**Christiano Farina Haesbaert**

# *OpenMdns - Proposta TCC*

Porto Alegre - Rio Grande Do Sul, Brasil

22 de novembro de 2010

**Christiano Farina Haesbaert**

# *OpenMdns - Proposta TCC*

Orientador:
Ana Cristina Benso da Silva

Pontifícia Universidade Católica do Rio Grande Do Sul

Porto Alegre - Rio Grande Do Sul, Brasil

22 de novembro de 2010

# *Sumário*

**Lista de abreviaturas e siglas**

**Lista de Tabelas**

**1 Introdução** p. 5

**2 Protocolos** p. 7

2.1 MDNS . . . . . p. 7

2.2 DNS-SD . . . . . p. 8

2.3 Implementações Existentes . . . . . p. 10

**3 Proposta** p. 11

3.1 Objetivo . . . . . p. 11

3.2 Status . . . . . p. 13

3.3 Atividades Previstas . . . . . p. 14

**Referências** p. 16

# *Lista de abreviaturas e siglas*

| | | |
|---|---|---|
| IPv4 | Internet Protocol version 4, | p. 4 |
| NTP | Network Time Protocol, | p. 8 |
| FTP | File Transfer Protocol, | p. 8 |
| TCP | Transmission control Protocol, | p. 8 |
| FSF | Free Software Foundation, | p. 8 |
| LGPL | GNU Lesser General Public License, | p. 9 |
| IP | Internet Protocol, | p. 4 |
| DNS | Domain Name System, | p. 4 |
| TCP/IP | Internet Protocol Suite, | p. 4 |
| MDNS | Multicast DNS, | p. 4 |
| DNS-SD | DNS Service Discovery, | p. 4 |
| ISC | Internet Systems Consortium, | p. 5 |
| IETF | Internet Engineering Task Force, | p. 6 |
| SSH | Secure Shell, | p. 8 |

# *Lista de Tabelas*

1 Estado da Implementação das Seções do Draft MDNS(1) . . . . . p. 15

2 Estado da Implementação das Seções do Draft DNS-SD(2) . . . . . p. 15

# 1 Introdução

Os dispositivos de redes estão cada vez menores e mais portáveis, e as redes de computadores estão presentes em praticamente todo lugar, de residências à grandes corporações. Gradualmente a ideia de que cada dispositivo é um elemento estático em uma rede previamente configurada vem se tornando cada vez mais remota. Sendo assim deseja-se comunicação entre dispositivos com a menor infra estrutura possível, em especial, em ambientes nos quais não há um contingente técnico, como por exemplo uma residência familiar.

Por exemplo, uma funcionalidade desejável em uma rede IPv4 (*Internet Protocol version 4*) é a de mapear *hostnames* em endereços IP (*Internet Protocol*) sem a necessidade de um servidor de DNS (*Domain Name System*) (3) local ter sido previamente configurado nos *hosts* participantes da rede. Uma segunda funcionalidade que revela-se interessante é a de realizar enumeração de serviços (*Network Browsing*), para que um *host* possa pesquisar quais serviços são oferecidos na rede local. Estas são funcionalidades desejáveis em qualquer rede IPv4, seja em uma residência, escola ou corporação, portanto é importante tê-las com o mínimo esforço possível.

Pode-se enumerar então duas funcionalidades desejáveis na rede local:

1. Resolver nomes de DNS sem configuração prévia e/ou servidor central.
2. Realizar enumeração de serviços.

Ambas funcionalidades não são fornecidas pela suite TCP/IP (*Internet Protocol Suite*), e portanto foram propostos dois protocolos que geralmente operam em conjunto, MDNS (*Multicast DNS*)(1) e DNS-SD (*DNS Service Discovery*)(2).

O **MDNS** fornece a primeira funcionalidade, este é basicamente uma forma de realizar requisições de DNS via *multicast*, obtendo uma resposta também via *multicast*, neste cenário cada *host* é um agente de MDNS autonômo não existindo um servidor central como no DNS convencional.

O **DNS-SD** nos fornece a segunda funcionalidade e permite que através de requisições de DNS seja possível realizar a enumeração de serviços. Ele é compatível com DNS convencional (*unicast*) e MDNS, sua popularização no entanto, deu-se pelo uso em conjunto com o MDNSD.

Os protocolos MDNS/DNS-SD geralmente operam como protocolos complementares, com eles é possível por exemplo perguntar quais impressoras estão *online* ou qual o endereço IP de um *host* qualquer, ou até mesmo perguntar quais computadores possuem um serviço de compartilhamento de músicas.

O interesse em abordar tais protocolos neste trabalho deve-se ao fato de que diversas redes, em sua maioria pequenas, são limitadas em sua funcionalidade pela falta de infra estrutura, pois não se pode esperar que toda rede possua um servidor de DNS central e que os usuários comuniquem uns aos outros como acessar os serviços fornecidos pelos seus computadores. Ao pesquisar sobre o assunto, encontrou-se como alternativa o MDNS/DNS-SD e percebeu-se que até então não existia uma implementação viável do mesmo para o sistema operacional OpenBSD(4), este que não poderia suportar totalmente as implementações existes por divergências quanto a licença(5), filosofia e características técnicas.

Portanto, a proposta deste trabalho é desenvolver uma implementação de MDNS/DNS-SD para o OpenBSD com licença(6) ISC (*Internet Systems Consortium*). Espera-se que esta atenda as expectativas e filosofias da equipe do sistema operacional. Propõem-se que o fruto deste trabalho poderá ser fornecido abertamente sob a licença ISC em prol da comunidade dos usuários e equipe do OpenBSD.

# 2 Protocolos

## 2.1 MDNS

O MDNS e seu protocolo companheiro DNS-SD foram criados para fornecer a praticidade e conveniência encontrada anteriormente no protocolo AppleTalk(7). Quando a Apple realizou sua transição de AppleTalk para IPv4, esta desejava ter as funcionalidades encontradas no protocolo AppleTalk na rede IPv4(7), tais como maior autonomia dos *hosts* e enumeração de serviços. Assim sendo os engenheiros Stuart Cheshire e Marc Krochmal desenvolveram os protocolos MDNS e DNS-SD. Ambos ainda possuem status de *Internet Drafts* junto a IETF (*Internet Engineering Task Force*), tendo sua expiração para 23 de Setembro de 2010, mas já foram implementados em diversos sistemas operacionais.

No protocolo MDNS, são feitas requisições via *multicast* que possuem validade apenas no enlace local. Para que isto seja possível foi definido o domínio *.local* para o uso exclusivo no enlace local, este domínio possui semântica especial e é o domínio utilizado para toda a comunicação no MDNS. Os *hosts* do enlace local ao receberem uma requisição a analizam e emitem uma resposta caso esta seja apropriada. Esta resposta também é enviada via *multicast* e faz com que todos outros *hosts* partilhem deste conhecimento. Os dados recebidos são armazenados em uma cache em todos os participantes da rede, fazendo com que cada *host* evite emitir requisições que já foram respondidas recentemente, diminuindo portanto o tráfego da rede. Este procedimento ajuda a diminuir o tráfego da rede, mas exige um protocolo de coerência de cache(1) e outras complicações.

Não há nenhum tipo de alteração quanto a estrutura das mensagens de DNS *unicast* e dos *Resource Record* disponíveis. É possível realizar *qualquer* tipo de requisição DNS via *multicast*, isto torna-se interessante a medida que permite um certo nível de interoperabilidade com clientes de DNS *unicast*, chamados clientes legado. O fato de utilizar um protocolo antigo e maduro como o DNS é um benefício sob os seguintes pontos(1):

1. Tecnologia madura e bem documentada.

2. Diversos livros foram escritos sobre o assunto.

3. Grande número de implementações de diferentes grupos e vendedores.

4. Familiaridade dos implementadores com o protocolo.

5. As diversas falhas e *bugs* já foram enderaçadas e são bem conhecidas.

Um exemplo do uso do MDNS é apresentado a seguir. Supõem-se que existam três *hosts*: *bilbo.local*, *frodo.local* e *gandalf.local*. Neste exemplo o *host bilbo.local* está interessado em descobrir o IP de *frodo.local*, portanto, assim como no DNS convencional ele está interessado no *Resource Record* de tipo A cujo nome é *frodo.local*. Para mapear o nome em endereço os seguintes passos seriam executados:

1. O *host bilbo.local* envia uma requisição via multicast pelo *Resource Record* tipo A, de *frodo.local*.

2. Os *hosts frodo.local* e *gandalf.local* recebem e processam a requisição.

3. Apenas o *host frodo.local* responde a requisição, pois este é o detentor do nome *frodo.local*.

4. Tanto *bilbo.local* quanto *gandalf.local* armazenam a resposta em seu *cache*. Futuramente ambos irão consultar seu cache antes de realizar novamente esta mesma requisição, diminuindo o tráfego na rede.

## 2.2 DNS-SD

O DNS-SD é um protocolo que pode ser utilizado tanto com DNS convencional como com MDNS, porém ele é quase que exclusivamente utilizado com MDNS. O protocolo especifica uma convenção para estrutura de nomes dos *Resource Record* já existentes, com isto é possível se obter duas importantes operações:

1. Enumeração de serviços (*Network Browsing*).

2. Resolução de serviços.

Na enumeração de serviços é possível descobrir todos os serviços que estão sendo anunciados na rede. Para que isto ocorra, cada *host* envia pacotes de MDNS na rede local

perguntando sobre o serviço desejado, os *hosts* que possuem o mesmo irão responder com uma lista das suas instâncias do serviço. Do ponto de vista do protocolo MDNS, este é apenas um conjunto de requisições e respostas como qualquer outra. É na convenção dos nomes que o DNS-SD se aplica, tais nomes são convencionados de uma forma que torna possível a enumeração de serviços, como por exemplo servidores de SSH (*Secure Shell*), NTP (*Network Time Protocol*), FTP (*File Transfer Protocol*), entre outros. Quando o usuário decide procurar um determinado serviço, ele envia uma requisição via *multicast* com o nome do serviço na convenção do DNS-SD, este então recebe uma lista com as instâncias do mesmo.

A resolução de serviços toma parte após a enumeração. Depois do usuário receber a lista com todas instâncias do serviço, ele escolhe qual instância deseja acessar e envia uma requisições pedindo a resolução da mesma. Na resposta há instruções de como acessar a instância, tais como porta TCP (*Transmission Control Protocol*, nome de usuário e outras informações.

Um exemplo exemplo do uso de MDNS/DNS-SD é apresentado a seguir. Supõem-se que existam quatro *hosts*: *bilbo.local*, *frodo.local*, *gandalf.local* e *elendil.local*. Neste exemplo o *host bilbo.local* está interessado em descobrir todos os servidores de FTP existentes em sua rede, os passos seguidos seriam os seguintes:

1. O *host bilbo.local* envia uma requisição solicitando a **enumeração** de todos os serviços do tipo FTP, no caso, todos os servidores de FTP que estão sendo anunciados via MDNS/DNS-SD.

2. O *host gandalf.local* não possui tal serviço, portanto não responde. Já os *hosts frodo.local* e *elendil.local* enviam uma resposta anunciando seus respectivos servidores de FTP, com os nomes 'frodoFTP' e 'elendilFTP'.

3. O *host bilbo.local* agora conhece dois serviços de FTP, 'frodoFTP' e 'elendilFTP'. Ele opta por acessar o serviço 'elendilFTP' enviando uma requisição para **resolver** o mesmo.

4. O *host elendil.local* envia uma resposta informando porta, usuário e senha do servidor.

## 2.3 Implementações Existentes

Existem apenas duas implementações totais de MDNS/DNS-SD, o **Bonjour**(8) da Apple e o **Avahi**(9) da FSF (*Free Software Foundation*).

A Apple sendo a criadora dos dois protocolos forneceu a primeira implementação sob o nome de Bonjour, foi implementada pelo próprio criador do protocolo Stuart Cheshire. É a implementação MDNS/DNS-SD utilizada no MACOS-X e recentemente adotada pelo NetBSD(10), é licenciada sob a licença Apache-2(11). Seu código é bastante portável e com algum esforço deve funcionar em qualquer UNIX, também há um *port* para o Windows, possui cerca de 60 mil linhas de código.

O Avahi é a alternativa fornecida pela FSF sob licença LGPL (*GNU Lesser General Public License*), seu código é uma continuação do abandonado projeto Howl, seus principais autores são: Lennart Poettering, Trent Lloyd e Sjoerd Simons, fornece compatibilidade binária com Bonjour e seu código é também bastante portável. É reconhecido como uma excelente implementação por Stuart Cheshire, este citando até mesmo que no futuro talvez a Apple abandone o Bonjour em favor do Avahi. Possui cerca de 45 mil linhas de código.

# 3 Proposta

## 3.1 Objetivo

O objetivo deste trabalho é desenvolver uma implementação de MDNS/DNS-SD para o sistema operacional OpenBSD licenciado sob a licença ISC. Propõem-se implementar os requisitos descritos nos *drafts* do MDNS(1) e DNS-SD(2). O resultado da implementação deverá fornecer meios para que um programa qualquer possa:

1. Realizar consultas básicas de MDNS.
2. Realizar enumeração e resolução de serviços de DNS-SD.
3. Publicar serviços de DNS-SD.

A motivação de uma nova implementação, ao invés de fazer o *port* de uma implementação existente, deve-se ao fato de que nenhuma destas pode ser aceita na base do OpenBSD, por não possuírem licença compatível com o sistema e por não partilharem dos mesmos ideais técnicos e filosóficos. No OpenBSD apenas projetos com licença BSD ou mais livres que BSD são aceitos na base(5). As licenças GPL/LGPL como no caso do Avahi são aceitas apenas em último caso e em componentes críticos nos quais não há alternativa viável.

O OpenBSD presa por implementações pequenas e simples, sendo que o Avahi e o Bonjour são implementações grandes. Na visão da equipe do sistema operacional isto é visto como *bloat*[1], como citado pelo fundador do OpenBSD Theo De Raadt em entrevista ao site *www.thejemreport.com*(12):

> Some people think we hate GNU code. But the thing is we hate large code, and buggy code that upstream does not maintain. That's the real problem...

[1] A produção de código excessivo percebido como desnecessáriamente longo, lento ou um desperdício de recursos computacionais.

> gcc gets about 5-6 per cent slower every release, has new bugs, generates crappy code, and drives us nuts. This is just an attempt to see if something better can show up.

Implementações com um maior número de linhas de código tendem a ter mais *bugs* e possuir menos qualidade que uma equivalente com menos linhas, sendo geralmente o número de *bugs* proporcional ao número de linhas(13)(14). Portanto, a inclusão de novas funcionalidades é um processo cauteloso.

Neste trabalho é proposto uma implementação menor e mais simples que as alternativas existentes, atendendo assim as expectativas do OpenBSD.

O nome escolhido para esta implementação foi OpenMDNS seguindo a linha de outros projetos parceiros do OpenBSD, como OpenBGP(15), OpenOSPF(16) e OpenSSH(17).

Pelo estudo já realizado, propõem-se que a arquitetura do OpenMDNS seja dividida em três partes:

1. Um *daemon*[2] chamado **mdnsd** que implementa os protocolos MDNS/DNS-SD.
2. Uma biblioteca chamada **libmdns** para a comunicação dos programas com o **mdnsd**.
3. Um programa de controle chamado **mdnsctl** para realizar requisições simples e para *debug* do **mdnsd**.

Assim, todo programa que desejar utilizar as funcionalidades do MDNS/DNS-SD deve referenciar a biblioteca **libmdns** para a comunicação com o **mdnsd**, este que é responsável pela comunicação na rede local e de implementar as funcionalidades do MDNS/DNS-SD.

A implementação deve ser portável a medida que deve funcionar nas arquiteturas suportadas pelo sistema OpenBSD, sendo assim devidos cuidados devem ser tomados para que não se assuma características pertinentes a apenas uma arquitetura. Uma vantagem obtida ao se garantir que o código funcione em todas arquiteturas é o fato de que nem todos *bugs* manifestam-se da mesma maneira em máquinas distintas. Por exemplo uma premissa da arquitetura sparc64 é que todo acesso a memória deve ser alinhado, arquiteturas como i386 e ARM não consideram isto um problema e por muitas vezes um acesso errôneo a memória pode passar despercebido no i386 ou ARM (18).

Propõem-se que fruto deste trabalho seja distribuído livremente sob a licença ISC, permitindo portanto que toda a comunidade do OpenBSD possa se beneficiar do mesmo.

[2] Um programa que não possui um terminal controlador e executa em *background*

Ao término da implementação deverá ser feito um pedido para inclusão desta na base do sistema operacional OpenBSD.

## 3.2 Status

O OpenMDNS, proposto neste trabalho, encontra-se em fase de desenvolvimento, que teve início em Fevereiro de 2010 tendo em vista ser abordado no Trabalho de Conclusão.

Atualmente já é possível realizar algumas das operações propostas neste trabalho, tais como resolver *hostnames*, enumerar serviços e resolvê-los. Foram feitos testes nas arquiteturas: alpha, i386, amd64 e sparc64. Foi utilizado uma máquina com o sistema operacional Linux e a implementação MDNS/DNS-SD Avahi para testes de interoperabilidade.

Ainda não é possível publicar serviços através da biblioteca *libmdns*, sendo esta a principal funcionalidade ausente no momento.

Em Maio de 2010 o OpenMDNS foi apresentado à dois integrantes da equipe do OpenBSD: Marco Peereboom e Nicholas Marriot, e estes fizeram diversos comentários, críticas e realizaram revisões de código, apontando *bugs* e imperfeições. Em Junho de 2010 o projeto foi levado à discussão entre a equipe do OpenBSD e outros membros demonstraram interesse. Sendo assim crê-se que o trabalho possui chances de ser aceito no futuro na base do OpenBSD. A inclusão do projeto representaria um marco importante pois rapidamente milhares de pessoas estariam indiretamente utilizando o projeto, reportando suas experiências e *bugs* encontrados.

Portanto o objetivo deste trabalho é finalizar a implementação do OpenMDNS. Para isto será implementado a funcionalidade de publicação de serviços e outras secundárias, além dos itens descritos nas seções do *draft* MDNS(1) e *draft* DNS-SD(2) que ainda não foram implementados.

A seguir são apresentados alguns itens importantes para o desenvolvimento do trabalho:

- Suporte a múltiplas interfaces de rede.
- Suporte a publicação de serviços.
- Biblioteca *libmdns* em forma de *shared library*.

- Melhorar a gramática do programa *mdnsctl*.
- Melhorar o tratamento de erros na biblioteca *libmdns*.
- Fornecer um meio síncrono de utilização da biblioteca, abstraindo o conceito de eventos, atualmente esta é toda orientada a eventos assíncronos.
- Implementar suporte a *cache dump*, isto é, de alguma forma conseguir realizar a visualização de todas as entradas no cache do *mdnsd*.

Além destes, na tabela 1 é apresentado o estado da implementação de cada seção descrita no *draft* MDNS(1) e na tabela 2 é apresentado o mesmo para o *draft* DNS-SD (2). O estado da implementação de cada seção é descrito como *I = "Implementado"*, *NI = "Não Implementado"*, *PI = "Parcialmente Implementado"* e *NS = "Não será implementado"*.

Neste trabalho serão implementados as seções descritas como *NI* e *PI*.y

## 3.3 Atividades Previstas

A seguir é apresentado a enumeração das atividades propostas assim como sua data de execução:

1. *(14/09/10 - 18/09/10)*: Estudo detalhado das implementações existentes Avahi e Bonjour, comparar estruturas de dados, arquitetura e interface da biblioteca.
2. *(18/09/10 - 24/09/10)*: Definição dos atributos a serem comparados das implementações existentes ao término do trabalho.
3. *(24/09/10 - 26/09/10)*: Re-leitura dos *drafts* do MDNS e DNS-SD.
4. *(26/09/10 - 27/09/10)*: Definição de quais funcionalidades adicionais serão implementadas.
5. *(27/09/10 - 29/09/10)*: Revisão bibliográfica.
6. *(30/09/10 - 02/10/10)*: Definição do ambiente de testes.
7. *(02/10/10 - 03/10/10)*: Definição dos revisores do código.
8. *(03/10/10 - 22/11/10)*: Elaboração do volume final.

Tabela 1: Estado da Implementação das Seções do Draft MDNS(1)

| Seção | Item | Status |
|---|---|---|
| 4 | Reverse Address Mapping | I |
| 5.1 | One-Shot Multicast DNS Queries | NS |
| 5.2 | One-Shot Queries, Accumulating Multiple Responses | I |
| 5.3 | Continuous Multicast DNS Querying | I |
| 5.4 | Multiple Questions per Query | I |
| 5.5 | Questions Requesting Unicast Responses | NI |
| 5.6 | Direct Unicast Queries to port 5353 | NI |
| 6.1 | Known Answer Suppression | I |
| 6.2 | Multi-Packet Known Answer Suppression | I |
| 6.3 | Duplicate Question Suppression | NI |
| 7 | Responding | PI |
| 7.1 | Negative Responses | NI |
| 7.2 | Responding to Address Queries | PI |
| 7.3 | Responding to Multi-Question Queries | I |
| 7.4 | Response Aggregation | I |
| 7.5 | Wildcard Queries (qtype "ANY" and qclass "ANY") | NI |
| 7.6 | Legacy Unicast Responses | NI |
| 8 | Probing and Announcing on Startup | I |
| 8.1 | Probing | I |
| 8.2 | Simultaneous Probe Tie-Breaking | NI |
| 8.2.1 | Simultaneous Probe Tie-Breaking for Multiple Records | NI |
| 8.3 | Announcing | I |
| 8.4 | Updating | NI |
| 9 | Conflict Resolution | NI |
| 10.2 | Goodbye Packets | NI |
| 10.3 | Announcements to Flush Outdated Cache Entries | NI |
| 10.4 | Cache Flush on Topology change | NI |
| 10.5 | Cache Flush on Failure Indication | NI |
| 10.6 | Passive Observation of Failures (POOF) | NI |
| 11 | Source Address Check | I |

Tabela 2: Estado da Implementação das Seções do Draft DNS-SD(2)

| Seção | Item | Status |
|---|---|---|
| 4 | Service Instance Enumeration (Browsing) | I |
| 5 | Service Name Resolution | I |
| 8 | Flagship Naming | NI |
| 9 | Service Type Enumeration | I |
| 12 | Discovery of Browsing and Registration Domains | NS |

# *Referências*

1 CHESHIRE, M. K. S. *Multicast DNS*. [S.l.], set. 2010. At http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt.

2 CHESHIRE, M. K. S. *DNS Service Discovery*. [S.l.], maio 2010. At http://files.dns-sd.org/draft-cheshire-dnsext-dns-sd.txt.

3 MOCKAPETRIS, P. V. *RFC 1035: Domain names — implementation and specification*. nov. 1987. Disponível em: <ftp://ftp.internic.net/rfc/rfc1035.txt>.

4 OPENBSD, Operating System. http://www.openbsd.org.

5 OPENBSD Policy. http://www.openbsd.org/policy.html.

6 ISC License. http://opensource.org/licenses/isc-license.txt.

7 CHESHIRE, M. K. S. *Requirements for a Protocol to Replace AppleTalk NBP*. [S.l.], mar. 2010. At http://files.multicastdns.org/draft-cheshire-dnsext-nbp-08.txt.

8 BONJOUR. http://developer.apple.com/networking/bonjour/download/.

9 AVAHI. http://www.avahi.org.

10 NETBSD, Operating System. http://www.netbsd.org.

11 APACHE-2 License. http://www.apache.org/licenses/LICENSE-2.0.html.

12 THEO De Raadt Interview. http://www.thejemreport.com/content/view/369/.

13 RAYMOND, E. S. *The Art of UNIX Programming*. pub-AW:adr: Addison-Wesley, 2004. xxxii + 525 p. ISBN 0-13-124085-4.

14 Brooks, Jr., F. P. *The Mythical Man Month: Essays on Software Engineering*. first. Addison-Wesley, 1975. 200 p. Disponível em: <http://www.amazon.com/Mythical-Man-Month-Essays-Software-Engineering/dp/0201006502>.

15 OPENBGP. http://www.openbgp.org.

16 OPENOSPF. http://www.openbsd.org.

17 OPENSSH. http://www.openssh.com.

18 HUSEMANN, M. Fighting the lemmings. ????

19 MOCKAPETRIS, P. *Domain Names - Concepts and Facilities*. [S.l.], nov. 1987. At http://info.internet.isi.edu/in-notes/rfc/files/rfc1034.txt.

20 CHESHIRE, M. K. S. *Requirements for a Protocol to Replace AppleTalk NBP*. [S.l.], set. 2010. At http://files.dns-sd.org/draft-cheshire-dnsext-nbp.txt.

21 BSD License. http://www.opensource.org/licenses/bsd-license.php.

22 STEVENS, R. W. *UNIX Network Programming*. [S.l.]: Prentice Hall PTR, 1990. (Software Series).

23 STEVENS, R. W. *TCP/IP Illustrated, Volume 1: The Protocols*. Reading: Addison Wesley, 1994. ISBN 0-201-63346-9.

24 STEVENS, W. R. *TCP/IP Illustrated, Volume 2*. [S.l.]: Addison Wesley, 1994.

25 STEVENS, W. R. *Advanced Programming in the UNIX Environment*. [S.l.]: Addison-Wesley, 1992.