SDFS: SIMPLE DISTRIBUTED FILE SYSTEM

Antônio Augusto Medeiros Fröhlich*
Thadeu Botteri Corso

Universidade Federal de Santa Catarina
Centro Tecnológico
Pós Graduação em Ciências da Computação
88.049.970 - Florianópolis - SC - Brasil
Tel.:(0482)32-9738 Fax:(0482)32-9770
E-mail: {guto/thadeu}@inf.ufsc.br

Resumo

Este artigo apresenta o SDFS (SIMPLE DISTRIBUTED FILE SYSTEM), um sistema de arquivos distribuído, portável, com alto grau de paralelismo e desenhado para ser flexível no que diz respeito ao ambiente sobre o qual seus componentes são distribuídos, possibilitando sua execução em multicomputadores ou em redes de computadores. O projeto foi desenvolvido no Curso de Pós-Graduação em Ciências da Computação da Universidade Federal de Santa Catarina (CPGCC/UFSC), onde se encontra em processo evolutivo, e deverá integrar um projeto coletivo das universidades federais de Santa Catarina (UFSC), Rio Grande do Sul (UFRGS) e Santa Maria (UFSM), que visa desenvolver um multicomputador e um ambiente para programação paralela sobre ele.

Palavras chave: sistemas distribuídos, sistemas de arquivos, sistemas de arquivos distribuídos e servidores de arquivos.

Abstract

This paper presents the SDFS (SIMPLE DISTRIBUTED FILE SYSTEM), a portable, highly parallel distributed file system, designed to be very flexible about the level of distribution of its components, allowing it to run on multicomputers as well on computers networks. The project was developed at the Computer Science Department of the Federal University of Santa Catarina (CPGCC/UFSC), where it's in an evoluting process, and will take part in a larger project, gathering the federal universities of Santa Catarina (UFSC), Rio Grande do Sul (UFRGS) and Santa Maria (UFSM), to develop a multicomputer and a parallel programming environment for it.

Keywords: distributed systems, file systems, distributed file systems and file servers.

1 Introdução

Os últimos anos têm registrado o surgimento de uma série de idéias inovadoras na área de sistemas operacionais, muitas delas motivadas pela contínua evolução do *hardware*, outras resultando do amadurecimento natural de alguns conceitos. Termos como comunicação, distribuição, paralelismo e heterogeneidade têm sido intensamente explorados, entretanto, não raras vezes esses conceitos aparecem restringidos pela tecnologia atual.

O SDFS (SIMPLE DISTRIBUTED FILE SYSTEM) é uma proposta inovadora de sistema de arquivos, livre de vários preconceitos e baseada numa visão ainda futurista de um ambiente computacional com número de unidades processadoras equiparável ao número de tarefas e, onde as entidades comunicantes dispõem de canais de comunicação hoje encontrados apenas em barramentos internos de computadores. A ânsia pela comunicação efetiva de dados, imagens e voz, resultará muito em breve em canais de comunicação de alguns Gbytes/s. Isso, aliado à rápida redução do custo das unidades processadoras, permitirá a construção de sistemas computacionais mais eficientes, onde problemas complexos poderão ser tratados na sua essência, simplificando muitas das estruturas formadoras de um sistema operacional.

Este artigo descreve a implementação de uma versão preliminar da proposta de um sistema de arquivos distribuído contida em [FRO 93], analisando principalmente os aspectos inovadores, como a divisão do sistema de arquivos em um conjunto de servidores, um mecanismo original de localização de servidores, a transparência de localidade alcançada pela extensão do mecanismo de *link* e outros. A funcionalidade do sistema de arquivos está inteiramente baseada no UNIX [BAC 87], tendo sua descrição sido excluída deste texto.

2 Metas do Trabalho

A fase de definição das metas do projeto, pela complexidade inerente de um sistema de arquivos distribuído, se estendeu por longo tempo. Algumas das características do SDFS foram definidas a priori, outras apenas durante o desenvolvimento da primeira versão. Entretanto houve uma linha de pensamento que esteve presente durante todas as etapas: fazer um sistema de arquivos distribuído que fosse realmente simples, apesar da complexidade dos temas envolvidos.

As principais metas definidas para o projeto são:

- Portabilidade: O SDFS não é um projeto isolado, devendo ser incorporado por um projeto de âmbito maior, além de ser usado como material de apoio ao ensino de sistemas distribuídos. Para permitir o porte para outras plataformas, definiu-se que o sistema deveria ser escrito em C ANSI (na verdade foi utilizado um compilador C++ com capacidade de gerar código C ANSI). Ainda no tocante a portabilidade, definiu-se que os serviços do SDFS deveriam ser compatíveis com a norma 1003.1 (POSIX.1) do INSTITUTE OF ELETRICAL AND ELECTRONICS ENGINEER [IEE 88], de forma a possibilitar a integração desse sistema de arquivos com sistemas operacionais do tipo UNIX.

- Paralelismo: Um aspecto muito explorado na definiçao do SDFS foi o aproveitamento do paralelismo natural de um sistema de arquivos, tanto no sentido de paralelisar a execução de um serviço como no sentido de executar múltiplos serviços concomitantemente. O primeiro esforço para a paralelisação do SDFS foi definir a sua implemetação não como um processo, mas como uma coleção de processos seqüenciais comunicantes [HOA 85].

- Distribuição dos servidores: Nos primórdios do desenvolvimento do projeto não se tinha uma plataforma definida, as possibilidades envolviam equipamentos tão diferentes como PCs e um multicomputador sob desenvolvimento no CPGCC/UFSC [COR 93]. Isso influenciou o projeto no sentido de que não se tinha ao certo o nível de distribuição do sistema, que poderia variar de zero, no caso do sistema estar executando inteiramente em uma única máquina, até o caso do multicomputador, onde cada servidor poderia executar em um processador próprio. Definiu-se então que o sistema deveria ser extremamente flexível no que diz respeito a distribuição de seus

servidores, de tal forma que uma alteração no nível de distribuição envolvesse apenas uma reconfiguração do sistema, dispensando recompilações e religações.

- Execução em redes de computadores: A INTERNET, através do protocolo TCP/IP, foi escolhida para caracterizar a implementação do SDFS por ser a rede mais abrangente e mais estável dos dias atuais. Assim, os aspectos do sistema relativos a nomenclatura de *hosts*, identificação de *hosts*, roteamento de mensagens e outros, são diretamente herdados da INTERNET [POS 87]. Isso possibilita que o SDFS tenha seu escopo de utilização equiparado ao da própria rede INTERNET.

- Heterogeneidade de equipamentos: Decidiu-se que o sistema não deveria levar em conta qualquer aspecto de *hardware*, possibilitando, assim, que o SDFS seja executado em ambientes heterogêneos. Toda a funcionalidade necessária a execução do sistema é fornecida por um núcleo que gerencia os processos e implementa um mecanismo de comunicação entre eles. Esse núcleo encontra-se em desenvolvimento no CPGCC/UFSC, por hora sua funcionalidade foi simulada sobre o sistema operacional SUNOS 4.1.3 da SUN MICROSYSTEMS.

- Distribuição da árvore de diretórios: Uma vez que se possibilitou a distribuição dos servidores do sistema de arquivo, decidiu-se também permitir a distribuição da árvore de diretórios, de forma que um cliente possa acessar um arquivo remoto anexado à árvore de diretórios local de forma transparente.

- Idempotência: Uma questão de extrema relevância para sistemas distribuídos sobre redes de computadores é a tolerância às falhas da rede, assunto que não está no escopo deste projeto. Entretanto, para dar alguma robustez ao sistema, decidiu-se que o sistema de arquivos seria idempotente. Desta forma, os servidores não guardam informações sobre o estado dos clientes e cada mensagem de solicitação de serviço carrega todas as informações relativas ao serviço, o que, além de facilitar a definição de um mecanismo que permita a replicação de servidores, simplifica o processo de reativação dos servidores após uma falha.

- Autenticação de mensagens: O mecanismo de autenticação de mensagens de solicitação de serviços adotado pelo SDFS consiste em gerar-se uma chave, levando em conta o identificador do arquivo em questão, uma informação temporal e o identificador do usuário que está solicitando o serviço. Sempre que um cliente solicita um serviço ao sistema de arquivos, ele fornece o identificador do arquivo e a chave, que validará ou não a solicitação.

3 Modelo Arquitetural

Durante muito tempo os sistemas de arquivos foram trechos obscuros do código de sistemas operacionais monolíticos. A evolução natural da construção de sistemas operacionais levou primeiro aos sistemas operacionais modulares, mais claros e organizados, e culminou no modelo cliente-servidor, onde os sistemas operacionais são compostos por processos que se comunicam exclusivamente através de mensagens. Esse foi o modelo adotado para a implementação do SDFS.

Na maior parte dos sistemas operacionais, o sistema de arquivos é implementado por um único servidor. No caso do SDFS, levou-se em conta a complexidade de um sistema de arquivos e, com vistas a aproveitar o seu paralelismo implícito, decidiu-se subdividí-lo em três servidores:

- Um servidor encarregado de traduzir nomes de arquivos em identificadores internos, implementando as estruturas de diretórios. Esse servidor é também responsável pela transparência de localidade dos arquivos, i. e., é ele quem conhece a localização dos arquivos e dos servidores associados.

- Um servidor responsável pela gerência de memória secundária, implementando o conceito de arquivo e estendendo o mesmo aos dispositivos de entrada e saída.

- Um servidor para armazenar, temporariamente, dados de memória secundária, visando otimizar o acesso a esses dados.

Além destes três servidores, deverão tomar parte na implementação do sistema de arquivos servidores específicos para cada tipo de dispositivo suportado. Esses servidores estão fora do escopo deste trabalho, mas foram

desenvolvidos ou simulados para permitir a implementação do sistema. Uma visão parcial de um sistema computacional rodando SDFS pode ser vista na figura 1.

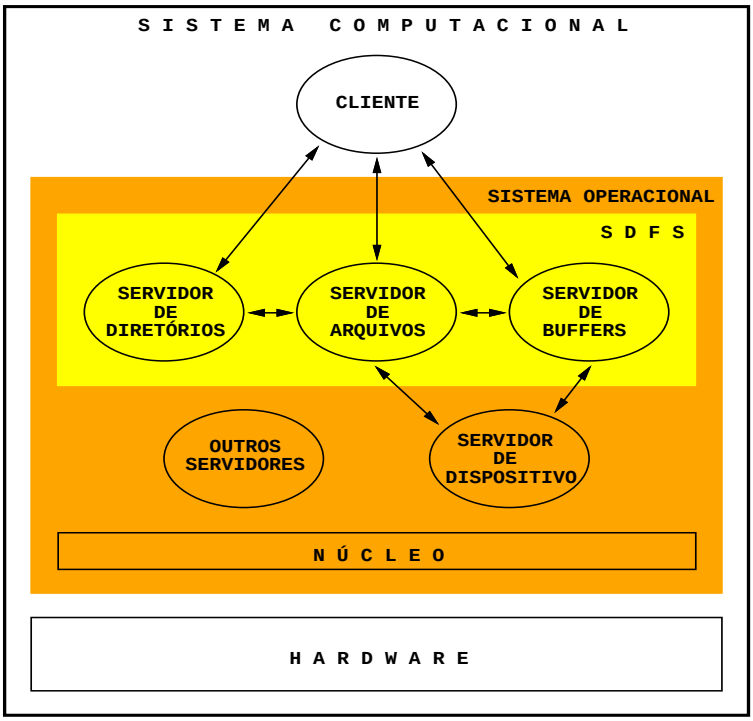

Figura 1: O SDFS em um sistema computacional

A distribuição das tarefas do sistema de arquivos em três servidores pode, por si só, trazer grande ganho de desempenho ao sistema. Se, por exemplo, o SDFS estiver executando em um multicomputador, pode-se ter o caso onde cada um dos servidores executa paralelamente em um processador, indubitavelmente elevando a taxa de execução de serviços do sistema de arquivos. Mesmo quando o SDFS estiver rodando concorrentemente em um único processador, existem vantagens que compensam a perda de desempenho causada pela troca de mensagens, como por exemplo a facilidade de substituir-se ou acrescentar-se servidores com o sistema em execução, sem necessidade de religações de código.

Pelo que foi descrito até o momento, não se pode fazer considerações sobre a execução paralela dos serviços do SDFS, uma vez que a maior parte deles é de natureza seqüencial. Considere o exemplo do serviço utilizado para se obter um descritor de arquivo: primeiramente o processo de usuário (cliente) envia uma mensagem para o servidor de diretórios solicitando a tradução do nome do arquivo para um identificador interno. Para efetuar a tradução, o servidor de diretórios solicita a leitura de diretórios, que são implementados como arquivos, ao servidor de arquivos. Isso termina bloqueando o servidor, não pela maneira como ocorre a comunicação, que é assíncrona, mas pela natureza seqüencial do serviço solicitado. O servidor de diretórios não pode responder a solicitação antes de ter recebido a resposta do servidor de arquivos.

Para permitir que os servidores atendam a mais do que uma solicitação de serviço ao mesmo tempo, decidiu-se implementar cada um deles como uma *task* com múltiplos *threads* [TAN 92]. Assim, sempre que um serviço puder bloquear um servidor por tempo considerável, ele será executado por um novo *thread*. Mais especificamente, um novo *thread* é gerado sempre que um serviço envolver alguma operação de entrada e saída. Então, um servidor é uma *task* com um *thread* básico que faz a inicialização das estruturas de dados e fica em um laço, recebendo mensagens de solicitação de serviços, chamando as respectivas funções (que podem ou não gerar um novo *thread*) e, dependendo da situação, enviando mensagens de resposta (em geral a resposta de um serviço é enviada pela função que o executa).

4 Comunicação e Localização dos Servidores

A comunicação entre clientes e servidores, bem como entre os próprios servidores está baseada no conceito de *port* [ACC 86]. Quando um processo deseja se comunicar com outros ele cria um ou mais *ports* que serão usados para receber e enviar mensagens. Desta forma, a entidade identificada na comunicação é o *port* e não o processo que o criou.

O uso de *ports* para a comunicação entre processos permite que se defina um espaço "bem conhecido" de *ports*, de forma similar aos WKS (*Well Known Services*) da INTERNET [SUN 92]. Esse conjunto engloba todos os serviços básicos do sistema, de tal forma que, quando um cliente deseje um serviço, ele conheça, a priori, o *port* correspondente. Da maneira como os WKS são implementados na INTERNET, se um dado serviço está disponível em um *host*, então o respectivo servidor também está disponível naquele *host*. Essa é uma restrição que contraria as metas definidas para o projeto, pois não permite que os servidores do SDFS executem em *hosts* distintos. Decidiu-se então expandir este mecanismo para que um *port* local possa estar associado a um servidor remoto.

Na primeira versão do SDFS o sistema de comunicação incluiu uma tabela de localização com uma entrada para cada "serviço bem conhecido". Cada entrada da tabela identifica até três servidores aptos a executar os serviços associados ao respectivo *port*. Quando um cliente solicita um serviço, ele o faz enviando uma mensagem a um *port*. Para proceder a entrega da mensagem, o sistema de comunicação consulta a tabela de localização em busca de um par (*host*, *port*) que identifica globalmente um servidor. Caso as três referências da tabela sejam nulas, uma mensagem indicando a indisponibilidade do serviço é retornada ao cliente. Caso alguma das entradas seja válida, o sistema de comunicação procederá ao envio da mensagem ao primeiro *port* indicado na tabela. Cada servidor tem associado um tempo máximo de execução do serviço, e assim, o sistema de comunicação pode, decorrido o limite de tempo sem que se tenha recebido uma resposta do servidor, proceder ao envio da mensagem ao próximo servidor da lista. Um exemplo de tabela de localização de servidores pode ser visto na tabela 1.

	SERVIDOR 0			SERVIDOR 1			SERVIDOR 2		
Serviço	Host	Port	Tmp	Host	Port	Tmp	Host	Port	Tmp
0	local	0	5	local	4	5	-	-	-
1	local	1	5	vega	1	50	juno	8	100
2	vega	2	80	-	-	-	-	-	-
:	:	:	:	:	:	:	:	:	:
n	local	n	5	vega	n	50	baco	n	60

Tabela 1: Tabela de localização de servidores

O mecanismo de localização de servidores descrito permite a replicação de servidores, uma vez que eles são idempotentes, bem como a definição de serviços equivalentes, pois um *port* pode ser redirecionado para outro no mesmo *host*. A tabela de localização de servidores pode ser alterada por meio de chamadas ao núcleo do sistema operacional, permitindo que, futuramente, um servidor externo faça reconfigurações dinâmicas na tabela, otimizando o tempo de resposta dos serviços.

A tabela de localização de servidores permite que um cliente localize um servidor de forma transparente, porém, o próprio sistema de arquivos está dividido em mais de um servidor, sendo necessário definir-se um mecanismo para que os servidores se localizem entre si. Como o SDFS não permite a replicação transparente de arquivos, definiu-se que sempre que um *host* contiver um arquivo, ele conterá também toda a coleção de servidores correspondentes ao arquivo. Desta forma, pode-se tirar do próprio identificador do arquivo a informação sobre a localização dos servidores, ou seja, assim que o servidor de diretórios obtém o identificador de um arquivo, as mensagens relativas a ele passam a ser enviadas diretamente ao *host* que o contém.

Um caso especial ocorre quando temos cada um dos servidores do SDFS executando em um processador particular de um multiprocessador. Neste caso, o multicomputador é visto como um único *host*, sendo que o espaço de "serviços bem conhecidos" é distribuído pelos processadores. O núcleo do sistema operacional ou o controlador de comunicações é responsável por dizer qual *port* pertence a qual processador.

Para manter a interface do sistema de comunicações homogênea, todo o endereçamento de mensagens compreende o par (*host*, *port*), sendo que quando o campo *host* for nulo, o sistema tentará definí-lo através de uma pesquisa na tabela de localização de servidores.

5 Distribuição dos Servidores

O SDFS foi desenhado de forma a possibilitar vários esquemas de distribuição, sem que se precise recompilar ou religar o sistema. Todas as alterações necessárias para reconfigurar o sistema se concentram no sistema de comunicações e não se propagam pelo resto do sistema. Existem três configurações de especial importância que serão discutidas a seguir: computadores isolados, multicomputadores e computadores conectados em rede.

5.1 Computadores Isolados

Apesar de ser um sistema de arquivos distribuído, nada impede que o SDFS rode em um único computador isolado. Neste caso, o sistema não apresenta qualquer nível de distribuição, pois todos os arquivos e servidores acessíveis são locais. Esta configuração pode ser vista na figura 2.

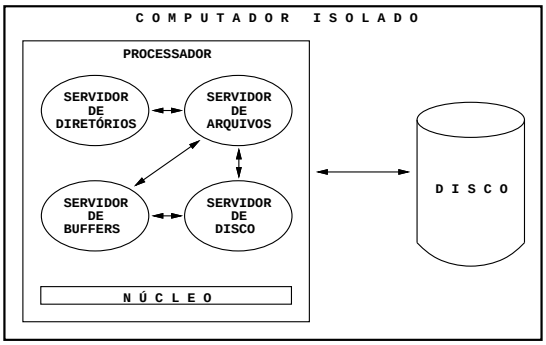

Figura 2: O SDFS em um computador isolado

5.2 Multicomputadores

Para a análise da distribuição do SDFS em um multicomputador, considerar-se-á que cada servidor executará em um processador distinto, o que pode ser visto na figura 3. Um multicomputador é visto como um único *host*, sendo que seus processadores compartilham um espaço comum de *ports*. Desta forma, qualquer solicitação de serviço será local, porém, envolvendo a troca de mensagens entre os nodos processadores. A responsabilidade de associar um determinado *port* a um nodo é responsabilidade do sistema de comunicações do multicomputador.

Reconfigurações dinâmicas do multicomputador, como migrações de processos, podem ser feitas, no que diz respeito ao SDFS, simplesmente redefinindo-se as tabelas de localização de servidores em cada um dos nodos ou no nodo de controle, dependendo da arquitetura da máquina.

5.3 Redes de Computadores

Um configuração comum para o sistema é distribuí-lo através de *hosts* conectados por uma rede de computadores. Para tal, basta definir-se o estado inicial das tabelas de localização de servidores de acordo com a distribuição desejada.

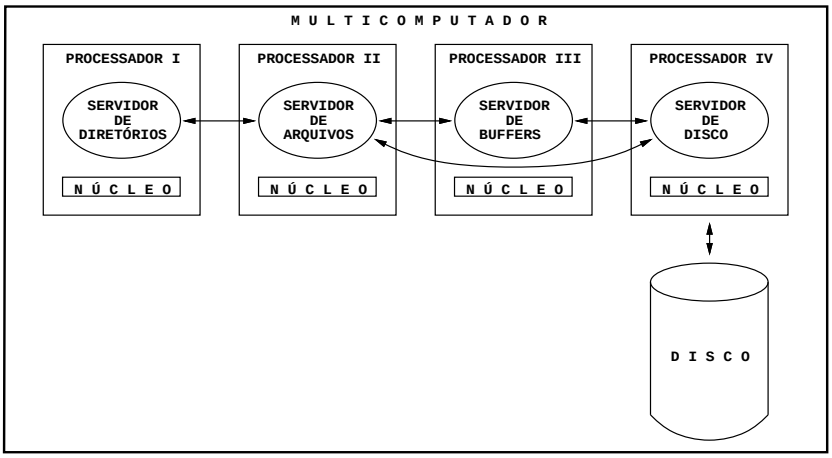

Figura 3: O SDFS em um multicomputador

Duas configurações são de especial interesse: *hosts* sem disco, que acessam exclusivamente arquivos remotos; *hosts* com disco que compartilham arquivos com outros *hosts*.

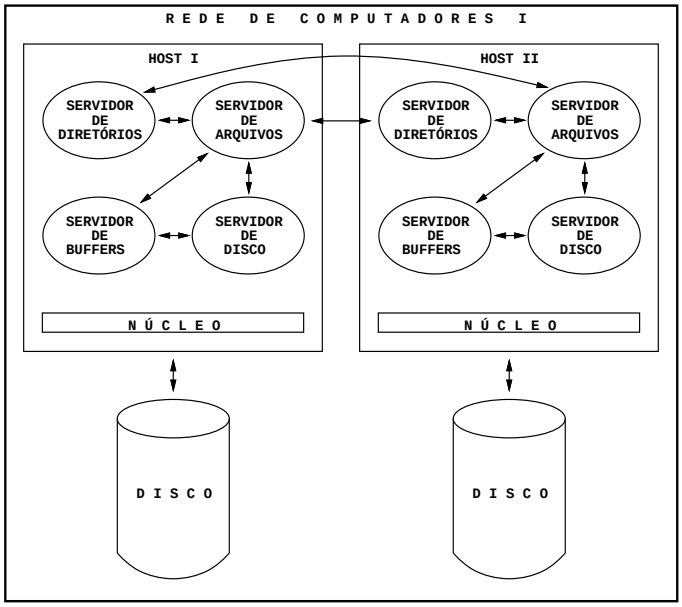

Figura 4: O SDFS em uma rede de computadores

A figura 4 mostra o SDFS configurado de forma a permitir o compartilhamento de arquivos entre dois *hosts* de uma rede. Nesta primeira versão do SDFS, um serviço sobre um arquivo remoto é encaminhado ao *host* que contém o arquivo tão logo se conheça a sua localização. Por exemplo, um cliente no *host* I solicita o serviço "open" para um arquivo localizado no *host* II. A solicitação é enviada ao servidor de diretórios do *host* I, que retornará o identificador do arquivo do *host* II. Se o cliente solicitar agora o serviço "read" fornecendo o identificador previamente obtido, a mensagem será enviada diretamente para o servidor de arquivos do *host* II. Isso acontece devido ao mecanismo de localização de servidores descrito anteriormente, que extrai do próprio identificador do arquivo a localização do respectivo servidor.

Uma segunda técnica para manter o serviço no *host* local até o nível físico poderia ser implementada dando-se preferência aos servidores locais. Desta forma, a cadeia de solicitações de serviço gerada por um cliente seria local

até o nível dos servidores de dispositivo, implicando na buferização local de dados remotos e, conseqüentemente, aumentando o desempenho do sistema. Essa técnica não foi adotada pelo SDFS por dois motivos: a complexidade do sistema aumentaria bastante, pois seria necessário a definição de um esquema para garantir a propagação de um bloco de dados alterado em um servidor de *buffers* para os outros servidores de *buffers* que mantivessem uma cópia do mesmo bloco; e o crescente aumento da velocidade de transmissão de dados em redes locais atenua e talvez até compense a perda de desempenho.

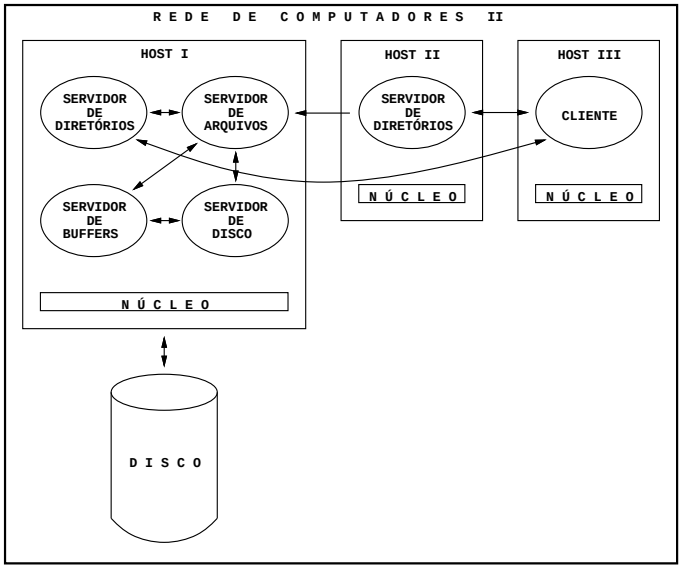

Figura 5: O SDFS em uma rede de computadores com *hosts* sem disco

A figura 5 mostra o SDFS rodando uma rede com três *hosts*, dois dos quais sem disco. A execução do servidor de diretórios em um *host* sem disco é opcional, caso decida-se executá-lo, deve-se fornecer um identificador de arquivo remoto para ser o diretório raiz da árvore de diretórios. Esse identificador deve ser obtido a priori, pois o servidor ainda não está executando, sendo impossível a tradução de nomes de arquivos. A vantagem de se rodar o servidor de diretórios em um *host* sem disco é a manutenção local de algumas estruturas, reduzindo um pouco o tráfego de mensagens entre os *hosts*. De qualquer maneira, toda a manipulação de dados de usuário será solicitada ao *host* que contém o disco. Outra opção é não se ter qualquer um dos servidores do SDFS executando localmente, bastando definir-se uma tabela de localização de servidores adequada. O *host* III da figura 5 exemplifica esta situação.

Executar os servidores de arquivos, de *buffers* ou de dispositivos em um *host* sem disco é absolutamente sem propósito. Nenhum identificador de arquivo fará referência a um *host* sem disco, logo, os servidores nunca receberão solicitações de serviço.

6 Distribuição da Árvore de Diretórios

Os clientes do SDFS podem solicitar serviços relativos a arquivos remotos fornecendo um *pathname* apropriado, entretanto, existem arquivos remotos que são freqüentemente acessados, tornando esse método inconveniente. Uma boa alternativa é a replicação de arquivos, todavia, se esses arquivos vierem a ser alterados em algum dos *hosts* envolvidos, a replicação pode trazer grandes dificuldades para que se possa garantir a integridade dos arquivos envolvidos. Além da consistência dos arquivos, memória secundária ainda é um recurso razoavelmente caro e geralmente disponível em menor quantidade do que se deseja. Isso justifica o esforço de definir-se um mecanismo de compatilhamento de arquivos entre *hosts*.

Uma das técnicas utilizada pelo SDFS para permitir que arquivos remotos sejam acessados transparentemente por vários *hosts* é similar a utilizada pelo NFS da SUN MICROSYSTEMS [SUN 92], onde a funcionalidade do serviço

"mount" é expandida para permitir que a árvore de diretórios seja acrescida de sub-árvores remotas. A outra técnica envolve *links* remotos. No SDFS é possível que um arquivo (diretório) remoto seja ligado à árvore local. Isso dispensa a necessidade de montar-se explicitamente uma sub-árvore remota, entretanto, degenera a estrutura da árvore de diretórios em um grafo, possivelmente, cíclico. A implementação dos diretórios do SDFS envolve identificadores globais de arquivos, permitindo que aplicativos como o "find" ou o "tar" distingam arquivos remotos de arquivos locais, minizando assim os efeitos negativos da ocorrência de ciclos no grafo de diretórios. Um exemplo de árvore de diretórios com *links* remotos pode ser vista na figura 6.

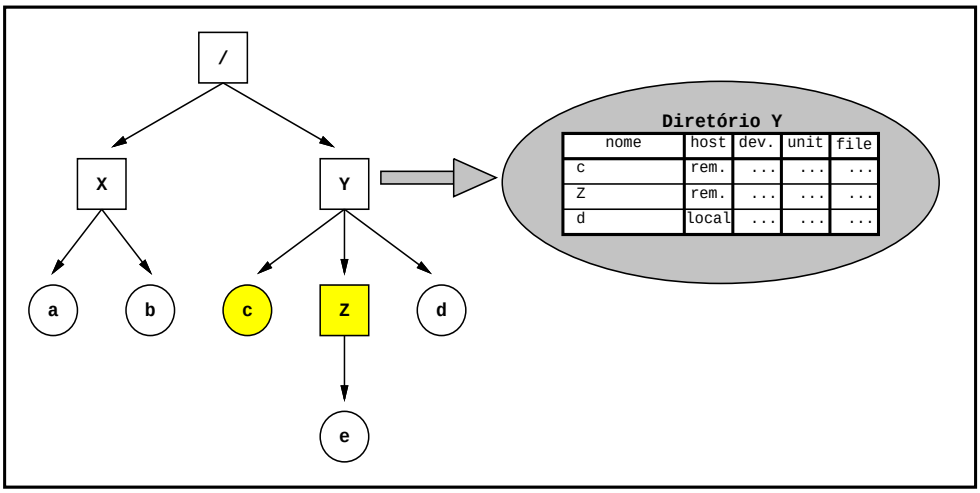

Figura 6: Árvore de diretórios com *links* remotos

7 Autenticação de Mensagens

Um problema grave no que diz respeito à segurança de sistemas distribuídos é a autenticação das mensagens de solicitação de serviço. No caso do SDFS, os servidores não guardam informações sobre o estado dos cliente, agravando bastante esse problema. Todas as mensagens enviadas aos servidores do sistema de arquivos contém o identificador do arquivo a ser operado. Sem um mecanismo seguro, os servidores não poderiam dizer se o identificador de arquivo foi obtido do servidor de diretórios com o respectivo serviço ou se ele foi arbitrado por um cliente descuidado ou mal intencionado.

Para autenticar uma mensagem, o SDFS faz uso de uma chave gerada pelo servidor de diretórios em resposta à solicitação de um serviço especial. Por ser idenpotente, o SDFS não implementa um serviço "open" convencional, resumindo esse serviço a obtenção do descritor de arquivo e da respectiva chave. Para solicitar qualquer outro serviço, o cliente deve apresentar o identificador do arquivo juntamente com a chave, que será usada para validar a solicitação. A chave é gerada como função do próprio identificador do arquivo, da data de criação do arquivo e do identificador do usuário.

A inclusão da data de criação do arquivo na composição da chave de autenticação se deu pelo fato dos servidores do sistema de arquivos não manterem uma tabela de arquivos abertos, mas sim uma *cache* com os descritores dos arquivos mais recentemente acessados. Assim, um cliente pode obter o identificador de um arquivo e a respectiva chave e ficar algum tempo sem acessar o arquivo, de forma que seu descritor seja removido da *cache*. Um outro cliente pode agora solicitar a remoção deste arquivo. É matematicamente possível que um outro arquivo venha a ser criado e receba o mesmo identificador. Aquele primeiro cliente que obteve o identificador do arquivo passará a acessar, indevidamente, um novo arquivo. Incluindo a data da criação do arquivo na função geradora da chave elimina-se esse problema.

O identificador do usuário foi envolvido na geração da chave de autenticação para evitar que um usuário abra um arquivo, obtenha o seu identificador e a chave, e transfira autorizações de acesso a um outro usuário originalmente

não autorizado. Considere um exemplo onde as autorizações de um arquivo permitam sua leitura por qualquer usuário do grupo do dono do arquivo, mas não pelos demais usuários. Um processo de um usuário do grupo do dono (cliente) poderia abrir o arquivo e mandar uma mensagem contendo o identificador e a chave a um processo de um usuário de fora do grupo do dono do arquivo, desvirtuando o esquema de controle de acesso aos arquivos.

8 Conclusão

Este artigo apresentou o SDFS (SIMPLE DISTRIBUTED FILE SYSTEM), um sistema de arquivos distribuído, portável, com alto grau de paralelismo e desenhado de forma a permitir sua execução em ambientes com diferentes níveis de distribuição. A proposta original desse sistema de arquivos descrita em [FRO 93] supunha a existência de um núcleo de sistema operacional que suportaria a execução de seus servidores. Esse núcleo se encontra em desenvolvimento no CPGCC/UFSC e só deverá estar operacional no fim deste ano. Decidiu-se então simular esse núcleo sobre um outro sistema operacional, o SUNOS 4.1.3 da SUN MICROSYSTEMS, o que impossibilitou a avaliação do desempenho do sistema, que ficou condicionado ao do sistema hospedeiro.

Os resultados até agora obtidos são muito satisfatórios. Existe uma versão completamente operacional do SDFS e a simplicidade e clareza de sua implementação permitirão que muito em breve o sistema seja migrado para um plataforma própria, possivelmente integrando um projeto cooperativo, envolvendo UFSC, UFRGS e UFSM, para o desenvolvimento de um multicomputador e de um ambiente de programação paralela sobre ele.

Referências

[ACC 86] ACCETTA, M. et al., *Mach: a New Kernel Foundation for UNIX Development*, Pittsburgh: Proccedings of the Summer 1986 USENIX conference, p. 93-112, julho de 1986.

[BAC 87] BACH, M., *The Design of the UNIX Operating System*, Englewood Cliffs: Prentice-Hall, 1987.

[COR 93] CORSO, T., *Ambiente para Programação Paralela em Multicomputador*, Florianópolis: UFSC/CTC/INE, 1993 (Relatório Técnico).

[FRO 93] FROHLICH, A., *Um Sistema de Arquivos Experimental.*, Florianópolis: UFSC, 1993 (Trabalho Individual).

[HOA 85] HOARE, C., *Communicating Sequential Processes*, Englewood Cliffs: Prentice-Hall, 1985.

[IEE 88] INSTITUTE OF ELETRICAL AND ELECTRONICS ENGINEER, *POSIX - Portable Operating System Interface for Computing Environments*, IEEE Computing Society, 1988.

[MUL 90] MULLENDER, S. J. et al., *Amoeba: a Distributed System for the 1990s*, IEEE Computer, p. 44-53, maio de 1990.

[POS 87] POSTEL, J. & REYNOLDS, J., *Official Internet Protocols*, Marina del Rey, USC, 1987 (RFC1011).

[RES] RESESSE, R. & TANENBAUM, A. & WILSCHUT, A., *The Design of a High-Performance File Server*, p. 72-82.

[SAT 89] SATYANARAYANAN, M., *A Survey of Distributed File Systems*, Annual Review of Computer Science, v.4, 1989.

[SUN 92] SUN MICROSYSTEMS, *Network Interfaces Programmer's Guide*, Mountain View: Sun Microsystems, 1992.

[TAN 92] TANEMBAUM, A., *Modern Operating Systems*, Englewood Cliffs: Prentice-Hall, 1992.