragmentos aos servidores até obter um total de **k**
  - ☞as impressões digitais são usadas para detectar fragmentos corrompidos
  - ☞(o vetor **D** correto é o devolvido pela maioria dos **k** servidores)
- Valores de **k**: $f+1 = k = N-2f$
- Resistência máxima: $N=3f+1 \Rightarrow k=f+1$

cAVID – confidencialidade

- controle de acesso:
  - ☞ para cada arquivo é guardada uma lista **L** dos clientes que o podem ler
- usa criptografia de limiar:
  - ☞ cada servidor tem uma chave privada $\mathbf{SK}_i$
  - ☞ todos os clientes têm a chave pública **PK**
- **escrita**:
  - ☞ o cliente gera uma chave secreta **K** e cifra o arquivo usando **K**
  - ☞ o cliente cifra **K** usando **PK**
  - ☞ o cliente usa o AVID para armazenar: o arquivo, **K** cifrada e **L**
- **leitura**:
  - ☞ é necessária a colaboração de **k** servidores para recuperar a chave **K** e decifrar o arquivo

cAVID – dicussão

- O esquema usado para obter confidencialidade é (quase) ortogonal à fragmentação
  - ☞ podia ser usado com RME ou quoruns desde que se refrescasse a chave secreta **K** (como?)
  - ☞ usando fragmentação o refrescamento é desnecessário (assume-se que $k>f$ e que não falham mais de f servidores)

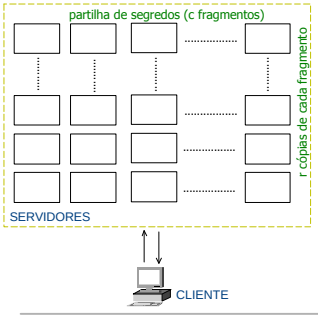
Partilha de segredos

Partilha de segredos

- é uma solução evidente para obter confidencialidade
- mas quase não é usada
  - ☞ eficiente para armazenar arquivos “grandes”?
- excepção: **secure store**
  - ☞ combina partilha de segredos com replicação

Secure store

OASIS



- N = rc
- **escrever**:
  - ☞ usar *partilha de segredos (k,c)* para obter **c** fragmentos
  - ☞ copiar cada fragmento numa coluna completa
- **ler**:
  - ☞ obter **k** fragmentos de servidores em colunas diferentes
- a resistência é baixa...
- faz refrescamento de fragmentos
- não tolera clientes maliciosos

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

Fragmentação

se tiver muito tempo o pirata pode conseguir vários fragmentos...

Navigators FCUL

Recuperação proativa

periodicamente faz-se uma *renovação*

Navigators FCUL

Processamento de erros e TI

- processamento de erros faz parte da *tolerância a faltas*
- técnica mais intuitiva em TI:
  - detectar intrusões + renovar servidor
  - mas IDSs geram muitos falsos positivos e falsos negativos...
- **recuperação proativa**:
  - periodicamente renova-se cada servidor
  - a premissa habitual passa a ser *não podem falhar mais do que **f** servidores em cada período* – **janela de vulnerabilidade**

Navigators FCUL

Premissas

- Recuperação proativa parte uma diversas premissas nem sempre deixadas claras – modelo de adversário móvel
  - cada réplica leva tempo a corromper
  - o atacante ataca um servidor de cada vez, sucessivamente
- Realista? O que demora num ataque?
  - o ataque propriamente dito?
  - descobrir a vulnerabilidade e preparar o *exploit?*

Navigators FCUL

BFT-PR

Navigators FCUL

BFT-PR

- evolução do BFT (replic. máq. de estados)
- recuperação envolve 3 operações:
  1. renovar as chaves secretas usadas para proteger a comunicação (p/obter os MACs)
  2. repor o código do sistema caso tenha sido corrompido
  3. repor o estado do sistema caso tenha sido corrompido

Navigators FCUL

BFT-PR - premissas

- Em cada réplica é necessário:
  - ☞coprocessador criptográfico
    - armazena a chave privada da réplica e assina e decifra mensagens sem expor essa chave
  - ☞memória não volátil só de leitura
    - guarda as chaves públicas de cada uma das outras réplicas e o monitor de recuperação (p.ex. a BIOS)
  - ☞temporizador seguro
    - para disparar a recuperação (existem em hw)
- Premissa: o atacante não tem acesso físico ao servidor

Navigators FCUL

BFT-PR - operações

- Renovação de chaves secretas
  - ☞periodicamente o servidor s_i envia ao servidor s_j:
    $\langle \text{NEW-KEY}, i, \ldots, \{k_{j,i}\}_{\varepsilon_j}, \ldots, t \rangle_{\sigma_i}$ (contador (evita replays))
  - ☞k servirá para proteger mensagens de s_j para s_i
  - ☞igual para cliente-servidor mas 1 chave / 2 direcções
- Recuperação do código do sistema
  - ☞periodicamente é criada imagem em disco e *reboot*
  - ☞detecta-se corrupção usando sínteses criptográficas
  - ☞se corrompido, obtém-se cópia de outro servidor
- Recuperação do estado
  - ☞protocolo complicado baseado em *checkpointing*

Navigators FCUL

COCA

Navigators FCUL

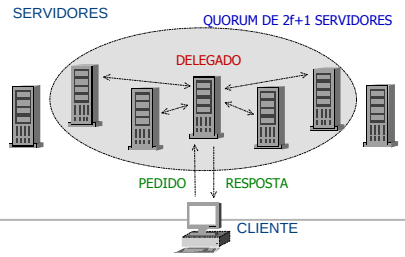
COCA

- *Cornell On-line Certification Authority* (OASIS)
- autoridade de certificação *on-line*
  - ☞fornece certificados com associações {nome, chave pública}
  - ☞update: criar, actualizar ou invalidar certificados
  - ☞query: obter o certificado correspondente a um nome
- baseado em **quorums de disseminação**
  - ☞***N = 3f+1*** ***|Q| = 2f+1***
- usa esquema de *assinatura de limiar-(k,N)*
  - ☞todos os clientes e servidores têm a chave pública
  - ☞a chave privada está dividida pelos servidores e são necessários ***k=f+1*** para assinarem um certificado

Navigators FCUL

COCA - funcionamento

- pedido enviado ao delegado que reenvia para um quorum (bastam f+1 para assinar certificado)
- delegado malicioso pode exigir retransmissão (p/ f+1)



Navigators FCUL

COCA - recuperação proativa

- três operações:
  - ☞renovar partes da chave privada de cada servidor
  - ☞repor código do servidor se corrompido
  - ☞repor estado do servidor se corrompido
  - } ~ BFT-PR
- renovar partes da chave privada
  - ☞se o atacante apanhasse ***f+1*** poderia personificar o serviço...
  - ☞*protocolo proativo de partilha de segredos APSS*
    - periodicamente gera novas partes
    - a chave privada mantém-se a mesma
    - a chave privada nunca é materializada num servidor
- sistema CODEX: versão do COCA para armazenamento de dados

Navigators FCUL

Recuperação proativa assíncrona?

Navigators FCUL

Recuperação proativa em sistemas assíncronos?

- modelo assíncrono
  - ☞não estabelece hipóteses temporais
  - ☞logo não cria vulnerabilidades desse tipo
- *não se pode fazer recuperação proativa de forma segura (safe) em sistemas assíncronos*
  - ☞resultado recente
  - ☞Sousa, Neves, Veríssimo 2005

Navigators FCUL

Demonstração (1)

- um protocolo ou algoritmo depende de um modelo – conjunto de premissas, p.ex.:
  - ☞tipo e quantidade de faltas que podem ocorrer (p.ex. bizantinas e não mais do que **f**)
  - ☞sincronia da execução
- estas hipóteses são uma abstração dos recursos de que o protocolo necessita para a sua execução
  - ☞p.ex. assumir que não mais do que **f** servidores podem falhar, significa que o protocolo precisa de **N-f** servidores corretos para funcionar corretamente

Navigators FCUL

Demonstração (2)

- interessam hipóteses quantitativas de segurança (safety) sobre recursos
  - ☞não as de progresso (liveness)
- dado um recurso qualquer **r**
- **r** diz-se **exausto** se tiver sido violada uma hipótese quantitativa de segurança sobre **r**
- um sistema diz-se **seguro-de-exaustão-r** se for capaz de garantir que o recurso **r** não será exausto
  - ☞ex: **r** = nº máx. servidores que podem falhar (**f**)

Navigators FCUL

Demonstração (3)

- **A** é um sistema
- $\mathbf{At_{start}}$ – instante de início da execução de **A**
- $\mathbf{At_{end}}$ – instante de terminação
- $\mathbf{At_{exaust}}$ – instante no qual um recurso **r** é exausto
  - ☞todos estes instantes são instantes de tempo real
- O sistema **A** é seguro-de-exaustão-r sse $\mathbf{At_{end}} < \mathbf{At_{exaust}}$

Navigators FCUL

Demonstração (4)

- Sistema **A** = serviço replicado
- **A** é seguro-de-exaustão-servidor se o *nº máx. de servidores que podem falhar (**f**)* não for exausto
- Existe um $\mathbf{At_{exaust}}$
  - ☞desconhecido e dependente do poder do atacante
- Num sistema assíncrono não há limites para tempos de processamento e comunicação...
- logo não é possível garantir que $\mathbf{At_{end}} < \mathbf{At_{exaust}}$

Demonstração – recuper.proativa

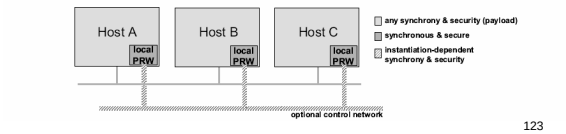
- objetivo da recuperação proativa: garantir que não se atinge $\mathbf{At_{exaust}}$ através da remoção periódica de intrusões
- logo o período de recuperação $\mathbf{T_r < At_{exaust}}$
- mas num sistema assíncrono não é possível garantir isso!

Recuperação proativa assíncrona?

- os sistemas de recuperação proativa que se dizem assíncronos escondem hipóteses temporais; ex:
  - ☞ *"na prática, esperamos que possam ser feitas hipóteses temporais sobre partes do sistema que não tenham sido comprometidas"* Zhou et al. 2002
  - ☞ há quem diga que em segurança todas as premissas são vulnerabilidades...
  - ☞ mas as premissas escondidas são vulnerabilidades ainda mais graves!

Solução

- sistema assíncrono + componente distribuída segura que encapsula a sincronia necessária
- ex: wormhole síncrono
  - ☞ a premissa é clara: o wormhole é seguro e tem uma certa sincr. ⇒ desafio é forçar essa premissa
  - ☞ como o wormhole é um componente simples e com interface bem definida...



Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

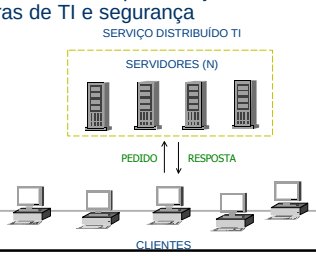
ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

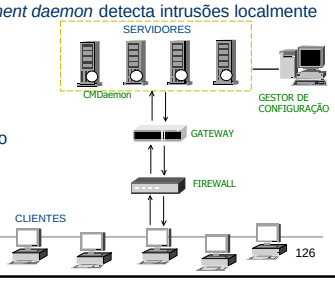
Arquiteturas

- vimos quatro tipos se soluções:
  - ☞ RME, quoruns, fragmentação, recuperação proativa
- todas baseadas numa arquitetura simples
- arquiteturas mais complicadas juntam essas soluções com outras de TI e segurança



Encaminhamento centralizado / gestão centralizada

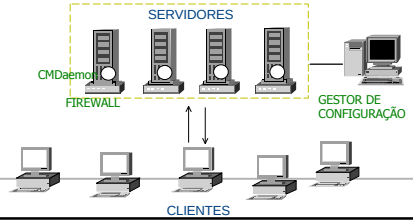
- objectivo: proteger um conjunto de servidores usando um pequeno número de componentes de confiança
- *gateway* encaminha aleatóriamente os pedidos
- *configuration management daemon* detecta intrusões localmente
- *gestor de configuração* recupera os servidores com intrusões
- não é *safe*!
- bastam **f+1** servidores
- processamento paralelo



Navigators FCUL

Encaminhamento por difusão / gestão centralizada

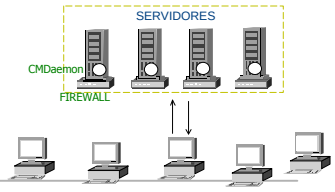
- semelhante à anterior
- não existe *gateway* e usa *firewalls pessoais* em cada servidor
- cada servidor decide quais os pedidos que processa de acordo com uma política (p.ex. atendendo à carga)
- não é *safe*! bastam **f+1** servidores; processamento paralelo



Navigators FCUL

Encaminhamento centralizado / gestão descentralizada

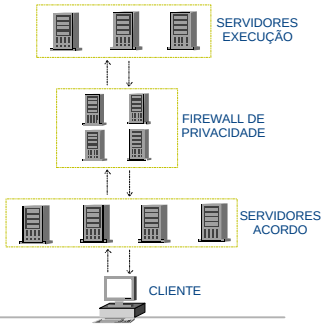
- procura combinar o bom desempenho das duas anteriores com redundância de todos os componentes
- semelhante às anteriores mas *não* tem *gestor de configuração*
- cada pedido é processado por apenas um servidor
- servidores fazem acordo sobre reconfigurações (remoção de servidores com intrusões)
- precisa de suporte de um sistema de comunicação em grupo (p.ex. Rampart)
- **3f+1**



Navigators FCUL

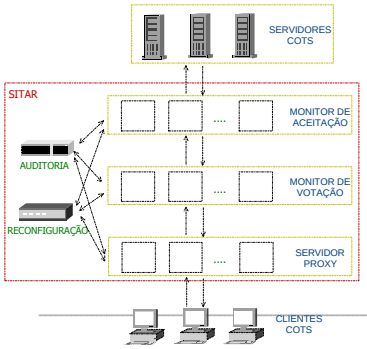
Arq. c/firewall de privacidade

- variante da arquitetura que separa execução de acordo
- *replicação de máquinas de estados:*
- uma intrusão permite quebrar fuga de informação (violar confidencialid. / privacidade)
- *esta arquitectura:*
- **firewall de privacidade** filtra info. enviada pelos servs.
- h+1 níveis cada com h+1 réplicas permite tolerar h intrusões



Navigators FCUL

Arquitetura SITAR



- *Scalable Intrusion.-Tolerant Architecture for Distributed Systems* (OASIS)
- integração com COTS
- *serv. proxy:* representa o serviço perante os clientes
- *monitor aceitação:* validam respostas dos servidores (detecção de intrusões)
- *monitor de votação:* mascaramento de intrusões

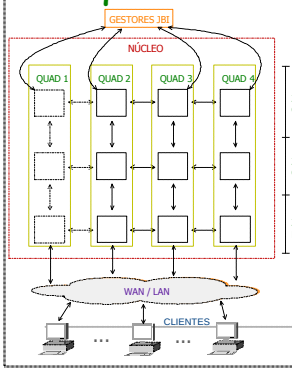
Navigators FCUL

DPASA

- *Designing Protection and Adaptation into a Survivability Architecture* (OASIS Dem/Val)
- usada em aplicação da força aérea americana: *Joint Battlespace Infosphere (JBI)*
  - ☞fornece serviços de publicação-subscrição-interrogação (PSQ) de forma a que os clientes possam trocar informação sob a forma de objetos de informação (OIs).
- o IT-JBI consiste num núcleo central que fornece serviços de comunicação a um conjunto de clientes

Navigators FCUL

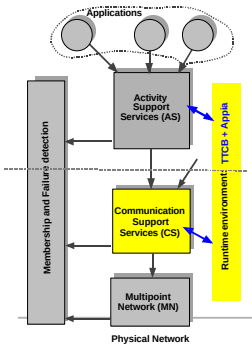
Arquitetura DPASA (simplificada)



- SOs diferentes nos *quads:*
  - ☞SELinux, Solaris, Windows XP, Windows 2000
- adaptadores de rede 3COM
  - ☞firewall embutida e VPN
- zona embate
  - ☞proxy de acesso c/interface protocolos e middleware dos clientes (em VPN)
- zona operações
  - ☞processamento, detecção OIs corrompidos
- zona de gestão
  - ☞correlaciona e filtra alarmes
- resistiu a ataques de 2 red teams (conhecimento total) excepto alguns ataques negação serviço

Navigators FCUL

Arq.Middleware MAFTIA



- Activity services
  - Transaction Service Univ.Newcastle upon Tyne(UK)
- Communication services
  - Partially synchronous protocols Univ. Lisboa (P)
  - Asynchronous protocols: IBM, Zurich (CH)
- Runtime support
  - TTCB and Appia Univ. Lisboa (P)

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

Serviços distribuídos TI

- do que vimos conclui-se que a TI já atingiu alguma maturidade
- já é possível construir *sistemas distribuídos TI* que ofereçam diversas propriedades e graus de segurança
- há ainda um conjunto de problemas não triviais com os quais o arquiteto de sistemas tem de se defrontar

Navigators FCUL

Independência de faltas e diversidade

Navigators FCUL

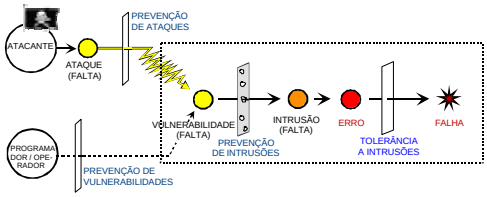
Independência de faltas

- todas as soluções de replicação e fragmentação que vimos partem da mesma hipótese:
  - *é significativamente mais difícil penetrar em* m *servidores do que num só*
  - i.e., independência de faltas ou não há modos de falha comum

Navigators FCUL

Independência de que faltas?

- ataques, vulnerabilidades, intrusões?



Navigators FCUL

Vulnerabilidades?

- são problemas desconhecidos de um sistema computacional
  - como vamos garantir que N servidores não sofrem do mesmo problema se nem sabemos do que é que sofre cada um deles?
- solução intuitiva: **diversidade**
  - os servidores devem ter sistemas operacionais diferentes, concretizações diferentes do código do sistema, etc.
  - *debate sobre a monocultura Windows na Internet*

Navigators FCUL

Diversidade

- diversos níveis (v. Obelheiro et al. SBSeg'05)
  - diversidade de software da aplicação
  - diversidade de software de suporte à execução
    - sistema operacional, middleware, bases de dados, ...
  - diversidade de hardware
  - diversidade de administradores dos servidores
  - diversidade de localização
- programação com n-versões
  - cada versão custa 0.7 a 0.85 do custo normal

Navigators FCUL

Problemas abertos

- todos estes níveis de diversidade *mais* "programação segura" podem levar a níveis de confiança elevados
  - mas qual é a independência de faltas que essa combinação realmente dá?
  - como garantir essa independência com determinado nível de confiança?
  - criação automática de diversidade?

Navigators FCUL

Recuperação proativa

- premissa: atacar um servidor toma um certo tempo
  - verdade? o que toma tempo é atacar ou descobrir como atacar?
  - se um atacante ao fim de alguns dias descobrir como atacar **f+1** servidores e o ataque demorar alguns milisegundos, a recuperação proativa é inútil
- solução:
  - em cada recuperação modificar o servidor de tal forma que o atacante tenha que recomeçar o processo de compreender como atacá-lo
  - como conseguir essa diversidade num mesmo servidor após cada recuperação?

Navigators FCUL

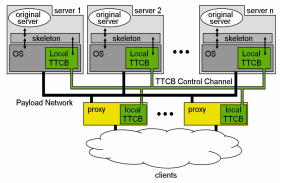
Determinismo

Navigators FCUL

Determinismo

- exigência da replicação de máquinas de estados:
  - o mesmo comando executado no mesmo estado inicial tem de gerar o mesmo estado final independentemente do servidor onde seja executado
  - mais simples nos quoruns e fragmentação (escrita / leitura de variáveis)

Protótipo Distract



- cabeçalho HTTP tem 3 campos problemáticos:
  - estampilha temporal (data/hora em que a página foi retornada)
  - *Etag:* identificação unívoca da resposta ao pedido
  - identificador do servidor
- servidores iguais em máquinas iguais; HTTP + HTML; não se usou HTTP-S, PHP, ASP.....

BASE

- *BASE - BFT with Abstract Specification Encapsulation*
  - permite o uso de réplicas diferentes ou que não sejam deterministas
- especificação abstrata do serviço
  - estado abstrato + operações que manipulam esse estado
- wrapper de conformidade
  - esconde cada réplica atrás de uma interface conforme com a especificação abstrata do serviço
- função de abstração que traduz o estado de uma concretização para o estado abstrato e vice-versa
  - caso seja necessário fazer transferência de estado entre réplicas

Problemas abertos

- Como é que se lida com:
  - dados com estrutura complexa?
  - dados comprimidos?
  - dados cifrados?
    - em sistemas seguros muitas vezes é preciso cifrar a comunicação...

Detecção e remoção de intrusões

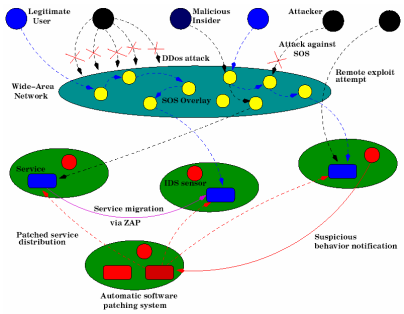
Detecção e tolerância

- detecção de intrusões:
  - seria interessante para substituir a recuperação proativa (detecção e reacção)
  - permitiria reconfigurar o sistema para tolerar essas faltas

Arquitetura SABER

Keromytis et al.

- IDS detecta scan ? filtrar tráfego dessa origem
- IDS detecta intrusão em servidor ? patching
- IDS detecta intrusão ? migração do serviço



Navigators FCUL

Problemas abertos

- falsos positivos e falsos negativos, logo resposta automática é inviável
- se reduzirmos as intrusões a detectar às que podem ocorrer num determinado serviço TI, não será possível fazer essa detecção com precisão suficiente para fazer resposta automática?

Navigators FCUL

Avaliação

Navigators FCUL

Avaliação

- **Sistemas distribuídos**:
  - número (ou complexidade) de ciclos comunicação
  - número (ou complexidade) de mensagens trocadas
  - *em geral essas métricas permitem avaliar algoritmos sem ambiguidade*
- **Confiança no funcionamento**:
  - diversos métodos para avaliar a tolerância a faltas
  - ex: métodos estocásticos
    - probabilidade de um sistema falhar dada a probabilidade de ocorrerem faltas em certos componentes e dessas faltas se propagarem

Navigators FCUL

Avaliação em TI

- trabalhos recentes têm feito avaliação usando métodos estocásticos e redes de Petri
  - assumem comportamentos probabilísticos dos ataques e intrusões; realista?
- métricas simples que permitam avaliar esses sistemas como é feito com os algoritmos distribuídos?

Navigators FCUL

Negação de serviço e privacidade

Navigators FCUL

Negação de serviço

- ataques DoS mais graves quanto maior for o processamento feito por cada pedido
  - serviços TI: cada pedido é processado por diversos servidores, operações criptográficas...
- arquitetura SABER: filtragem agressiva de tráfego na rede overlay
- serviços TI pretendem garantir a disponibilidade; ataques DoS são ataques contra a disponibilidade!
- não seria possível aproveitar as próprias características dos serviços distribuídos TI para combater esses ataques?
- qual a arquitetura adequada?

Navigators FCUL

Privacidade

- não tem sido abordada na área exceto Yin et al.
- privacidade perante administrador malicioso?

Navigators FCUL

Organização do minicurso

CONCEITOS BÁSICOS DE TI

REPLICAÇÃO DE MÁQ. DE ESTADOS | SISTEMAS DE QUORUNS | FRAGMENTAÇÃO

RECUPERAÇÃO PROATIVA

ARQUITETURAS E SISTEMAS

PROBLEMAS ABERTOS

CONCLUSÃO

Navigators FCUL

A TI permite melhorar a segurança?

- sim...
- uma arquitetura TI completa inclui numerosos componentes e mecanismos que vêm da *segurança* clássica, não são nenhuma novidade da TI (p.ex., IT-JBI)
- a possibilidade de multiplicar por N (ou ***f+1*** ou ...) a dificuldade de atacar e controlar um serviço traz claramente mais segurança

Navigators FCUL

A TI está pronta para ser usada?

- sim
- existem lacunas que podem ser pesquisadas
- a TI pode ser tornada muito mais prática de usar
- mas as soluções que vimos são reais e utilizáveis já hoje

Navigators FCUL

Em que tipo de aplicações será realmente usada?

Navigators FCUL

Tolerância a faltas

- soluções complexas e caras são usadas em aplicações que justificam o seu custo
  - ☞aviação, indústria espacial, centrais nucleares...
- soluções mais simples estão disponíveis em todo lugar:
  - ☞cabos de telecomunicações replicados
  - ☞discos rígidos (RAID)
  - ☞detecção de erros na memória dos PCs comuns

Tolerância a intrusões

- podemos esperar algo de semelhante...
- soluções complexas e caras para aplicações críticas
  - aplicações financeiras (rede SWIFT)
  - aplicações militares (IT-JBI)
  - há alguns anos a chave de raiz do sistema SET da MasterCard/VISA se encontrava distribuída por diversas empresas diferentes usando criptografia de limiar
- soluções simples poderão ser usadas para tornar mais seguros componentes/sistemas de menor custo
  - ex: soluções de armazenamento de dados em rede (NAS, SAN) TI comerciais

Perguntas?

- Página pessoal
  http://www.di.fc.ul.pt/~mpc
- Grupo Navigators:
  http://www.navigators.di.fc.ul.pt/
- Email:
  mpc@di.fc.ul.pt