Desenvolvimento de um Detector de Defeitos para Sistemas Distribuídos baseado em Redes Neurais Artificiais

Nivea Ferreira[1], Raimundo Macêdo
Laboratório de Sistemas Distribuídos - LaSiD
Universidade Federal da Bahia - UFBA
{niveacf, macedo}@ufba.br

Resumo

Detectores de defeitos são mecanismos importantes para a implementação de sistemas distribuídos tolerantes a falhas, i.e., sistemas que garantam serviços continuados mesmo na presença de falhas. Em sistemas distribuídos assíncronos, sem limites de tempo conhecidos para a transferência de mensagens e/ou computação de ações locais, detectores de defeito perfeitos (ou confiáveis) não podem ser implementados. Nesses ambientes, falhas podem apenas ser suspeitadas. Uma forma de diminuir a ocorrência de falsas suspeitas é a utilização de *timeouts* adaptativos que possam ser calculados em função, por exemplo, da carga da rede de comunicação. Em [2] discutimos o uso desses *timeouts* adaptativos para a implementação do módulo CTI (C*onnectivity Time Indicator*) que, por sua vez, foi utilizado para implementar detectores de defeitos não confiáveis do tipo <>S [1]. Em outro artigo [11], mostramos como o CTI pode ser usado para, além de detectar defeitos, controlar os níveis de qualidade de serviço (QoS) de comunicação entre processos distribuídos, no nível da aplicação. No presente artigo mostramos uma implementação do módulo CTI através de Redes Neurais Artificiais que interagem com agentes SNMP (*Simple Network Management Protocol*)[5] e MIB (*Management Information Base*)[6] para predizer tempos de conectividade baseados nas características operacionais dinâmicas de uma rede IP (*Internet Protocol*) como congestão, perda de pacotes, etc.

Abstract

Failure detectors are mechanisms used to implement fault tolerant distributed systems, i.e., systems that can provide continuous services even when failures may occur. In the so-called asynchronous distributed systems, without bounded and known time delay for message transfer and local processing, perfect - or reliable - failure detectors can not be implemented. In such systems, failures can only be suspected. In order to avoid false suspicions, adaptive timeouts can be calculated based on, for instance, the communication network load. In [2] we discussed the use of such adaptive timeouts to implement a mechanism called CTI (Connectivity Time Indicator) which in turn was used to implement a failure detector of the class <>S [1]. In another paper [11], we showed how the CTI could be used to control the levels of quality-of-service (QoS) of communication time between distributed processes at the application level. In this paper we show an implementation of the CTI mechanism through artificial neural network which interact with SNMP (Simple Network Management Protocol)[5] agents and MIB (Management Information Base)[6] in order to predict communication times based on the dynamic operational conditions of an IP (Internet Protocol) network.

[1]Bolsista CNPq/ProTeM-CC pelo projeto ARGO (processo número: 381520/2000-5).

1. Introdução

Prover tolerância a falhas é fundamental para que as aplicações em sistemas distribuídos, principalmente aquelas de segurança crítica, garantam os seus serviços mesmo na presença de falhas de alguns de seus componentes. Um módulo básico de sistemas tolerantes à falhas é o detector de defeitos, que informa sobre processos que falharam.

Em sistemas distribuídos assíncronos, onde não há limites no atraso de entrega de mensagens e tempos de execução, não é possível distinguir entre processos falhos daqueles muito lentos. Por outro lado, esse modelo de sistema é de especial interesse devido ao fato de ser mais realista para computações distribuídas atuais, como por exemplo, na Internet. Chandra e Toueg estenderam o modelo puramente assíncrono com a noção de detectores de defeitos [1]. Estes mecanismos de detecção de falhas podem cometer erros, e, por isso, são chamados de detectores não-confiáveis, e são como um oráculo do estado de funcionamento de processos. Nesses sistemas, *timeouts* não podem ser tomados como uma indicação precisa de falhas.

Valores de *timeout* muito altos diminuem a possibilidade de haver uma falsa suspeita de falha, caso um processo esteja muito lento, mas impede que falhas reais sejam detectadas mais rapidamente. Para resolver o impasse que recai sobre o usuário, que nos algoritmos de detecção de falhas tradicionais deve sugerir os valores de *timeouts* a serem utilizados, surgem os detectores de defeitos adaptativos. Com valores adaptáveis de *timeout* é possível detectar-se falhas de maneira mais precisa, já que teremos valores mais adequados às características atuais do canal de comunicação. *Timeouts* adaptativos sugerem a determinação de valores mais coerentes com a sobrecarga da rede.

Em [2] discutimos o uso de timeouts adaptativos para a implementação do mecanismo chamado CTI (*Connectivity Time Indicator* ou Indicador de Tempo de Conectividade), que visa indicar dinamicamente os valores de conectividade entre os processos monitorados. O CTI foi utilizado para implementar detectores de defeitos não confiáveis do tipo <>S. Em outro artigo [11], mostramos como o CTI pode ser usado para, além de detectar defeitos, controlar os níveis de qualidade de serviço (QoS) de comunicação entre processos distribuídos, no nível da aplicação. Nos trabalhos citados, o CTI foi implementado usando-se o último tempo de *round-trip* das mensagens. No presente artigo mostramos uma implementação do modulo CTI através de Redes Neurais Artificiais que interagem com agentes SNMP (*Simple Network Management Protocol*)[5] e MIB (*Management Information Base*)[6] para predizer tempos de conectividade baseados nas características operacionais dinâmicas de uma rede IP (*Internet Protocol*) como congestão, perda de pacotes, etc.

Redes neurais artificiais são poderosas ferramentas quando considera-se algumas classes de problemas complexos [3, 4, 13]. Em nossa proposta, os valores de variáveis da MIB são utilizados como entradas e os valores de tempo de conectividade associados caracterizam a saída da rede neural.

Este trabalho integra o projeto ARGO, que objetiva a concepção e desenvolvimento de serviços de comunicação em grupo, a partir de uma abordagem que permita limitar e controlar as disfunções típicas de sistemas distribuídos assíncronos. O ARGO tem, portanto, na detecção de falhas nestes sistemas uma das principais motivações[2].

[2] Para maiores informações, acesse: http://www.lasid.ufba.br/projetos/argo.

Na próxima seção apresentaremos o CTI. A seção 3 fala sobre redes neurais artificiais e na seção 4 será explicada a implementação do nosso detector de defeitos. A seção 5 é reservada à apresentação dos resultados obtidos. Os trabalhos relacionados serão apresentados na seção 6. A conclusão e os próximos passos desta pesquisa serão apresentados na seção 7.

2. Indicador de Tempo de Conectividade (CTI)

O tempo de conectividade, *ct*, entre dois processos P_i e P_j, é definido como o tempo que uma mensagem leva para trafegar de P_i a P_j (ou vice-versa) num dado momento. Em sistemas como a Internet, o tempo de conectividade pode assumir valores distintos, podendo variar de 0 (se i = j) até infinito (se P_i e P_j estão desconectados).

Como é impossível predizer precisamente o futuro, o CTI deve sugerir o tempo de conectividade atual. Assim, existirá um módulo CTI sendo executado em cada nodo do sistema distribuído e ele estará atualizando constantemente as informações sobre conectividade dos processos locais e remotos.

Os *timeouts* utilizados na detecção de defeitos são determinados pelos tempos de conectividade estabelecidos entre os processos.

Na próxima seção serão explicados os principais conceitos de redes neurais artificiais, que facilitarão o entendimento da implementação do módulo CTI, apresentada na seção 4.

3. Redes Neurais Artificiais

Genericamente, uma rede neural é composta por um número de unidades conectadas. Cada ligação tem um peso, um valor numérico, associado. Este pesos representam a capacidade de armazenamento/codificação de informação da rede neural, e o processo de aprendizado é associado à alteração desses pesos.

O modelo escolhido para a tarefa de implementação do CTI foi uma rede MLP (*Multilayer Perceptron*). Redes MLP são redes associativas formadas basicamente por:

- Uma camada que recebe os estímulos de entrada, sendo constituída, normalmente, por um número de neurônios correspondente à quantidade de atributos dos dados;
- Uma ou mais camadas intermediárias, cujas unidades recebem como dado de entrada os sinais provenientes de camada de entrada ou da camada intermediária anterior;
- Uma camada de saída que, recebendo as entradas da camada intermediária, chega a uma predição ou classificação.

Os pesos são modificados com base no erro verificado na saída da rede neural (aprendizado supervisionado). Mais detalhadamente: o estímulo de entrada é apresentado à rede e, então, propagado para as diversas camadas. Em cada unidade, as entradas são ponderadas pelos valores dos pesos, associados a cada uma das conexões, para determinar a ativação desta unidade. Com base nesta ativação, cada unidade transmite às unidades da camada seguinte, às quais está conectada, a sua saída. A saída produzida pela última camada é então comparada ao resultado esperado e o erro, a diferença entre o valor produzido e o esperado, é calculado. Esse erro é propagado de volta até alcançar as conexões que ligam a camada de entrada à primeira camada intermediária. Este aprendizado recebe o nome de *backpropagation*.

4. Implementação

Como dito anteriormente, a idéia principal do nosso processo de detecção de falhas em sistemas assíncronos considera que existirá um módulo CTI sendo executado em cada nodo, atualizando as informações sobre a conectividade dos processos distribuídos. Assim, a proposta deste trabalho é a utilização de redes neurais para implementar o módulo CTI, e realizar a tarefa de determinar *timeouts* que sejam mais adequados, de acordo com as características atuais do canal de comunicação.

O SNMP é o protocolo utilizado para fazer a interface entre os objetos gerenciáveis da rede de comunicação (descritos na MIB) e a rede neural. Utilizando o SNMP, os objetos são lidos a partir de:

- Grupo de Interface, que representa a interface de rede
- Grupo IP, que representa o *Internet Protocol* , e
- Grupo UDP, que representa o *User Datagram Protocol*.

Estes diferentes grupos da MIB representam diferentes tipos de funcionalidade.

A rede neural MLP com aprendizado *backpropagation* foi implementada utilizando-se a linguagem Java. Como padrões de entrada desta rede neural temos os valores de variáveis da MIB, através dos quais podemos caracterizar o tráfego da rede de comunicação. Desta forma, o tempo de conectividade associado a cada padrão de entrada é a saída dada pela rede a cada instante.

Os dados utilizados para treinamento e teste da rede neural implementada foram coletados nos módulos do Laboratório de Sistemas Distribuídos, onde esta pesquisa está sendo desenvolvida. Para isso, foram utilizadas 4 máquinas.

5. Resultados

A nossa base de dados inicial é composta por 4697 padrões de entrada da rede[3], divididos em 6 arquivos de tamanhos variáveis (i.e., possuindo número diferente de padrões cada um). Foram feitas várias etapas de treinamento e teste, utilizando-se os dados destes arquivos.

O treinamento da rede neural é feito utilizando-se um dos arquivos, sendo o número de ciclos de treinamento determinado por um erro mínimo especificado. Tendo encerrado o treinamento, o desempenho da rede é verificado utilizando-se um outro arquivo de dados, diferente daquele utilizado para o treinamento. Ou seja, na fase de testes são apresentados à rede neural padrões desconhecidos. A avaliação da precisão dos valores sugeridos como saída da rede neural são comparados com os valores reais, registrados durante a coleta dos dados.

A Figura 1 mostra o tempo de conectividade associado a cada padrão registrado. Estes padrões são referentes a um arquivo de teste. Na Figura 2 são mostradas as saídas da rede neural usando os padrões registrados no arquivo de teste como padrões de entrada. Comparando-se os gráficos, os valores sugeridos pela rede se aproximam e muito dos valores reais observados.

[3]Que são os valores coletados nos objetos da MIB.

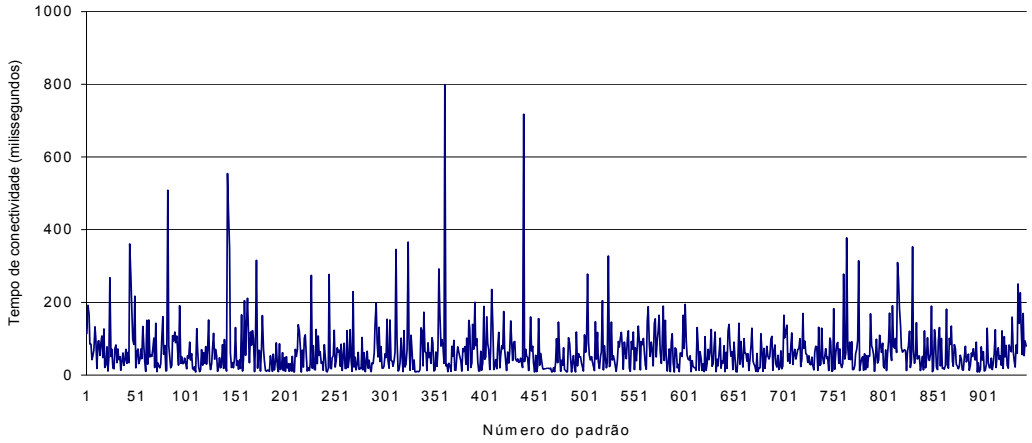

Figura 1: Tempo de conectividade real relacionado a cada padrão.

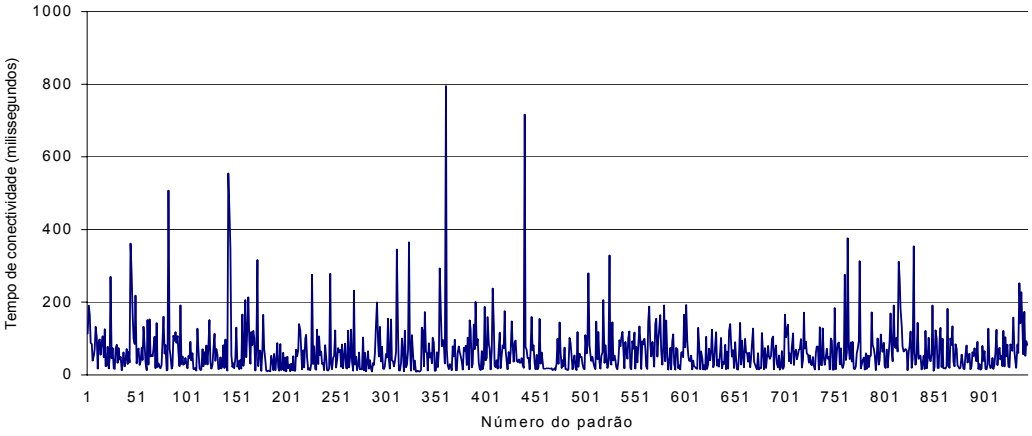

Figura 2: Tempo de conectividade sugerido pela rede neural relacionado a cada padrão.

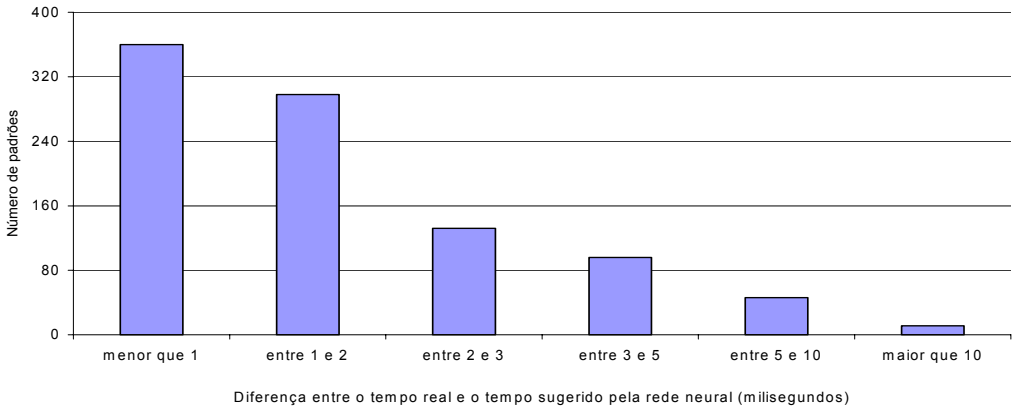

Figura 3: Classificação dos padrões quanto à diferença entre o tempo real e o tempo sugerido pela rede neural.

A Figura 3 apresenta a diferença dos valores de tempo de conectividade sugeridos pela rede neural e os valores reais, apresentados nos gráficos anteriores. Considerando os dados

descritos neste gráfico, 99% desta diferença está entre 0 e 10 milissegundos, o que consideramos como um desempenho satisfatório.

6. Trabalhos Relacionados

O trabalho de Hood e Ji [10] utiliza a abordagem de redes bayesianas para monitoração de falhas de sistemas. Essa abordagem está baseada na implementação de agentes SNMP inteligentes, que aprendem o comportamento anormal da rede e combina isto com a modelo probabilístico da rede bayesiana. No nosso trabalho, visamos detectar defeitos de processos, definidas com base no uso de *timeouts*, trazendo uma nova concepção de cálculo destes valores de modo a torná-los adaptativos. O uso de variáveis da MIB, neste caso, auxilia-nos na identificação de mudanças no sistema que caracterizariam as suspeitas de defeitos.

Uma outra abordagem é apresentado em [9], que descreve um algoritmo, chamado DPCP, para cálculo de *timeouts* e intervalos de monitoração adaptativos. Este algoritmo tem como idéia principal o uso do tempo atual, descartando-se os anteriores, para calcular estes valores. Isto difere da nossa abordagem uma vez que não considera as características de sobrecarga da rede para este cálculo e pelo fato de não considerar valores anteriores de tempo de conectividade.

Em [12] foi descrito um conjunto de métricas para especificar detectores de defeitos para sistemas com comportamentos probabilísticos e apresentado o algoritmo que será analisado de acordo com estas métricas. O algoritmo adaptativo deve ser capaz de se reconfigurar de acordo com as mudanças nas características probabilísticas de rede. Os detectores de defeito, neste caso, não são definidos como nos sistemas assíncronos, pois as aplicações com restrições de tempo requerem detectores que possam prover qualidade de serviço com garantias quantitativas de tempo. Ou seja, neste artigo são estudados os sistemas onde a perda e o atrasos das mensagens seguem distribuições probabilísticas.

O trabalho apresentado em [13] é uma proposta de uso de redes neurais para um controle adaptativo de qualidade de serviço (QoS) para redes ATM. Para garantir parâmetros de qualidade de serviço como tempo de atraso de células ou probabilidade de perda de células, é necessário gerenciamento de tráfego e controle de congestão. A idéia neste artigo é utilizar redes neurais para estimar QoS para o controle de admissão de chamadas. Para tal, a rede neural usa a combinação do número de conexões usando áudio, vídeo e serviços de comunicação de dados, sendo o número de entradas da rede determinada principalmente pelo número de parâmetros de tráfego de cada umas destas categorias. Neste artigo são mostrados os resultados obtidos por uma rede MLP.

Nos sistemas de tempo-real com altas restrições de recurso, o mecanismo de tolerância deve se adequar aos recursos disponíveis e às condições do ambiente. O objetivo do trabalho apresentado em [14] é a operação autônoma de funções críticas de espaçonaves, isto é, executá-las de maneira autônoma sem que haja perda significativa de performance ou de segurança, mesmo na presença de eventos indesejáveis. Mais detalhadamente, o objetivo do mecanismo de adaptação é minimizar a utilização de recursos, mantendo os requisitos de confiabilidade; deve ser genérico o suficiente para lidar com vários tipos de falhas (de sensor, de processador, de aplicativos); responder às mudanças de ambiente, missão, fases, estados do sistema e perfis de usuário, que provê os parâmetros da aplicação que afetam a tomada de decisão sobre a adaptação. Para isso, o mecanismo usa informações sobre os recursos

disponíveis, a demanda do ambiente e um histórico de falhas recentes, como indicativo da probabilidade de falhas futuras.

7. Conclusões

O uso t*imeouts* é comum em detecção de falhas. Valores de *timeouts* constantes comprometem a eficiência dos detectores de defeitos, uma vez que não considera as variações na sobrecarga da rede, que interferem diretamente nos tempos de resposta dos processos ativos (isto é, não-falhos). Valores baixos são responsáveis por suspeitas incorretas de processos corretos como sendo falhos, enquanto valores altos adiam a suspeita de ocorrência real de falhas de processos.

A partir da necessidade de definição de valores mais coerentes com as características correntes da rede e da comunicação entre processos, surgem os *timeouts* adaptativos. Tendo valores que estão de acordo com essas características, busca-se minimizar a ocorrência de erros na detecção de falhas em sistemas distribuídos assíncronos, dado que nestes sistemas, devido às suas características, valores de *timeouts* são aproximativos.

O uso de redes neurais para a implementação do mecanismo do CTI com características adaptativas apresentou ótimos resultados. O nível de acerto dos tempos de conectividade sugeridos pela rede neural se aproximaram, com grande precisão, dos tempos de conectividade registrados.

Com base nestes resultados, estamos iniciando uma nova fase que se caracteriza pela integração deste módulo de detecção de falhas aos outros módulos que compõem a plataforma ARGO de tolerância a falhas.

Além dessa integração, considerando a característica estática da rede neural explicada anteriormente, decidimos verificar o comportamento de uma rede que possuísse características dinâmicas e fosse adequada, portanto, para predição de séries aleatórias [8]. Desta forma, seria possível realizar a tarefa de predição de valores de conectividade, ou seja, mais do que valores atuais, seríamos capazes de determinar valores em instantes futuros. Neste caso, é necessário estender a arquitetura de redes neurais descrita anteriormente.

Novos testes estão sendo realizados para atestar a capacidade preditiva deste tipo de topologia de rede neural. Os resultados já obtidos mostraram-se bastante interessantes, indicando ser esta um linha de pesquisa com um chance concreta na obtenção de importantes resultados.

8. Agradecimentos

Ao CNPq, que através do ProTeM-CC, viabilizou o desenvolvimento desta pesquisa.
À Fábio Lima, aluno da Graduação do Curso de Ciência da Computação, que realizou a tarefa de implementação dos agentes SNMP.

Referências

[1] Chandra T., Hadzilacos V. and Toueg S., *The weakest Failure Detector for Solving Consensus*. Journal of the ACM, 43(4): 685-722, July 1996.

[2] Macêdo R., *Failure Detection in Asynchronous Distributed Systems*. Proc. of II Workshop on Tests and Fault-Tolerance, pp. 76-81, July 2000, Curitiba, Brazil.

[3] Rietman, E.A. and Frye, R.C. Neural Control of a Nonlinear System with Inherent Time Delays. Conference on Analysis of Neural Network Applications, pp.140-145, 1991.

[4] Chow, M. and Yee, S.O. Real Time Application of Artificial Neural Networks for Incipient Fault Detection of Induction Machines. Proceedings of the 3rd International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 90), pp. 1030-36, Charleston, USA, July 1990.

[5] Case, J., Fedor, M., Schoffstall, M., Davin, J. Connected: An Internet Encyclopedia – A Simple Network Management Protocol at www address http://deese.univ-lemans.fr:8003/connected/RFC/1157/index.html

[6] McCloghrie, K., Rose, M. Connected: An Internet Encyclopedia – Management Information Base for Network Management of TCP/IP-based internets: MIB-II at www address http://deese.univ-lemans.fr:8003/connected/RFC/1213/index.html

[7] Russell, S. and Norvig, P. *Artificial Intelligence- A Modern Approach*. 1st ed. New Jersey, Prentice-Hall, 1995.

[8] Haykin, S. *Neural Networks – A Comprehensive Foundation*. 1st ed. New York, Macmillan, 1994.

[9] Sotoma, I. and Madeira, E.R.M. DPCP (Discard Past Consider Present) – A Novel Approach to Adaptive Fault Detection in Distributed Systems. 8th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'2001), vol.1, pp.76-82, Bologna, Italy, October 2001 .

[10] Hood, C. S. and Ji, C. Intelligent Agents for Proactive Fault Detection. Internet Computing, v.2, n.2, pp.65-72, March/April, 1998.

[11] Batalha, M. and Macêdo R.J.A.Um Serviço Tolerante a Falhas para o Gerenciamento de Sistemas Distribuídos Sobre CORBA. Proceedings of the Latin-American Conference on Informatics (CLEI'2001). Mérida, Venezuela. September/2001.

[12] Chen W., Toueg, S. and Aguilera, M.K. On the Quality of Service of Failure Detectors. Proceedings of the International Conference on Dependable Systems and Networks (DSN 2000), New York, June 2000.

[13] Hiramatsu, A. Training Techinques for Neural Network Applications in ATM. IEEE Communications Magazine, pp.58-67, October 1995.

[14] Shokri, E. and Beltas, P. An Experiment with Adaptive Fault Tolerance in Highly-Constraint Systems. Proceedings of the Fifth International Workshop on Object-Oriented Real-Time Dependable Systems, California, USA, November 1999.