

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Estudo Comparativo entre Protocolos de Transporte

TCP, SCTP e XTP

Solange Sonaglio

Florianópolis – SC

2009/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Estudo Comparativo entre Protocolos de Transporte
TCP, SCTP e XTP

Solange Sonaglio

Trabalho de conclusão de curso submetido
à Universidade Federal de Santa Catarina
como parte dos requisitos para obtenção
do grau de Bacharel em Sistemas de
Informação.

Florianópolis – SC

2009/1

Solange Sonaglio

**Estudo Comparativo entre Protocolos de Transporte
TCP, SCTP e XTP**

Trabalho de conclusão de curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador

Prof. Mário Antônio Ribeiro Dantas, Dr.
Universidade Federal de Santa Catarina
mario@inf.ufsc.br

Banca examinadora

Prof. Roberto Willrich, Dr.
Universidade Federal de Santa Catarina
willrich@inf.ufsc.br

Prof. Vítório Bruno Mazzola, Dr.
Universidade Federal de Santa Catarina
mazzola@inf.ufsc.br

AGRADECIMENTOS

À Deus.

Aos meus pais pelo amor incondicional e apoio em todos os momentos.

Ao meu esposo pela paciência e conselhos.

Às minhas irmãs pela ajuda.

Ao meu orientador pela motivação e apoio.

Ao professor Eraldo da Silveira (CEFET/SC) pela disposição em me auxiliar sempre que precisei.

E a todos os que colaboraram com o desenvolvimento deste trabalho.

RESUMO

Em uma era onde as redes ganham cada dia mais vazão e capacidade de transmissão, onde os equipamentos de ponta e de interconexão melhoram em termos de hardware e software a cada dia, os protocolos de transporte também precisam se adaptar e ganhar desempenho.

O protocolo mais utilizado na Internet de hoje, o TCP, foi desenvolvido em uma era onde as redes eram de baixíssima velocidade e desempenho, por isso ele é um protocolo robusto e, para as redes de hoje, “pesado”, apresentando muitas limitações, principalmente na parte de segurança.

Assim, para suprir algumas necessidades específicas, outros protocolos de transporte foram surgindo para corrigir as deficiências do TCP e melhorar o desempenho das aplicações e da rede. Entre estes protocolos estão o SCTP (Stream Control Protocol) e o XTP (Xpress Transport Protocol), que são protocolos projetados para redes de alto desempenho, com baixa taxa de erro e alta largura de banda.

Este estudo tem como escopo a apresentação das principais características dos três protocolos citados, assim como seus mecanismos, fazer um comparativo dessas características e apresentar o resultado de testes empíricos realizados com os mesmos. Desta forma, espera-se identificar a melhor adequação dos protocolos de transporte para uma classe de aplicação.

Palavras-chave: Protocolo de transporte, comparativo, TCP, SCTP, XTP.

ABSTRACT

In an era where the networks gain each day more throughput and capacity of transmission, where the equipments of end and of interconnection improve in terms of hardware and software to each day, the transport protocols also need adapted themselves and to gain performance.

The protocol most used in Internet the today, the TCP, was developed in an era where the networks were of very low speed and performance, therefore it is a robust protocol and, for the today networks, "heavy", presenting many limitations, principally in the part of security.

So, to provide some specific necessities, other transport protocols were appearing to correct the deficiencies of the TCP and to improve the performance of the applications and of the networks. Between these protocols they are the SCTP (Stream Control Protocol) and the XTP (Xpress Transport Protocol), which are protocols projected for networks of high performance, with low rate of error and high width of band.

This study takes as an aim the presentation of the principal characteristics of three quoted protocols, as well as his mechanisms, to do a comparative one of these characteristics and to present the result of empirical tests realized with same. In this way, it is waited to identify the best adaptation of the transport protocols for a class of application.

Key-words: Transport protocol, comparative, TCP, SCTP and XTP.

SUMÁRIO

LISTA DE FIGURAS	9
LISTA DE TABELAS.....	10
1. Introdução	14
1.1. Tema.....	14
1.2. Escopo	14
1.3. Objetivo Geral.....	15
1.4. Objetivos Específicos	15
1.5. Motivação.....	15
2. Protocolos de Transporte	17
2.1. Mecanismos dos Protocolos de Transporte.....	20
2.1.1. Mecanismos de mapeamento de unidades de dados	20
2.1.2. Mecanismos de Delimitação de Mensagem.....	21
2.1.3. Mecanismos de Multiplexação.....	21
2.1.4. Mecanismos de Gerenciamento de Caminho	21
2.1.5. Mecanismo de Fluxo de Dados.....	22
2.1.6. Mecanismos de Controle de Erro	23
2.1.7. Mecanismos de Gerenciamento de Conexão	26
2.1.8. Mecanismos de Segurança.....	27
2.1.9. Mecanismos de Multicast.....	27
2.1.10. Mecanismos de Prioridade de Processamento	28
2.1.11. Mecanismos de Dados Fora da Banda.....	28
2.2. Transmission Control Protocol	29
2.2.1. Formato de Dados	29
2.2.2. Mecanismos de Fluxo de Dados.....	30
2.2.4. Mecanismos de Gerenciamento de Conexão	33
2.2.5. Mecanismos de Segurança.....	34
2.2.6. Características Especiais.....	41
2.3. Stream Control Transport Protocol.....	43
2.3.1. Formato de Dados	44
2.3.2. Mecanismos de Fluxo de Dados.....	45
2.3.3. Mecanismos de Controle de Erro	47
2.3.4. Mecanismos de Gerenciamento de Conexão	48

2.3.5.	Mecanismos de Segurança	50
2.3.6.	Características Especiais.....	51
2.4.	Xpress Transport Protocol	55
2.4.1.	Formato de Dados	55
2.4.2.	Mecanismos de Fluxo de Dados.....	57
2.4.3.	Mecanismos de Controle de erro	59
2.4.4.	Mecanismos de Gerenciamento de Conexão	61
2.4.5.	Mecanismos de Segurança	63
2.4.6.	Características Especiais.....	63
3.	Estudo Comparativo dos Protocolos de Transporte.....	66
3.1.	Comparativo dos Mecanismos do TCP, SCTP e XTP	66
3.1.1.	Mecanismos de mapeamento de unidades de dados	67
3.1.2.	Mecanismos de Delimitação de Mensagem.....	69
3.1.3.	Mecanismos de Multiplexação.....	70
3.1.4.	Mecanismos de Gerenciamento de Caminho	70
3.1.5.	Mecanismo de Fluxo de Dados.....	72
3.1.6.	Mecanismos de Controle de Erro	74
3.1.7.	Mecanismos de Gerenciamento de Conexão	78
3.1.8.	Mecanismos de Segurança.....	79
3.1.9.	Mecanismo de Multicast.....	79
3.1.10.	Mecanismo de Prioridade de Processamento	79
3.1.11.	Mecanismo de Dados Fora da Banda.....	80
3.2.	Metodologia	81
3.2.1.	Descrição do Ambiente	81
3.2.2.	Descrição dos Testes.....	83
3.2.3.	Resultado dos Testes.....	84
3.2.4.	Conclusão dos Testes.....	93
4.	Conclusão e Trabalhos Futuros	94
4.1.	Trabalhos Futuros.....	95
	Referências Bibliográficas	96
	Apêndice	101

LISTA DE FIGURAS

Figura 2.1 - Mapeamento do TCP/IP para o Modelo OSI (RAMOS, 2006, p. 2)	17
Figura 2.2 - Formato do Segmento TCP (FARREL, 2005)	30
Figura 2.3 - Transmissão Confiável TCP (VIEGAS, 2008)	33
Figura 2.4 - Three-way Handshake TCP (SCTP Programmer's Guide, 2007)	34
Figura 2.5 - SYN Flooding (CHAMBERS et al, 200x)	35
Figura 2.6 - IP Spoofing via Sequence Guessing (CHAMBERS et al, 200x)	36
Figura 2.7 - Connection Hijacking	37
Figura 2.8 - Máquina de estados do TCP (STEVENS, 1994)	39
Figura 2.9 - Abertura Simultânea de Conexão (STEVENS, 1994).....	41
Figura 2.10 - Localização lógica do SCTP na pilha TCP/IP (COSTA, 2005)	43
Figura 2.11 - Formato dos pacotes SCTP (SANTOS, RIBEIRO e MINICZ, 2008).....	44
Figura 2.12 - Formato das Mensagens SCTP (COSTA, 2005, p. 36).....	45
Figura 2.13 - Associação SCTP (SANTOS, RIBEIRO e MINICZ, 2008).....	49
Figura 2.14 - Fluxos de uma associação SCTP (SANTOS et al, 2008, p.74)	52
Figura 2.15 - Pacote XTP (STRAYER, 1995, p. 14).....	56
Figura 2.16 - Formato dos pacotes XTP (SANDER e WEAVER, 2007)	56
Figura 2.17 - Estabelecimento de Associação XTP (STRAYER, 1995, p. 3).....	61
Figura 2.18 - Fechamento Gracioso Completo (STRAYER, 1995, p 60)	62
Figura 3.19 - Vazão em Loopback	85
Figura 3.20 - Latência em Loopback.....	86
Figura 3.21 - Latência Loopback Multi 278MB	86
Figura 3.22 - Latência Loopback Multi 1456MB	87
Figura 3.23 - Latência Loopback Multi 23256MB	88
Figura 3.24 - Vazão em Rede 100 Mbps.....	89
Figura 3.25 - Latência em Rede 100 Mbps	90
Figura 3.26 - Latência em Rede 100 Mbps Multi 278MB.....	91
Figura 3.27 - Latência em Rede 100 Mbps Multi 1456MB.....	91
Figura 3.28 - Latência em Rede 100 Mbps Multi 23256MB.....	92

LISTA DE TABELAS

Tabela 3.1 - Comparativo Geral.....	66
Tabela 3.2 - Mecanismo de Fragmentação.....	67
Tabela 3.3 - Mecanismo de Segmentação.....	68
Tabela 3.4 - Mecanismo de Blocking.....	68
Tabela 3.5 - Mecanismo de Delimitação de Mensagem.....	69
Tabela 3.6 - Mecanismo de Multiplexação de Aplicação.....	70
Tabela 3.7 - Mecanismo de Multiplexação de Fluxo.....	70
Tabela 3.8 - Mecanismo de Heartbeat/Keep-alive.....	71
Tabela 3.9 – Mecanismo de Is-alive.....	71
Tabela 3.10 - Mecanismo de Troca de Caminho.....	71
Tabela 3.11 - Mecanismo de Controle de Fluxo.....	72
Tabela 3.12 - Mecanismo de Controle de Congestionamento.....	73
Tabela 3.13 - Mecanismo de Controle de Taxa.....	73
Tabela 3.14 - Mecanismo de Checksum.....	74
Tabela 3.15 - Mecanismo de Detecção de PDU Perdida.....	75
Tabela 3.16 - Mecanismos de Retransmissão.....	75
Tabela 3.17 - Mecanismo de Detecção de Duplicações.....	76
Tabela 3.18 - Mecanismo de Reordenação de Datagramas.....	77
Tabela 3.19 - Mecanismo de Reordenação de Mensagens.....	77
Tabela 3.20 - Mecanismo de Estabelecimento de Conexão.....	78
Tabela 3.21 - Mecanismo de Liberação de Conexão.....	78
Tabela 3.22 - Mecanismos de Multicast.....	79
Tabela 3.23 - Mecanismo de Priorização de Processamento.....	80
Tabela 3.24 - Mecanismo de Dados Fora da banda.....	80
Tabela 3.25 - Vazão em Loopback.....	85
Tabela 3.26 - Latência em Loopback.....	86
Tabela 3.27 - Latência Loopback Multi 278MB.....	87
Tabela 3.28 - Latência Loopback Multi 1456MB.....	88
Tabela 3.29 - Latência Loopback Multi 23256MB.....	88
Tabela 3.30 - Vazão em Rede 100 Mbps.....	89
Tabela 3.31 - Latência em Rede 100 Mbps.....	90
Tabela 3.32 - Latência em rede 100 Mbps Multi 278MB.....	91
Tabela 3.33 - Latência em Rede 100 Mbps Multi 1456MB.....	92
Tabela 3.34 - Latência em Rede 100 Mbps Multi 23256MB.....	92

ACRÔNIMOS

API: Application Programming Interface. Conjunto de funções que um aplicativo deve usar para comunicar-se com uma biblioteca ou com o sistema operacional.

ARPA: Advanced Research Projects Agency.

BS: Blind Spoof -Tipo de ataque contra o protocolo de transporte TCP.

BSD Unix: Sistema operacional compatível com UNIX.

BSD/Sockets: API do BSD Unix para acesso aos recursos de rede.

CRC: Cyclic Redundancy Check. Algoritmo de somatório de verificação baseado em divisão de polinômios.

CRC-32: versão de CRC que gera somatório de 32 bits.

CRC-32c: versão do CRC que gera somatório de 32 bits, com polinômio divisor diferente do CRC-32.

DoS: Denial-of-Service. Ataque de negação de serviço, que impede os usuários legítimos de usar determinado serviço.

FTP: File Transfer Protocol. Protocolo para transferência de arquivos.

IANA: Internet Assigned Numbers Authority.

IETF: Internet Engineering Task Force.

IPSEC: Internet Protocol SEcurity.

IP: Internet Protocol. Protocolo de rede da pilha TCP/IP.

ISDN: Integrated Services Digital Network.

ISN: Initial Sequence Number. TSN inicial de uma conexão TCP ou associação SCTP.

ITU-T: International Telecommunication Union – Telecommunication.

Kernel: Núcleo do sistema operacional.

Kernel level: Código que roda dentro do kernel.

LAN: Local Area Network.

LK-SCTP: Linux Kernel SCTP.

Linux XTP: Módulo de Kernel para XTP.

MAC: Midia Access Controle.

MTU: Maximum Transmission Unit.

Octeto: Byte com invariavelmente 8 bits.

OSI: Open Systems Interconnection. Pilha de protocolos de rede criada pela ITU-T.

PEI: Protocol Engines Incorporated.

PR-SCTP: Partial Reliability SCTP. Extensão do SCTP que permite relaxar a confiabilidade de mensagens.

RFC: Request For Comment.

RTO: Retransmission TimeOut. Tempo que o transmissor espera para receber uma confirmação do receptor.

RTT: Round-Trip Time. Tempo decorrido entre a transmissão do pacote e o recebimento da respectiva confirmação.

SCTP: Stream Control Transmission Protocol.

SIGTRAN: Comitê Signaling Transport.

Soquete: em BSD/Sockets, um manipulador de arquivo que corresponde a uma conexão de rede.

Soquete UNIX: método de comunicação interprocessos acessível através da API BSD/Sockets.

Spoof: nome genérico dado a ataques que envolvem falseamento da origem, que mascara o invasor.

SS7: Signaling System #7. Pilha de rede de comutação de pacotes usada em telefonia.

SSL: Secure Sockets Layer.

SSN: Stream Sequence Number.

SYN Flood: Ataque de negação de serviço contra o protocolo TCP.

TCP: Transmission Control Protocol.

TCP/IP: Transmission Control Protocol/Internet Protocol.

TCPM: no contexto deste trabalho é um protocolo de aplicação criado para testes de desempenho.

TLS: vide SSL.

TSN: Transmission Sequence Number.

UDP: User Datagram Protocol.

UNIX: Família de sistemas operacionais.

UNIX socket: vide soquete UNIX.

XTP: Xpress Transport Protocol.

WAN: Wide Area Network.

1. Introdução

O tráfego de serviços IP tem crescido de uma forma exponencial nos últimos tempos, aumentando a necessidade de melhorar e otimizar a transmissão dos dados nas redes IP. Neste cenário, os protocolos de transporte em uso hoje, como o TCP, o mais difundido e utilizado protocolo de transporte, apresentam limitações quanto ao desempenho e segurança.

Outros protocolos, voltados para prover às redes um alto desempenho, têm sido propostos. Exemplos clássicos são os protocolos SCTP (*Stream Control Transmission Protocol*) e o XTP (*Xpress Transport Protocol*).

Protocolos como o SCTP e o XTP têm o perfil de interesse no desenvolvimento deste trabalho, que visa identificar se os protocolos citados têm características de desempenho que os qualifica a serem utilizados como uma alternativa em relação ao TCP.

1.1. Tema

Estudo comparativo entre os protocolos de transporte leves SCTP e XTP com o protocolo TCP, a fim de identificar a adequação e o desempenho apresentado pelos protocolos SCTP e XTP em termos de equivalência, de desempenho em relação ao consolidado protocolo TCP.

1.2. Escopo

O escopo deste projeto delimita-se por um estudo comparativo entre o protocolo de transporte TCP e os protocolos SCTP e XTP. Pesquisa bibliográfica comparativa entre os três protocolos e um estudo empírico comparativo de desempenho entre os protocolos testados em aplicações reais.

1.3. Objetivo Geral

O objetivo geral é comparar, em termos de desempenho, o tradicional protocolo TCP, com os protocolos de alto desempenho SCTP e XTP, de forma a obter informações úteis para a divulgação e utilização destes protocolos em aplicações que requeiram funcionalidades particulares de transporte além das providas pelo TCP.

1.4. Objetivos Específicos

- Caracterização de propriedades intrínsecas dos protocolos TCP, SCTP e XTP;
- Comparação das funcionalidades em nível de transporte;
- Contrastar o desempenho dos protocolos SCTP e XTP, em relação ao TCP;
- Analisar os resultados obtidos nos testes empíricos de desempenho.

1.5. Motivação

É fundamental nas redes de comunicação e computadores, que os protocolos de transporte utilizados ofereçam segurança, sejam confiáveis e que possam garantir a integridade dos dados de uma forma diferenciada.

A maioria dos protocolos utilizados atualmente na pilha de protocolos TCP/IP em nível de transporte se apresentam obsoletos para uma grande classe de aplicações. O protocolo UDP é utilizado para aplicações que não requerem controle de erros ou fluxo, nem transmissão confiável, ou que para funcionar em tempo real, precisam abrir mão destas características, como a transferência de voz, áudio e vídeo.

O protocolo de transporte ideal deve apresentar a confiabilidade do TCP, com a rapidez do UDP, além de outras características importantes para

aplicações da atualidade, como segurança, facilidade de multicast e uma grande quantidade de fluxo simultânea.

A motivação deste trabalho deriva justamente deste fato de poder utilizar um protocolo eficiente, como o UDP e confiável como o TCP, porém sem os inconvenientes inerentes aos dois.

No desenvolvimento, serão comparados com o tradicional TCP, protocolos que oferecem segurança, confiança e integridade dos dados, de forma mais leve e com melhor desempenho.

Espera-se ao final do trabalho ter dados concretos sobre o desempenho destes protocolos, de forma que se demonstre que é possível utilizar um protocolo tão robusto quanto o TCP, porém bem menos complexo.

2. Protocolos de Transporte

Um protocolo é uma convenção ou padrão que controla e possibilita uma conexão, comunicação ou transferência de dados entre dois sistemas computacionais. De maneira simples, um protocolo pode ser definido como as regras que regem a sintaxe, semântica e sincronização da comunicação. Os protocolos podem ser implementados por hardware, software ou por uma combinação dos dois. Um protocolo é para os computadores o que uma linguagem é para os humanos.

Os protocolos de transporte são o objeto de estudo deste trabalho e, segundo Tanenbaum (2003), o núcleo de toda a hierarquia de protocolos. Sua função é promover uma transferência de dados confiável e econômica entre a máquina de origem e a máquina de destino, independente das redes físicas em uso no momento.

O protocolo de transporte é o quarto protocolo do modelo de referência OSI. O serviço de transporte é oferecido à camada de aplicação (quinta camada do OSI) e a camada de transporte utiliza vários serviços oferecidos pela camada de rede (terceira camada do OSI). A camada de Transporte separa as camadas de nível de aplicação (camadas 5 a 7) das camadas de nível físico (camadas de 1 a 3). O mapeamento entre o modelo OSI e o TCP/IP, que é o modelo utilizado por todos os protocolos de interesse neste estudo, é mostrado na **Figura 2.1**.

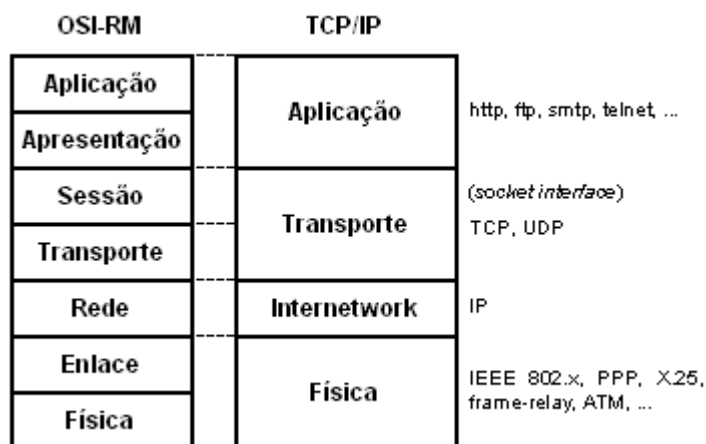


Figura 2.1 - Mapeamento do TCP/IP para o Modelo OSI (RAMOS, 2006, p. 2)

Dentre os protocolos de transporte existentes, destacam-se pela maior utilização e divulgação, o TCP (Transmission Control Protocol – Protocolo de Controle de Transmissão) e o UDP (User Datagram Protocol – Protocolo de Datagrama de Usuário).

O TCP é o protocolo de transporte mais aceito e utilizado nos dias de hoje. A versatilidade e robustez deste protocolo o tornaram adequado para redes globais, já que este verifica se os dados são enviados de forma correta, na sequência apropriada e sem erros, pela rede. O TCP é um protocolo orientado à conexão e, portanto, inclui vários mecanismos para iniciar, manter e encerrar a comunicação, negociar tamanhos de pacotes, detectar e corrigir erros, evitar congestionamento do fluxo e permitir a retransmissão de pacotes corrompidos, independente da qualidade do meio físico. As principais características deste protocolo serão detalhadas mais adiante.

O UDP não é objeto de estudo deste trabalho, porém não se pode deixar de mencionar algumas de suas características e sua importância no universo de protocolos de transporte. O UDP dá às aplicações acesso direto ao serviço de entrega de datagramas, como o serviço de entrega do protocolo IP. Ele é feito para transmitir dados pouco sensíveis, como fluxos de áudio e vídeo, ou para comunicação sem conexão. No UDP não existem checagens e nem confirmação. A ausência de estruturas de controle complexas garante ao UDP alta eficiência, já que cada pacote é composto praticamente somente por dados.

Tanto o TCP quanto o UDP possuem deficiências e algumas aplicações não se adaptam a nenhum dos dois. Desta forma, surgiram outros protocolos de transporte, para suprir estas necessidades. O principal objetivo dos novos protocolos é unir as facilidades do TCP com a eficiência do UDP. Desta forma, solucionando os problemas de segurança do TCP e a falta de confiabilidade do UDP.

Neste cenário, surgiu o protocolo de transporte SCTP (*Stream Control Transport Protocol* – Protocolo de Transporte de Controle de Fluxo). O SCTP foi inicialmente criado para o transporte de sinalização SS7, visto que o TCP se mostrou inviável para tal fim. Com o desenvolvimento, percebeu-se o potencial deste novo protocolo. A principal vantagem do SCTP é que ele fornece um número de funções robustas e flexíveis a diversas aplicações, funções estas

que o TCP e o UDP não oferecem. Além disso, ele possibilita benefícios às aplicações que necessitem de desempenho adicional. Mais adiante serão detalhadas as características deste protocolo.

O último protocolo de interesse neste estudo é o XTP (Xpress Transport Protocol – Protocolo de Transporte Expresso). O XTP faz parte dos protocolos de transporte da nova geração, projetados especificamente para suportar redes de alta velocidade e aplicações multimídia. Ele satisfaz as exigências de muitos sistemas de tempo real. O XTP foi projetado para acomodar a realidade dos sistemas modernos: alta vazão e baixas taxas de erro. É um protocolo programável no sentido em que o transmissor pode selecionar as opções da comunicação, ou seja, o protocolo oferece mecanismos e o usuário seleciona políticas, como será explicado adiante.

Sendo assim, este capítulo inicialmente descreverá os principais mecanismos que um protocolo de transporte possui. Estes mecanismos serão utilizados no Capítulo 3 para fazer o comparativo entre o TCP, SCTP e XTP. Após esta descrição serão apresentados os protocolos TCP, SCTP e XTP e, suas principais características.

2.1. Mecanismos dos Protocolos de Transporte

Alguns mecanismos implementados por protocolos de transporte, citados por Henrici (2002), são apresentados a seguir. Dentre eles, mapeamento de unidade de dados, gerenciamento de caminho, fluxo de dados, controle de erro, gerenciamento de conexão e mecanismos de segurança.

2.1.1. Mecanismos de mapeamento de unidades de dados

O protocolo de transporte deve oferecer um mecanismo para subdividir unidades de dados maiores em unidades menores, se o tamanho da unidade de dados do protocolo de transporte (TPDU) exceder o limite dado pela camada de enlace. Ele também deve prover um mecanismo oposto a este, que combina pequenas quantidades de dados em uma unidade maior, reduzindo assim o número destas unidades.

Os mecanismos de fragmentação, segmentação e repacotamento significam dividir uma quantidade de dados grande em pedaços pequenos. Se os dados são divididos usando fragmentação os fragmentos não são independentes entre si e possuem o mesmo checksum e número de seqüência. Se os dados são divididos usando segmentação os segmentos são independentes uns dos outros, cada um tem seu checksum e seu número de seqüência único. O repacotamento é o mesmo que segmentação, mas normalmente utilizado quando o TCP, por exemplo, faz retransmissão e os dados são segmentados novamente, não necessariamente da mesma forma que foram segmentados na primeira transmissão.

Já os mecanismos de blocking e concatenação trabalham de maneira oposta, combinando dados. A diferença entre blocking e concatenação é que o blocking combina várias SDUs (unidades de dados de serviço) para uma simples PDU, estas SDUs têm um cabeçalho e um trailer comum. Já a concatenação combina várias PDUs completas, cada uma com seu cabeçalho e seu trailer, para uma simples SDU da camada inferior.

O mecanismo de blocking foi planejado para economizar banda, sendo desfeito pelo deblocking no nó receptor, de forma transparente para a camada superior. A ação reversa da concatenação é a separação, que também faz com que o mecanismo seja transparente para a camada superior.

2.1.2. Mecanismos de Delimitação de Mensagem

Muitos protocolos provêm um meio de manter os limites da mensagem, chamado de "delimitação de mensagem".

2.1.3. Mecanismos de Multiplexação

Em geral, multiplexação é uma técnica que combina dois ou mais canais de informação em um único, de forma que o processo possa ser desfeito (demultiplexação). Dois mecanismos de multiplexação são de interesse neste estudo, a multiplexação de aplicação e a multiplexação de fluxo.

Entre um par de hosts mais que uma aplicação pode existir desejando trocar informação. Por isso, um protocolo da camada quatro deve ser capaz de distinguir entre vários receptores e emissores em um host único. Isso é a multiplexação de aplicação.

Alguns protocolos da quarta camada são capazes de manter múltiplos fluxos de dados independentes dentro de uma única conexão. Assim, uma vez que uma conexão é estabelecida ela pode ser usada simultaneamente para várias trocas de dados independentes. Isso é a multiplexação de fluxo.

2.1.4. Mecanismos de Gerenciamento de Caminho

Os mecanismos de gerenciamento de caminho cobrem as técnicas planejadas para monitorar e manter conexões de dados entre emissor e receptor.

Um mecanismo "heartbeat/keep-alive" periodicamente investiga o fim de uma conexão quando a conexão está ociosa, mesmo quando não há nenhum dado a ser enviado. Um mecanismo de investigação, enviado ao receptor, faz com que ele retorne uma confirmação de que a conexão está ainda ativa.

Esta característica é interessante para fechar conexões que ficam meio abertas por tempo indeterminado, porque este mecanismo fecha essa conexão completamente depois de um período de tempo.

A desvantagem é que o keep-alive consome largura de banda muitas vezes desnecessariamente, porque os nós são sondados regularmente e comparados. Além disso, falhas transitórias de rede poderiam ser transparentes se nenhum dado estiver sendo transferido e o mecanismo keep-alive não esteja em uso, porém, se o mecanismo é usado neste cenário ele poderia fazer com que boas conexões falhassem.

Uma aplicação pode querer investigar se um nó ainda está ativo. Ao contrário do mecanismo de keep-alive que periodicamente investiga através do tempo de inatividade, um "is-alive" é disparado por uma requisição da camada superior.

Além disso, protocolos que provêm um mecanismo heartbeat podem periodicamente checar caminhos diferentes para o host destino, quando disponível. Assim, se um único caminho falha a entidade protocolar pode trocar para outro para prover tolerância à falha.

2.1.5. Mecanismo de Fluxo de Dados

Mecanismos de fluxo de dados limitam o fluxo de dados em determinado volume. O objetivo é prevenir que dispositivos de rede sejam dominados por uma quantidade de dados muito grande e distribuir equitativamente os recursos disponíveis.

O controle de fluxo é um mecanismo de restrição do volume de dados que pode ser enviado para evitar que o emissor oprima o receptor. Normalmente o volume de saída é regulado por um mecanismo de janela fim a fim baseado em uma janela deslizante de números sequenciais. Então, o início

e o fim da janela deve ser especificado em cada pacote de controle. A borda inferior corresponde ao valor dado como confirmação cumulativa. A borda superior é dada em um campo separado relativo ao início, isto é, o tamanho de janela.

Uma vez que nós intermediários são usados por muitos fluxos de dados usando seus recursos, estes nós podem ser oprimidos pela quantidade de tráfego. Além dos recursos limitados dos nós para processamento de pacotes, as conexões têm largura de banda limitada. O mecanismo de diminuição da taxa de envio quando a rede fica congestionada é chamado controle de congestionamento.

Controle de Taxa é uma técnica de limitação da taxa em que um emissor pode transmitir dados, normalmente pela imposição de um tempo de atraso depois de cada pacote e/ou cada rajada de pacotes. O controle de taxa pode limitar o congestionamento em nós intermediários assim como limitar a perda de pacotes nos receptores.

2.1.6. Mecanismos de Controle de Erro

O termo controle de erro abrange os procedimentos para detectar, tratar e possivelmente recuperar dados perdidos. Os erros que podem afetar as trocas de PDUs entre pares de entidades protocolares operando em um serviço com conexão, são os seguintes: corrupção, perda, duplicação e entrega desordenada de PDUs.

Checksums são usados para detectar corrupção de uma PDU. Isto é necessário na camada três ou quatro se a camada inferior não garante a entrega correta dos dados. Pelo menos erros no cabeçalho que afetariam mais negativamente o processamento, devem ser tratados. A corrupção dos dados pode ser tolerada.

Normalmente pacotes que estão danificados são descartados para evitar processamento de informação potencialmente errada. Então, tal corrupção do conteúdo é observada como uma perda na camada superior.

Para detectar a perda de uma PDU contendo dados, um mecanismo de numeração é empregado. O receptor assume que um pacote foi perdido se

somente PDUs com números de seqüência maiores do que o esperado forem recebidas por um período de tempo.

Usando um mecanismo de confirmação e retransmissão, a PDU perdida pode ser reenviada para recuperação daquele erro. Neste caso a camada superior somente percebe a perda como um aumento no atraso. Se nenhuma retransmissão é empregada a mensagem completa da PDU é perdida. A camada superior no receptor pode ser informada sobre esta circunstância se suportado pelo protocolo.

Se uma comunicação confiável é requerida o receptor deve informar ao emissor os pacotes danificados ou pedidos para que o emissor retransmita estes dados. Existem vários métodos para fazer isso.

Um método comum e simples de confirmar o recebimento de dados é o envio de confirmações cumulativas (CUM ACK), ele confirma todos os pacotes anteriores, dando o número de seqüência do último pacote recebido sem erro.

Além disso, a confirmação seletiva pode ser utilizada para confirmar pacotes únicos ou intervalos de pacotes para evitar retransmissões de pacotes entregues com sucesso.

Confirmação cumulativa e seletiva são "confirmações positivas" indicando que os pacotes foram entregues com sucesso. Por outro lado, "confirmações negativas" podem ser utilizadas para informar o emissor à perda de pacotes. Se esta última deve ser utilizada, o receptor precisa saber quais pacotes esperar.

Para retransmissão existem várias técnicas, como stop-and-wait, go-back-N e retransmissão seletiva.

O mecanismo de stop-and-wait é a técnica mais simples, nele o emissor transmite um pacote único e espera confirmação, se dentro de um período de tempo ele recebe a confirmação, um novo pacote é enviado, senão o pacote é retransmitido. No recebimento de pacotes corrompidos, o receptor deve enviar uma confirmação negativa (NAK) requisitando retransmissão imediata. Este mecanismo somente é adequado para protocolos de comunicação simples uma vez que sempre somente um pacote pode ser não confirmado e muito tempo é gasto esperando por uma confirmação.

O mecanismo de retransmissão go-back-N é usado juntamente com confirmação cumulativa. Um emissor pode injetar vários pacotes no canal antes

de receber uma confirmação. Quando ocorre a perda de algum pacote, todos os pacotes a partir do último confirmado são retransmitidos independentemente se um ou mais pacotes foram afetados (go-back-N). Cada PDU é numerada e precisa ser mantida pelo emissor até que a entrega tenha sido confirmada. Desta forma, o go-back-N comparado ao stop-and-wait aumenta a vazão ao preço de uma maior complexidade e a necessidade de maior espaço em buffer, mas é robusto e ainda relativamente simples.

Utilizar confirmação seletiva ou negativa, além de cumulativa oferece a possibilidade de retransmitir somente os dados que foram danificados ou perdidos, ao invés de todos os dados de um ponto particular em um fluxo de dados, como no go-back-N.

A técnica de retransmissão seletiva minimiza a largura de banda usada para retransmissões ao preço da necessidade de conhecer exatamente o estado do receptor, mas a troca de informações relativas ao estado consome largura de banda também.

Uma PDU pode ficar duplicada, por exemplo, se é retransmitida e ambos, o original e a retransmissão, chegam ao receptor. Utilizando um mecanismo de numeração e mantendo uma lista de todas as PDUs, a duplicação pode ser detectada e descartada. Números de sequência são somente atribuídos a dados. Pacote de controle, por exemplo, para confirmações, não consomem espaço de número de sequência, uma vez que eles contêm informação que não é confirmada e frequentemente é redundante (como confirmação cumulativa).

Se a camada dois não assegura que as PDUs sejam entregues na sequência pode ocorrer na camada três que datagramas sejam entregues para o receptor em uma ordem que não é a ordem de envio. Se a entrega sequenciada é requerida um mecanismo de numeração deve ser empregado tornando o receptor capaz de reordenar os datagramas.

Mensagens podem consistir de múltiplos datagramas. Aqui, o mecanismo de reordenação de datagramas pode ser usado para ordenar os dados em cada mensagem, mas as mensagens algumas vezes não precisam ser entregues a camada superior em ordem. Para outros tipos de aplicações é necessária a entrega das mensagens na mesma ordem que elas tenham sido

enviadas para o nível superior do receptor. Esta ordenação parcial somente é relevante para protocolos orientados a mensagem, como o SCTP e o XTP.

2.1.7. Mecanismos de Gerenciamento de Conexão

No gerenciamento de conexão estão incluídos os procedimentos para estabelecer e liberar conexões. Como um nó muitas vezes precisa saber o estado do nó par para muitos outros mecanismos funcionarem, os mecanismos de gerenciamento de conexão são necessários em todos os protocolos mais complexos.

2.1.7.1. Estabelecimento de Conexão

Para alguns mecanismos, como o controle de erro, trabalharem, informação de estado a respeito do host par tem de ser mantida. Esta informação, como número de seqüência atual tem de ser criada. O procedimento é chamado “criação/ estabelecimento de conexão” e pode ser encontrado em cada protocolo que trabalha pelo menos virtualmente orientado à conexão.

Dois tipos de mecanismos podem ser empregados para realizar isto: criação de conexão baseada em handshake e criação de conexão. Os mecanismos à base de handshake necessitam uma troca explícita de PDUs. Além disso, outras informações, como o uso de características do protocolo, podem ser negociadas.

Os mecanismos implícitos abrem conexões quando o primeiro pacote é recebido já contendo informação. Desta maneira, uma conexão pode ser criada muito mais rapidamente do que com um processo à base de handshake. Isto é uma vantagem significativa para conexões de curta duração, como transações.

Estes mecanismos podem também, serem classificados de vários modos: se para o handshake ou o pacote de iniciação um formato especial de PDU é usado ou não, se a camada superior decidir sobre a aceitação de um pedido de conexão no momento que uma PDU inicial chega ou anteriormente

quando o contexto de escuta é criado, e se uma criação de conexão cruzada, isto é, uma conexão iniciada simultaneamente por ambos os lados, resulta em uma única ou em duas conexões separadas.

Além disso, em vez de se referir a uma conexão sempre com endereços completos, depois do estabelecimento da conexão um identificador especial pode ser empregado. Isto é feito, por exemplo, no XTP onde um campo "key" como identificador é usado.

2.1.7.2. Liberação de Conexão

Uma conexão previamente estabelecida será liberada se ela já não é usada, para liberar os recursos alocados para manter informação de estado. A liberação pode feita explicitamente enviando PDUs indicando que a conexão será encerrada (handshake), o que geralmente é encontrado, ou temporizador dirigido se a conexão não foi utilizada por um determinado período de tempo.

O fechamento gracioso (caso normal) e o fechamento abortivo (por exemplo, no caso de um erro irrecuperável) podem ser diferentes. Em alguns protocolos os dois fluxos de dados unidirecionais de uma conexão bidirecional somente podem ser fechados juntos, em outros o fechamento pode ocorrer independentemente.

2.1.8. Mecanismos de Segurança

Os mecanismos de segurança dos protocolos de transporte são discutidos para cada protocolo analisado, visto que não foram encontradas definições de métodos para inclusão de segurança na camada de transporte.

2.1.9. Mecanismos de Multicast

Um mecanismo multicast 1:n prove um meio para enviar pacotes de um único emissor para um grupo de receptores Outras formas de multicast, como

n:1 ou n:m são geralmente emuladas por várias conexões 1:1 ou 1:n. Para o trabalho de multicast, mecanismos de gerenciamento de grupo (junção e saída de grupos multicast) são necessários.

Todos os protocolos de transporte que não mantêm informação de estado, como o UDP ou que são preparados para multicast, como o XTP, podem se basear em um serviço de entrega multicast oferecido pela camada três (camada de rede).

2.1.10. Mecanismos de Prioridade de Processamento

Especificamente um processamento prioritário destina-se a prover um serviço de priorização de entrega de pacotes selecionados: no lado do transmissor, pode ser dada preferência às entidades que têm dados prioritários para transmitir ao invés daquelas que não. No lado do receptor, operação de priorização implica que, de todos os pacotes recebidos, enfileirados, e aguardando a entrega, o próximo pacote a ser processado será o que tiver a prioridade mais alta.

2.1.11. Mecanismos de Dados Fora da Banda

Alguns protocolos provêm como característica um meio para aplicações misturarem informação de controle com dados enquanto evitam a imposição do frame de outra camada dentro do segmento de dados.

2.2. Transmission Control Protocol

O “Transmission Control Protocol” é atualmente o protocolo mais utilizado na Internet. Este é o protocolo padrão para troca de informações em redes, sendo um protocolo independente e de fácil adaptação para utilização em outros sistemas de comunicação.

Em contrapartida, pode-se dizer que o protocolo TCP é um protocolo complexo. Quando o TCP foi desenvolvido era necessário o oferecimento de confiabilidade, trabalhando-se com vários ambientes de rede, com diferentes ferramentas e com muitos tipos de aplicativos.

O TCP é originalmente definido na RFC 0793, mas há muitas otimizações, extensões e propostas em outras RFCs. Algumas das mais importantes são a RFC 0813 (estratégia de janela e de confirmação), a RFC 0879 (tamanho de segmento máximo), a RFC 0896 e a RFC 2581 (controle de congestionamento) e, a RFC 2988 (temporizador de retransmissão).

Este protocolo é uma camada de transporte confiável orientada por fluxo (quarta camada do OSI) para comunicação unicast que, geralmente, trabalha em cima do protocolo IP.

2.2.1. Formato de Dados

A unidade de transferência do protocolo TCP é chamada de segmento ou mensagem. Os segmentos são trocados para estabelecer conexões, transferir dados, enviar reconhecimentos e fechar conexões.

Conforme citado por Farrel (2005), o TCP utiliza um único segmento para toda a sinalização e transferência de dados e a envia em um datagrama IP. As requisições de controle são sinalizadas nos segmentos pelo uso de bits de flag que indicam quais operações estão em andamento.

Os detalhes de controle e a negociação de parâmetros são gerenciados por opções do TCP no cabeçalho do segmento. Informações de controle e dados podem estar em um mesmo segmento, ou usar segmentos

distintos. O cabeçalho TCP tem um tamanho mínimo de 20 octetos e seu formato é apresentado na **Figura 2.2**.

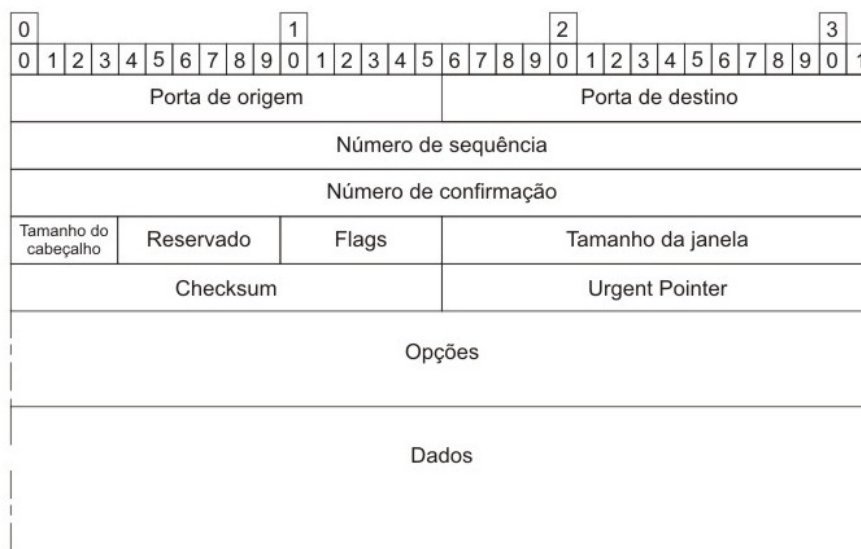


Figura 2.2 - Formato do Segmento TCP (FARREL, 2005)

O segmento é formado pelos campos *porta de origem* e *porta de destino*, cada um com 16 bits. O campo *número de seqüência* e *número de confirmação*, com 32 bits cada. O campo tamanho do cabeçalho, com 4 bits. O campo *Reservado* está reservado para uso futuro, tem tamanho de 6 bits e seu valor deve ser zero.

Os campos *Tamanho da janela*, *Checksum* e *Urgent Pointer* possuem tamanho de 16 bits cada. O campo *Opções* tem tamanho 32 bits e especifica um pequeno conjunto de valores opcionais. O campo *Dados* tem tamanho variável. Para assegurar um mínimo de 32 bits é feito preenchimento com bits zero extras.

Conforme Casad e Willsey (1999), o TCP precisa de todos estes campos de dados para gerenciar, confirmar e verificar com sucesso as transmissões na rede.

2.2.2. Mecanismos de Fluxo de Dados

Os mecanismos de fluxo de dados presentes no TCP são controle de fluxo e controle de congestionamento, explicados a seguir.

2.2.2.1. Controle de Fluxo

O método de controle de fluxo utilizado pelo TCP é chamado de janela deslizante e baseia-se no envio do tamanho da janela de destino determinada pelo lado TCP destino. O protocolo utiliza um processo específico de janela deslizante para fazer o controle de fluxo a fim de garantir uma transmissão eficiente.

Como apresentado por Casad e Willsey (1999), o campo *Tamanho da janela* no cabeçalho do TCP, de dois octetos, tem como finalidade assegurar que o emissor não envie dados em uma velocidade superior à suportada pelo receptor. Este campo informa o valor atual do buffer de recepção do emissor porque o mesmo está presente em todos os segmentos.

A janela deslizante permite que, em vez de apenas um segmento poder ser enviado antes de receber a confirmação, múltiplos segmentos possam ser enviados antes de receber qualquer mensagem de confirmação, garantindo, de acordo com Costa (2005), maior eficiência a comunicação.

2.2.2.2. Controle de Congestionamento

Em uma conexão, sobrecargas na rede podem provocar um estado de congestionamento, desta forma, não deve ocorrer transmissão de segmentos na mesma taxa utilizada em uma rede não congestionada, caso contrário a sobrecarga na rede pode ser ainda maior.

O TCP usa controle de congestionamento fim-a-fim, significando que o remetente limita ou aumenta a taxa de entrega de dados para conexão em função do congestionamento percebido por ele, por isso diz-se que o TCP é auto-regulado.

O algoritmo utilizado pelo TCP, de acordo com Henrici (2002) é chamado de prevenção de congestionamento. A suposição deste algoritmo é que a quantidade de pacotes perdidos por causa de danos é muito pequena (menor que 1%). Assim, a perda de um pacote geralmente sinaliza que ele tenha sido descartado por causa de um congestionamento em algum ponto da

rede, entre a fonte e o destino. Portanto, se houver perda de pacotes o TCP diminui a taxa de envio e, presumindo que a rede está congestionada, o mecanismo de *slow start* (início lento) é usado no início de uma transmissão.

Nas redes atuais com promissoras taxas de vazão isto leva a uma ostensiva subutilização da largura de banda disponível no início. Esta é uma desvantagem significativa, mas ao contrário de técnicas que usam notificações explícitas a maneira implícita de identificar um possível congestionamento no TCP não incrementa tráfego na rede.

2.2.3. Mecanismos de Controle de Erro

Os mecanismos de controle de erro empregados pelo TCP são descritos na sequência.

2.2.3.1. Checksum

O TCP emprega um checksum de complemento de um, de 16 bits, cobrindo tanto o pseudo-cabeçalho IP, quando os dados.

2.2.3.2. Confirmações e Retransmissões

O TCP fornece um serviço de transmissão confiável de dados, onde para cada segmento transmitido o terminal destino envia um segmento com a flag ACK, que sinaliza a confirmação do recebimento deste segmento.

A técnica utilizada pelo protocolo TCP visa oferecer uma confirmação positiva dos segmentos enviados. Para detecção de segmentos perdidos, o TCP fornece dois mecanismos.

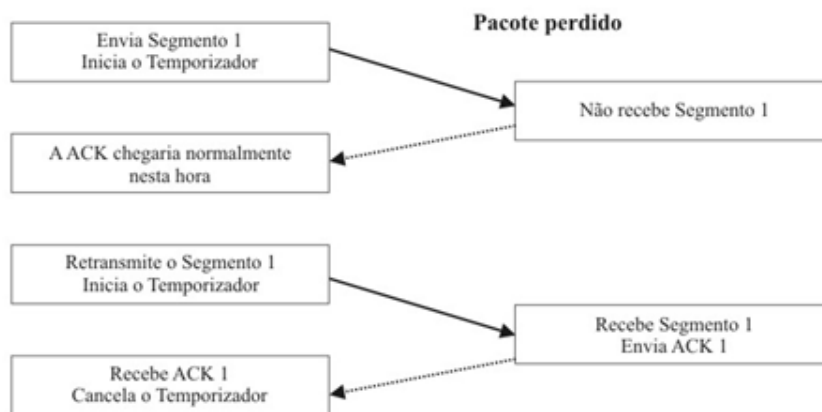


Figura 2.3 - Transmissão Confiável TCP (VIEGAS, 2008)

O primeiro através de um temporizador, definido como RTO (timeout de retransmissão), sendo ele, o tempo que o terminal origem tem para esperar pela chegada de um ACK enviado pelo terminal destino, confirmando o recebimento deste segmento. Caso o RTO expire antes do recebimento do ACK, então o protocolo assume que o segmento foi perdido e inicia a retransmissão do mesmo (COMER, 1998). A **Figura 2.3** apresenta um exemplo do mecanismo de detecção de pacotes perdidos.

O segundo mecanismo é a detecção de perdas através de ACKs duplicados, onde o receptor ao receber os segmentos detecta a falta de um e envia um ACK com o número de sequência do segmento faltante, ao receber outros segmentos envia novamente o ACK com o número daquele segmento faltante. Assim, o emissor vai receber vários ACKs com o mesmo número de sequência, ele então entende que o segmento foi perdido ou danificado e faz a retransmissão.

2.2.4. Mecanismos de Gerenciamento de Conexão

Uma vez que a informação de estado é mantida, o TCP é categorizado como orientado à conexão. De acordo com Farrel (2005), antes dos dados serem transferidos, uma conexão precisa ser estabelecida entre um par {endereço IP origem, porta origem} e {endereço IP destino, porta destino}. Isso

garante que tanto o emissor quanto o receptor estão presentes. Depois são negociadas as capacidades para uso da conexão.

O processo normal de estabelecer uma conexão entre dois terminais TCP envolve três passos: um terminal fonte envia uma mensagem SYN; o terminal destino envia a mensagem de confirmação SYN-ACK e então, o terminal fonte confirma com um ACK. Isto é chamado de handshake de três vias, como pode ser observado na **Figura 2.4**.

Segundo Costa (2005), o TCP utiliza um processo de encerramento de conexões por “metades”, para ele o encerramento de uma conexão *full duplex* deve ser feito explicitamente pelos dois terminais envolvidos. Os segmentos que iniciam o encerramento devem ser marcados com o bit FIN e devem ser confirmados. Cada terminal origina seu próprio segmento FIN.

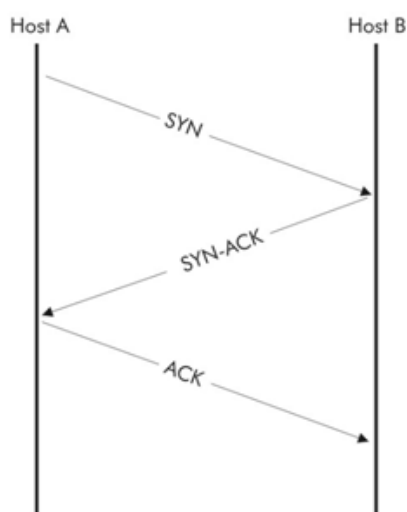


Figura 2.4 - Three-way Handshake TCP (SCTP Programmer's Guide, 2007)

Este processo de encerramento do TCP permite que conexões fiquem em um estado meio aberto, isto é, aberto em um sentido apenas.

2.2.5. Mecanismos de Segurança

Conforme Chambers et al (200x), quando o TCP foi projetado no início dos anos 1980, a segurança não era um assunto prioritário. Contudo, com o passar dos anos a falta da segurança no protocolo TCP tornou-se um

problema. O protocolo TCP, que hoje é a base de Internet, necessita até dos mecanismos mais básicos de segurança, como autenticação ou criptografia. A grande utilização na Internet e a disponibilidade do protocolo TCP expuseram as suas fraquezas. Aqui serão apresentadas algumas vulnerabilidades mais conhecidas.

2.2.5.1. Ataques de SYN

Conforme Guha e Mukherjeea (1997), os ataques de SYN (também conhecidos como SYN Flooding) tiram proveito de uma falha de implementação do handshake de três vias. Quando o host B recebe o pedido de SYN de A, ele deve manter informação da conexão parcialmente aberta em uma "fila de escuta" durante pelo menos 75 segundos, isto deve permitir o sucesso na conexão até com longos atrasos de rede.

Segundo Luckenbach (1996), o problema com a realização deste procedimento é que muitas implementações só podem manter informação de um número muito limitado de conexões (quando muito, de apenas 5 conexões).

Um host malicioso pode explorar o pequeno tamanho da fila de escuta enviando múltiplos pedidos SYN a um host, mas nunca respondendo ao SYN+ACK que o host envia. Desta forma, a fila de escuta do host é rapidamente enchida e ele deixará de aceitar novas conexões, até que uma conexão parcialmente aberta na fila seja concluída. Este tipo de ataque é mostrado na **Figura 2.5**.

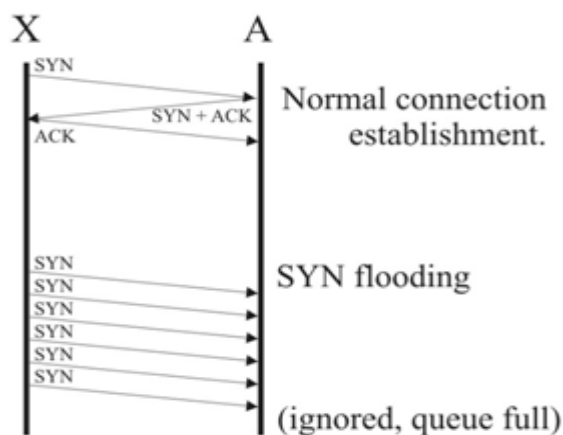


Figura 2.5 - SYN Flooding (CHAMBERS et al, 200x)

2.2.5.2. Sequence Guessing (Suposição de Sequência)

Conforme Chambers et al (200x), o número de sequência usado em conexões TCP é um número de 32 bits, portanto poderia parecer que a probabilidade de adivinhar o ISN correto seria muito baixa. Contudo, se o ISN de uma conexão é atribuído de um modo previsível, fica relativamente fácil adivinhar.

Esta falha em implementações TCP/IP foi reconhecida em 1985, quando Robert Morris descreveu como explorar os ISNs previsíveis no BSD 4.2, um tipo de Unix (MORRIS, 1985). Inicialmente o atacante estabelece uma conexão verdadeira com a vítima, determinando assim o estado atual do contador do sistema, assim fica mais fácil para determinar qual será o próximo ISN.

O atacante tem uma probabilidade realmente mais alta de adivinhar o ISN se ele falsificar frames IP, cada um com um diferente, mas provável ISN e para evitar que sua conexão seja abortada, o atacante pode usar o SYN attack para inundar o host que ele está falsificando. Este tipo de ataque é demonstrado na **Figura 2.6**.

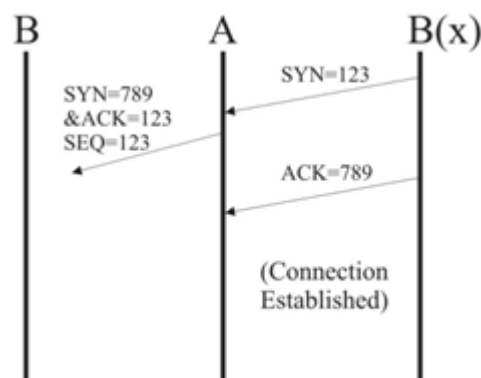


Figura 2.6 - IP Spoofing via Sequence Guessing (CHAMBERS et al, 200x)

Conforme Bellovin (1989), mesmo incrementando o contador de ISN 250.000 vezes por segundo (como sugerido pelo padrão TCP), um atacante ainda pode ser capaz de predizer o ISN aproximado.

2.2.5.3. Connecting Hijacking (Sequestro de uma Conexão)

Segundo Joncheray (1995), uma variação interessante do IP spoofing permite que um host se insira no meio de uma conexão entre dois hosts - sequestro de uma conexão em curso. O IP spoofing sozinho pode não passar pela segurança adicional, mas com este ataque, um atacante pode permitir a autenticação normal proceder entre os dois hosts, e então sequestrar o controle da conexão.

Conforme Chambers et al (200x), um sequestro de conexão explora um estado de “dessincronização” na comunicação TCP. Quando o número de sequência em um pacote recebido não é o número de sequência esperado, diz-se que a conexão está “dessincronizada”. Desta forma, os hosts dessincronizados descartarão pacotes um do outro. Um atacante então pode injetar pacotes forjados com os números de sequência corretos (e potencialmente modificar ou acrescentar comandos à comunicação).

Obviamente, isto requer que o atacante esteja localizado no caminho da comunicação entre os dois. Este tipo de ataque é demonstrado na **Figura 2.7**.

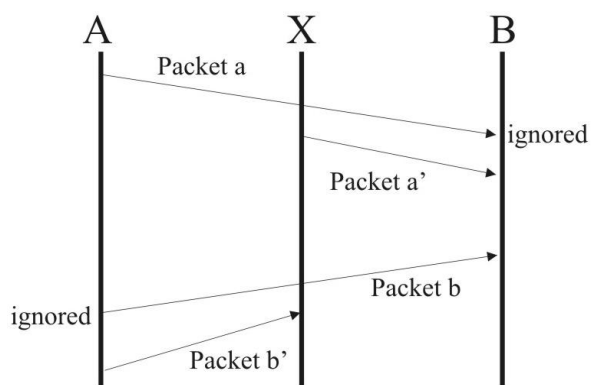


Figura 2.7 - Connection Hijacking

Um grande número de ACKs é gerado neste ataque e esta “assinatura” do ataque pode ser usada para detectar o sequestro da conexão.

De modo geral, a chave para este tipo de ataque é criar o estado de dessincronização. Joncheray (1995) descreve dois modos possíveis de fazer isto: um é durante o handshake, e o outro é no meio de uma conexão estabelecida.

2.2.5.4. Land

Conforme Menezes (2002), este tipo de ataque acontece enviando-se um pacote com a flag SYN setada para qualquer porta da máquina alvo. O pacote é modificado de tal maneira que os endereços de envio e de recebimento do pacote são idênticos (spoofing). Essa situação não é corretamente manipulada por algumas implementações do TCP/IP e a máquina alvo pára.

2.2.5.5. OOB – Out of Band

Segundo Menezes (2002), o ataque do tipo OOB, também conhecido com nuke, tem sido popular, principalmente em escolas. Esse tipo de ataque é voltado principalmente para computadores rodando Windows. O efeito principal é o congelamento do computador, que precisa ser reiniciado.

A razão pela qual o computador trava é que uma flag especial de status no protocolo TCP - a flag URG (para urgente) não é tratado adequadamente. Esse flag é setado em um pacote enviado para uma máquina, que ficará esperando por um fluxo da chamada Out-of-Band (OOB). Esse tipo de tráfego tem prioridade mais alta que o tráfego ordinário da rede. O problema surge quando tais mensagens não são enviadas e então o computador pára.

2.2.5.6. Ausência de Temporizadores

Segundo Chambers et al (200x), há uma máquina de estado para cada conexão. Cada conexão lógica inicia em um estado CLOSED, e faz transições como mostrado na **Figura 2.4**. Depois que a conexão é finalizada, o TCP retorna para o estado CLOSED. A máquina de estados do TCP é mostrada na **Figura 2.8**.

É fácil explorar algumas falhas nas máquinas de estado, e criar ataques de negação de serviço, que tentam parar a máquina de estado do TCP em um determinado estado indefinidamente ou durante um tempo finito.

Uma das vulnerabilidades da máquina de estados do TCP, segundo Chambers et al (200x), diz respeito à ausência de temporizadores em alguns estados, o que pode ser observado na **Figura 2.8**. O TCP não envia qualquer dado durante a conexão, exceto se usado com uma opção especial, o temporizador keep-alive. Isto significa que a máquina de estado de TCP de uma conexão pode ser feita para ficar em alguns estados para sempre.

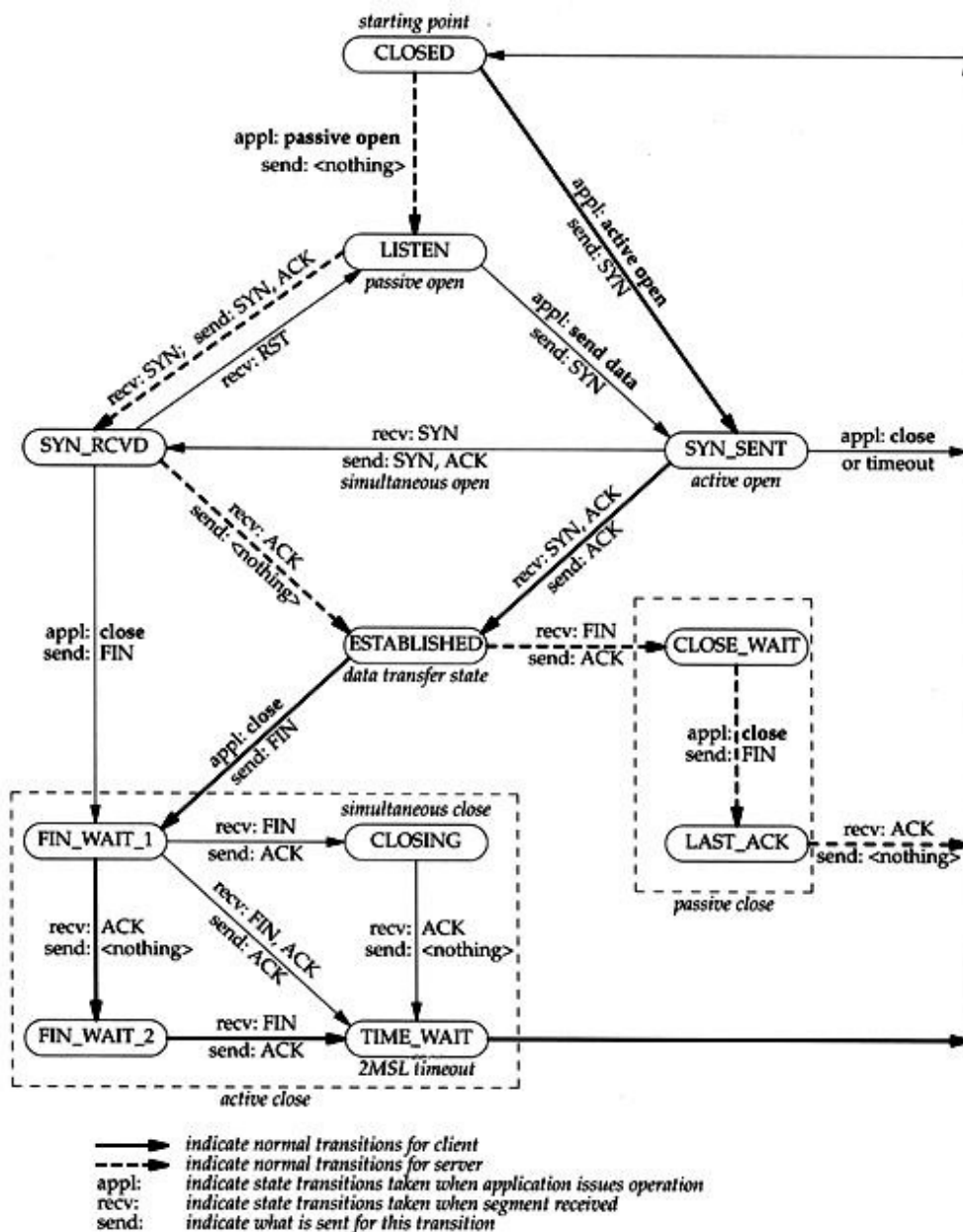


Figura 2.8 - Máquina de estados do TCP (STEVENS, 1994)

A maior parte das implementações não implementam temporizadores para estes estados. Um exemplo deste tipo de estado é o estado `CLOSE_WAIT`. Se o temporizador keep-alive é usado, o TCP seria capaz de reinicializar a conexão.

2.2.5.7. Bits SYN e FIN setados

De acordo com Guha e Mukherjee (1997), a especificação do TCP não especifica claramente certas transições e conseqüentemente permite algumas transições de estado falsas. Essas transições podem ser usadas para vários ataques, especialmente os ataques de negação de serviço.

Como no exemplo citado por Chambers et al (200x), com um cenário onde um host recebe um segmento TCP com os bits SYN e FIN setados. Recebendo um pacote com os bits SYN e FIN setados, o TCP faz uma transição ao estado `CLOSE_WAIT`. Não houve nenhum estabelecimento de conexão com sucesso e conseqüentemente o TCP não deveria fazer essa transição. A maior parte das implementações faz isto porque as especificações não conseguiram tratar esta questão. A transição é claramente perigosa, visto que `CLOSE_WAIT` é um estado sem timer associado. O receptor fica parado no estado `CLOSE_WAIT`.

2.2.5.8. Estabelecimento de Conexões Simultâneas

Quando dois hosts querem estabelecer uma conexão e ambos simultaneamente iniciam o handshake, temos um caso do estabelecimento de conexão simultâneo (RFC 793, 1981), como mostrado na **Figura 2.9**. Ambos os hosts emitem o SYN um ao outro. Quando os SYN são recebidos pelos pares correspondentes, ambos emitem um SYN+ACK. Os dois hosts devem detectar que o SYN e o SYN+ACK realmente se referem à mesma conexão. Se os dois hosts detectam que o SYN+ACK pertence ao SYN que foi recentemente enviado, eles desativam o temporizador de estabelecimento de conexão e mudam diretamente ao estado `SYN_RECV`.

Conforme Chambers et al (200x), esta falha pode ser usada para um ataque do tipo negação de serviço. Ao explorar esta falha pode-se parar uma porta em um host, usando protocolos como FTP onde o servidor inicia uma conexão para o cliente.

Segundo Stevens (1994), o TCP foi propositadamente projetado para tratar conexões simultâneas e a regra é que somente uma conexão resulta disto, não duas conexões.

Quando um estabelecimento simultâneo de conexão ocorre às transições de estado diferenciam-se das mostradas na **Figura 2.4**. Ambos os terminais enviam um SYN aproximadamente ao mesmo tempo, introduzindo o estado SYN_SENT. Quando cada terminal recebe o SYN, o estado modifica-se a SYN_RCVD, e cada terminal reenvia o SYN e confirma o SYN recebido. Quando cada um recebe o SYN-ACK, o estado muda para ESTABLISHED.

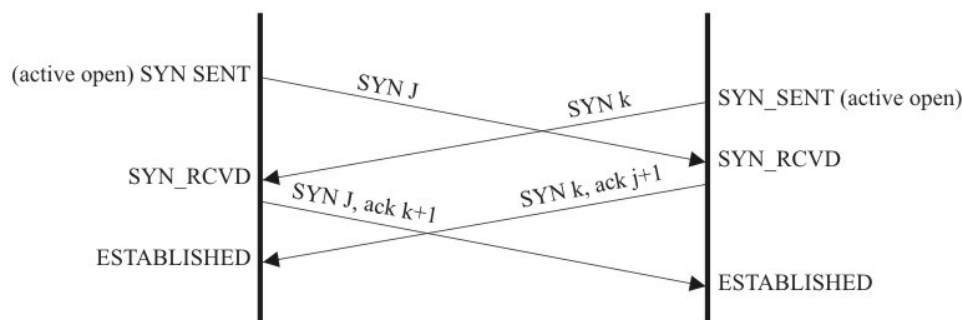


Figura 2.9 - Abertura Simultânea de Conexão (STEVENS, 1994)

Na afirmação de Stevens (1994), um estabelecimento simultâneo de conexão requer a troca de quatro segmentos, um a mais do que o handshake de três vias normal. Ambos os terminais atuam como cliente e servidor.

2.2.6. Características Especiais

O TCP fornece uma característica chamada modo urgente, descrita a seguir. Além disso, conforme Henrici (2002), o TCP permite um estado "meio-aberto", embora uma conexão TCP seja sempre bidirecional no começo, ela pode ficar unidirecional se um lado fechar a conexão e outro não o fizer.

Muito esforço foi gasto para melhorar este protocolo. Mais de cinquenta RFCs foram publicadas discutindo problemas e deficiências, propondo modificações e extensões de funcionalidade e eficiência, melhorando a heurística usada, e testando o desempenho. Questões como confirmações negativas (RFC 1106) e checksums alternativos (RFC 1146) foram discutidas.

2.2.6.1. Dados Urgentes

De acordo com Costa (2005), o TCP fornece um mecanismo para que dados presentes em determinado segmento sejam processados imediatamente no receptor, mesmo se o fluxo normal de dados foi parado pelo mecanismo de controle de fluxo, sem respeitar a fila de ordenação alocada para a conexão.

Quando o bit URG é marcado, o campo ponteiro urgente indica o primeiro octeto da área de dados do segmento que deve ser considerado urgente. Os dados urgentes terminam no último octeto da área de dados.

Esta característica pode ser usada por aplicações Telnet para enviar comandos ao servidor ou no FTP para abortar.

2.3. Stream Control Transport Protocol

Algumas aplicações atuais necessitam de um protocolo de transporte que possua características tanto do TCP quanto do UDP, além de prover outros recursos. O surgimento do SCTP (Stream Control Transmission Protocol) veio auxiliar neste ponto, visto que o mesmo é similar tanto ao TCP como ao UDP, mas com recursos adicionais.

O SCTP é um protocolo de transporte relativamente novo, foi definido oficialmente em outubro de 2000 na RFC 2960. Seu objetivo inicial era atender os requisitos de transporte das mensagens de sinalização telefônica SS7 sobre redes IP. A **Figura 2.10** apresenta a localização lógica do SCTP na pilha TCP/IP.

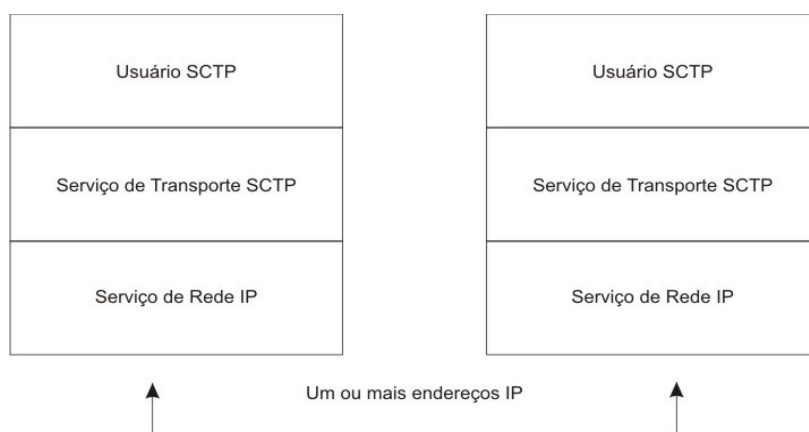


Figura 2.10 - Localização lógica do SCTP na pilha TCP/IP (COSTA, 2005)

De acordo com Costa (2005), a grande vantagem do SCTP é que ele fornece um número de funcionalidades robustas e flexíveis a diversas aplicações. Além disso, o SCTP também possibilita benefícios às aplicações que necessitem de desempenho adicional.

O SCTP apresenta todas as características de um protocolo de transporte confiável, assegurando que os dados sejam transmitidos na rede sem erros, sem duplicação e em sequência. A confiabilidade de transmissão advém do fato do SCTP ser orientado a conexão.

O SCTP é um protocolo unicast, embora os terminais participantes da comunicação possam ser representados por endereços multicast. Ele também possui controle de erro, de fluxo e de congestionamento, assim como o TCP.

2.3.1. Formato de Dados

A unidade de transferência do SCTP é conhecida como pacote. Citando Costa (2005), o pacote consiste de um cabeçalho comum de quatro campos, seguido por uma ou mais mensagens (ou blocos de informações). Para aumentar a eficiência de comunicação, múltiplas mensagens de usuário e mensagens de controle podem ser encapsuladas em um mesmo pacote SCTP.

Conforme mostra a **Figura 2.11**, citada por Santos, Ribeiro e Minicz (2008), o cabeçalho comum do pacote SCTP informa a *porta origem* e a *porta destino*, cada uma com 16 bits, também presentes no TCP e UDP, por ser um serviço básico e fundamental da camada de transporte.

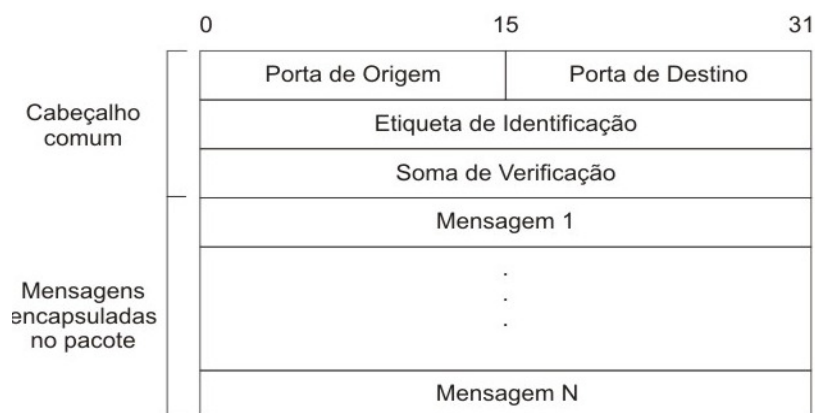


Figura 2.11 - Formato dos pacotes SCTP (SANTOS, RIBEIRO e MINICZ, 2008)

O campo *etiqueta de identificação* com 32 bits, que verifica a autenticidade do pacote para validar seu emissor, através da troca de valores gerados aleatoriamente pelas estações no estabelecimento da associação, que permanecem os mesmos durante todo o processo. O campo *soma de verificação*, ou checksum, também de 32 bits.

As mensagens SCTP também possuem formato definido, como é apresentado na **Figura 2.12**. Toda mensagem do SCTP possui uma parte fixa de quatro octetos e uma parte variável de acordo com o tipo da mensagem.

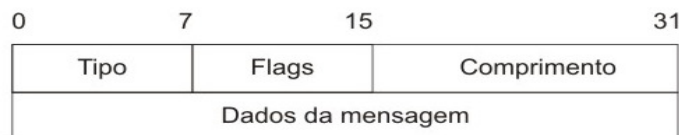


Figura 2.12 - Formato das Mensagens SCTP (COSTA, 2005, p. 36)

O primeiro campo, com 8 bits, indica o *tipo* de mensagem. O campo *flags*, também com 8 bits, oferece opções de controle (opcionais). E o campo *comprimento*, com 16 bits, indica o comprimento total da mensagem em octetos, incluindo o cabeçalho.

2.3.2. Mecanismos de Fluxo de Dados

Os mecanismos de fluxo de dados disponíveis no SCTP são o controle de fluxo e o controle de congestionamento, explicados a seguir.

2.3.2.1. Controle de Fluxo

O SCTP usa um mecanismo de controle de fluxo baseado em janelas, semelhante ao usado pelo TCP.

O receptor pode controlar a taxa de transmissão do emissor por meio da especificação do tamanho da janela (chamada janela de recepção). Esse valor é informado em todas as mensagens tipo SACK.

Todas as mensagens de dados devem ter a sua recepção confirmada (ACK) e o receptor pode esperar por certo tempo antes de retransmitir.

Por padrão, toda a transmissão é feita para um endereço previamente selecionado dentre um conjunto de endereços de destino (endereço principal). As retransmissões devem usar uma rota diferente de tal forma que se uma rota

estiver sobrecarregada, a retransmissão não piora a situação desta rota. Por outro lado, as confirmações de recebimento devem ser enviadas para o endereço do originador dos dados.

No caso da rota apresentar um alto número de falhas e for excedido um determinado limite, o processo da camada superior deve ser notificado que essa rota tornou-se inativa. Então, uma nova rota deve ser escolhida pela aplicação.

2.3.2.2. Controle de Congestionamento

De acordo com a RFC 2960, o controle de congestionamento deve impactar onde a entrega das mensagens é requerida. Isso garante um comportamento adequado do SCTP quando é introduzido em redes de pacotes de larga escala, como é o caso da Internet.

O mecanismo de controle de congestionamento do SCTP foi derivado da RFC 2581 (TCP Congestion Control) e adaptado para multi-homing. Assim, existe um conjunto de parâmetros de controle de fluxo e congestionamento para cada endereço de destino, de tal forma que, do ponto de vista da rede, uma associação SCTP que contenha um determinado número de rotas pode comportar-se de maneira semelhante ao mesmo número de conexões TCP.

Da mesma forma que o TCP, o SCTP apresenta dois modos de operação, o *slow start* e a prevenção de congestionamento. A especificação do modo de operação é feita para cada rota.

O transmissor usa uma variável conhecida como janela de congestionamento (CWND - Congestion Window), que controla o número máximo de bytes de saída que podem ser enviados mesmo sem o ACK correspondente. Para que a entrega das mensagens e o reconhecimento da recepção seja bem sucedido, esta variável é constantemente incrementada e, uma vez excedido certo limite, o modo de operação muda de *slow start* para prevenção de congestionamento.

Normalmente, quando no modo *slow start*, a variável CWND cresce rapidamente; quando no modo de prevenção de congestionamento, ela cresce mais lentamente.

2.3.3. Mecanismos de Controle de Erro

Nesta seção os principais mecanismos de controle de erro empregados no SCTP são discutidos.

2.3.3.1. Checksum

O SCTP implementa um algoritmo de soma de verificação mais robusto que o do TCP. Esse algoritmo foi atualizado na RFC 3309, em substituição ao checksum do tipo Adler-32, porque a detecção de erros do Adler-32 é pouco eficiente quanto se trata de pequenos pacotes. O novo mecanismo de checksum é o CRC-32c, um algoritmo que usa 32 bits para a verificação, deixando passar apenas 1 pacote errado em 2³² pacotes, para erros completamente aleatórios.

2.3.3.2. Confirmação e Retransmissão

Citando Santos, Ribeiro e Minicz (2008), no processo de estabelecimento de uma associação SCTP, a cada mensagem enviada um temporizador é iniciado. Caso uma das mensagens não chegue ao destino, ela será retransmitida após o vencimento do temporizador.

Segundo Farrel (2005), a transferência de dados no SCTP é controlada, como no TCP, como um único fluxo de bytes sequenciado e numerado na associação. Ou seja, cada mensagem de dados contém um TSN (Transmission Sequence Number – Número de Sequência de transmissão), que identifica o primeiro byte no contexto da associação. O SCTP pode multiplexar mais de um fluxo de dados na mesma associação.

As confirmações no SCTP utilizam, citando Farrel (2005), a confirmação seletiva, que confirma até um TSN específico (a confirmação acumulativa de TSN), indicando que todos os bytes até esse TSN, inclusive, foram recebidos com sucesso.

Esse mecanismo difere do TCP, de acordo com Farrel (2005), onde o número de confirmação indica o próximo número de sequência esperado. Uma única mensagem de confirmação seletiva pode servir as necessidades de uma associação inteira, não sendo necessária a emissão de uma mensagem de confirmação seletiva para cada fluxo.

A retransmissão seletiva acontece quando o receptor detecta intervalos na numeração seqüencial das mensagens de dados, então cada pacote SCTP será reconhecido através do envio de uma mensagem SACK (Confirmação Seletiva), onde estarão listados os intervalos.

2.3.4. Mecanismos de Gerenciamento de Conexão

O SCTP, assim como o TCP é orientado à conexão, portanto a transmissão dos dados é confiável e uma conexão precisa ser estabelecida antes da transmissão dos dados. SCTP difere-se do TCP por realizar uma associação que possui um contexto e significado mais amplo que a conexão TCP.

Conforme Costa (2005), no contexto SCTP, dois terminais estabelecem uma associação SCTP para se comunicarem. Uma associação, como uma conexão TCP, possui um caráter lógico fim a fim, sendo controlada por variáveis e protocolos da camada de transporte.

A principal diferença entre uma associação SCTP e uma conexão TCP, de acordo com Costa (2005) é que a primeira corresponde a um número arbitrário de fluxos simplex (unidirecionais) acertado durante o início da associação, enquanto a última é formada apenas por um único fluxo full duplex (bidirecional).

Citando Santos, Ribeiro e Minicz (2008), o estabelecimento de associações se dá através da troca de quatro mensagens, procedimento denominado handshake de quatro vias.

Essa forma, que difere do TCP (three-way handshake), faz com que o estabelecimento da associação seja mais seguro, uma vez que os recursos são alocados após o recebimento da terceira mensagem. Assim são evitados ataques do tipo DoS, comuns no TCP por alocar recursos logo no recebimento

da primeira mensagem. A **Figura 2.13** mostra o procedimento do handshake de quatro vias.

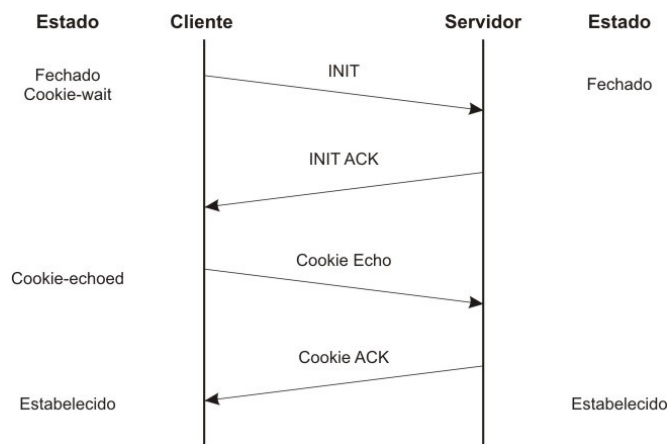


Figura 2.13 - Associação SCTP (SANTOS, RIBEIRO e MINICZ, 2008)

O mecanismo de *cookie*, descrito na RFC 2960, protege o servidor SCTP contra uma estação atacante que gere rajadas de mensagens *init* na tentativa de sobrecarregar seus recursos.

Além do mais, uma tag de verificação de 32 bits é aleatoriamente gerada para a vida de uma conexão. Ela permite que o receptor verifique que um pacote SCTP pertence à conexão atual e não é um pacote antigo ou envelhecido de uma conexão anterior. Além disso, esta tag preserva o receptor da necessidade de processar cada pacote recebido, os pacotes com uma tag de verificação desconhecida podem ser simplesmente descartados para prevenir determinados ataques de negação de serviço.

Para o encerramento de uma associação, o SCTP possui dois mecanismos, a liberação coordenada (graciosa) e a abortiva. Na coordenada não há perda de dados, porque o encerramento só é realizado após a confirmação de recebimento de todos os pacotes enviados. Nesta, o encerramento é baseado em mensagens SHUTDOWN, o terminal que deseja encerrar a associação, após confirmar o recebimento pelo terminal par de todas as mensagens, envia a mensagem SHUTDOWN, esta mensagem é respondida pelo receptor com uma mensagem SHUTDOWN ACK e, o terminal que iniciou a liberação responde com um SHUTDOWN COMPLETE e a associação é desfeita em ambos os lados.

Já na liberação abortiva, a mensagem utilizada é ABORT e o terminal que a envia libera seus recursos imediatamente após o envio, o terminal receptor ao receber a mensagem também libera seus recursos e encerra a associação.

2.3.5. Mecanismos de Segurança

Os protocolos de transporte da pilha TCP/IP, segundo Santos, Ribeiro e Minicz (2008), não foram inicialmente projetados com o objetivo de proporcionar segurança.

O TCP, conforme já citado, é sensível a ataques do tipo DoS e isto traz grande risco, já que ele é muito utilizado. Suas fraquezas se tornam muito visíveis na criação da conexão, uma vez que as partes não são autenticadas e o servidor aloca recursos antes da conexão estar completamente aberta. De fato a única proteção do TCP em relação a ataques do tipo *blind spoof* é confiar que as partes participantes gerem números de sequência de octetos iniciais imprevisíveis.

O mecanismo de handshake do SCTP resolve várias destas deficiências. Durante o processo de estabelecimento de uma associação, um *cookie* é enviado do servidor ao cliente através da mensagem *init ack* e recursos serão alocados apenas no recebimento da mensagem *cookie echo*, que retorna o *cookie* enviado sem alterações. Esse *cookie* é assinado digitalmente.

De acordo com Santos, Ribeiro e Minicz (2008), com esse mecanismo, o SCTP se garante em relação a ataques do tipo DoS, pois um *cookie* forjado será facilmente detectado por ele e descartado, não saturando a tabela de associações e não impedindo associações legítimas.

Outro mecanismo do SCTP em favor da segurança, de acordo com Costa (2005), é a utilização de etiquetas de identificação, que são empregadas na detecção de pacotes SCTP forjados.

Porém, o SCTP não oferece por conta própria garantias criptográficas (confidencialidade, autenticidade e integridade), nem é resistente a ataques do tipo *Connecting Hijacking* (Sequestro de uma Conexão).

Essas garantias podem ser providas por duas soluções usando outros protocolos, como o IPSec (Internet Protocol Security) e TLS (Transport Layer Security).

O IPSec garante integridade e confidencialidade dos dados transmitidos em uma associação SCTP, na afirmação de Santos et al (2008), no entanto, para cada endereço IP uma conexão IPSec deve ser criada, aumentando a sobrecarga. Esta forma de uso está especificada na RFC 3554.

Com o uso do TLS, que opera sobre o nível de transporte e é encapsulado na área de dados das mensagens de usuário, o problema é a desordenação dos dados de usuário, visto que o TLS utiliza comunicação orientada a fluxo, destinada a prover um serviço de entrega seqüencial de dados seguros. O SCTP não poderia enviar dados de usuário de forma desordenada (urgente).

2.3.6. Características Especiais

O SCTP se destaca por algumas características que lhe são peculiares, como múltiplos fluxos e múltiplos caminhos. Nesta seção serão apresentadas as duas principais extensões do SCTP, que são o PR-SCTP, que é o SCTP parcialmente confiável e, o SCTP Móvel.

2.3.6.1. Múltiplos Fluxos

Uma associação SCTP é full duplex, nesta associação são utilizados múltiplos fluxos (multistreamming) do tipo simplex para comunicação entre duas estações. O número de fluxos, de acordo com Costa (2005), é definido pelo usuário deste serviço no início da associação.

A representação de uma associação SCTP é mostrada na **Figura 2.14**, nela pode-se observar que fluxos de controle e de dados estão em uma mesma associação.

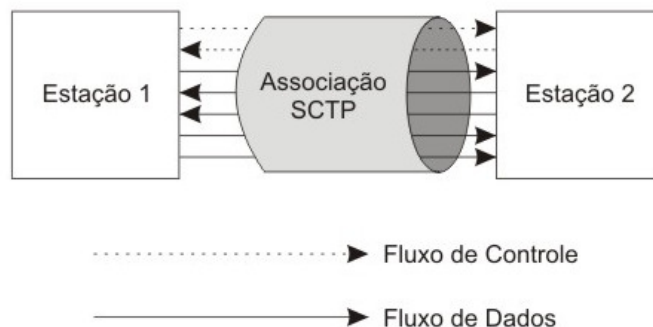


Figura 2.14 - Fluxos de uma associação SCTP (SANTOS et al, 2008, p.74)

Os dados de usuário a serem transmitidos são incluídos em mensagens de usuário SCTP. Cada mensagem de usuário é associada a um determinado fluxo da associação, recebendo, portanto um identificador denominado *número de fluxo*.

Além do identificador, cada mensagem também recebe um identificador em relação a sua posição no fluxo, chamado *número de sequência no fluxo*. As mensagens SCTP são encapsuladas em datagramas IP.

A idéia de múltiplos fluxos de transmissão e recepção, citando Costa (2005), é apenas um conceito lógico fim a fim, adotado para fins de ordenação de mensagens. Para o SCTP cada fluxo define uma fila de ordenação e um escopo de retransmissão. Dessa forma, se ganha em eficiência em caso de perdas ou erros de mensagens de usuário, pois, diferente do TCP, que possui apenas um escopo de ordenação e retransmissão, o SCTP garante que perdas de mensagens em determinado fluxo não afetem os demais fluxos da associação.

2.3.6.2. Múltiplos Caminhos

Conforme Santos, Ribeiro e Minicz (2008), a característica de múltiplos caminhos (multihoming) permite que uma associação tenha mais de um endereço IP em cada extremidade, tornando-a menos vulnerável a eventuais problemas de rede.

No estabelecimento da associação, na afirmação de Costa (2005), são ajustados dois endereços primários (uma para cada estação), além de um número qualquer de endereços alternativos. Para todos esses endereços, uma única porta de transporte é utilizada por cada estação em determinada associação.

Quando uma estação detecta um erro que torna inalcançável o endereço primário da outra estação, automaticamente um dos endereços alternativos é usado para a transmissão das mensagens. Os endereços alternativos podem, inclusive, citando Santos, Ribeiro e Minicz (2008), ser configurados em interfaces de naturezas diferentes.

2.3.6.3. SCTP Parcialmente Confiável

Conforme Costa (2005), para tornar o SCTP mais flexível em ambientes onde a confiabilidade dos dados não é requisito primário, como aplicações em tempo real, foi especificada uma extensão do SCTP, onde um nó SCTP pode informar ao outro que mensagens pedidas não serão retransmitidas. Esta extensão é conhecida como PR-SCTP (Partial Reliability SCTP – SCTP Parcialmente Confiável) e foi definida na RFC 3758.

Também é especificado um mecanismo para que o protocolo de nível superior saiba se o serviço de confiabilidade parcial está disponível ou não. Os elementos adicionados ao SCTP padrão não alteram sua forma de funcionamento.

A utilização do PR-SCTP traz alguns benefícios como:

- Em uma única associação SCTP podem existir dados com características confiáveis e não-confiáveis.
- Dados parcialmente confiáveis têm a mesma detecção de falha de comunicação e proteção de tráfego dos dados confiáveis.
- Transmissão de dados de forma ordenada e não-confiável e, não-ordenada e não-confiável.
- Mesmo esquema de controle de congestionamento que a transmissão confiável.

- Diminuição do número de datagramas IP e do overhead na rede porque um mesmo protocolo é utilizado para transporte confiável e não-confiável.

Não é usual utilizar o protocolo TCP para aplicações que não necessitem de confiabilidade ou manutenção da ordem na transmissão de dados porque causaria um overhead de processamento desnecessário, prejudicando o desempenho destas aplicações. Neste caso, o UDP seria mais adequado, porém todas as características de controle benéficas do SCTP não estariam disponíveis. Com esta extensão o SCTP é uma solução melhor em relação ao TCP e, em alguns cenários, que o UDP.

2.3.6.4. SCTP Móvel

De acordo com Costa (2005), novas extensões permitem que os endereços IP ligados a uma associação possam ser configurados (ou reconfigurados) dinamicamente, com isso permite-se que estações com requisitos de mobilidade adquiram um novo endereço e informem-no ao seu par da associação, sem qualquer perda de comunicação. A mobilidade é uma característica cada vez mais necessária na Internet, onde o número de acessos sem fio e estações móveis aumenta a cada dia.

O SCTP Móvel é uma extensão que permite explorar a característica de multihoming do SCTP, juntamente com mecanismos de configuração dinâmica de endereços, para permitir operações de handover (troca das informações dos novos endereços das estações participantes da comunicação) numa arquitetura com demanda por comunicação móvel, garantindo assim mobilidade de terminal, que nada mais é que a possibilidade de uma estação mudar de ponto de acesso à rede sem encerrar a comunicação em andamento.

As características adicionais ao SCTP padrão são: duas novas mensagens de controle, seis novos tipos de parâmetros e cinco novas causas de erro. O MSCTP tem evoluído desde 2001, ainda em forma de draft, mas deve ser incorporado à próxima RFC do SCTP.

2.4. Xpress Transport Protocol

Os protocolos da camada de transporte atuais, como o TCP, não foram desenvolvidos para as redes de alta velocidade. Tendo em vista este cenário o Xpress Transport Protocol (XTP) foi desenvolvido para ser um protocolo de alto desempenho projetado para satisfazer as necessidades de sistemas distribuídos, sistemas de tempo real e sistemas multimídia, em ambientes unicast ou multicast.

Conforme Sander et al (2007), o XTP foi desenvolvido por um grupo de pesquisadores e desenvolvedores coordenados pelo Protocol Engines Incorporated (PEI). A ênfase no desenvolvimento do XTP tem sido a provisão dos mecanismos básicos, a partir dos quais o usuário pode construir um serviço bem adaptado à aplicação.

De acordo com Strayer (1995), o protocolo XTP é considerado um protocolo leve por várias razões. Principalmente por ser bastante simples, mas ter um algoritmo flexível. E também porque o cabeçalho do pacote é de tamanho fixo e contém informações suficientes para que o mesmo seja exibido e direcionado através da rede.

O XTP é um protocolo bastante flexível, que consiste em um número independente de funcionalidades, como controle de fluxo, controle de erro, controle de taxa, entrega prioritária de dados, unicast / multicast, possui diversas semânticas de envio de dados, além de endereçamento parametrizado e especificação de tráfego.

2.4.1. Formato de Dados

Um pacote, de acordo com Strayer (1995), é basicamente a unidade de informação trocada entre nós de uma associação XTP. Os campos dentro de um pacote contêm a informação sobre o estado de uma associação ou dos dados sendo transferidos. A estrutura de um pacote facilita a recuperação da informação contida dentro dele.

Cada pacote XTP inclui uma estrutura de cabeçalho comum seguida por um segmento de payload, como mostrado na **Figura 2.15**. Segundo Sander e Weaver (2007), o cabeçalho do pacote XTP tem tamanho fixo e contém informações suficientes para exibir e direcionar os pacotes através da rede.

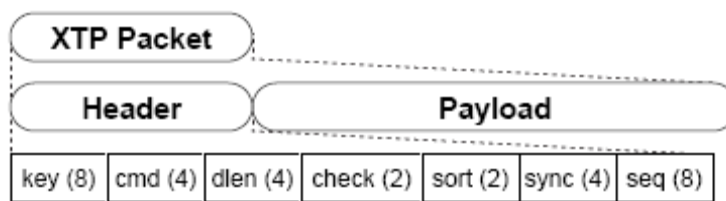


Figura 2.15 - Pacote XTP (STRAYER, 1995, p. 14)

O cabeçalho é composto dos seguintes campos: *key*, *cmd* (command), *dlen* (data length), *check* (checksum), *sort* (priority), *sync* (synchronizing number), e *seq* (sequence number). O campo *key* guia o pacote ao contexto de destino apropriado. O campo *cmd* dita como o pacote deve ser processado. Os campos *dlen* e *seq* identificam os conteúdos deste pacote com respeito ao fluxo de dados. Os campos *check* e *sync* são usados para determinar a validade do pacote, e o campo *sort* ordena o ato de analisar o pacote entre todas as atividades confirmadas.

Há dois tipos básicos de pacotes no XTP, de acordo com Strayer (1995), pacotes de controle e pacotes de informações, como mostrado na **Figura 2.16**.

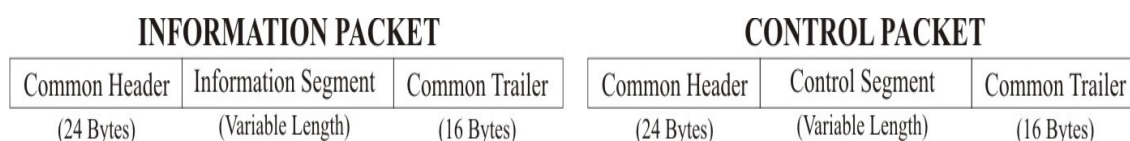


Figura 2.16 - Formato dos pacotes XTP (SANDER e WEAVER, 2007)

Os pacotes de controle incluem um Segmento de Controle e são usados para trocar informação de estado do protocolo entre contextos em uma associação. A informação em pacotes de controle, citando Strayer (1995), não é dada diretamente ao usuário, mas é usada pelo contexto para efetuar os algoritmos de controle.

A informação de usuário, na afirmação de Strayer (1995), incluindo dados de usuário e mensagens de diagnóstico do protocolo, é incluída no Segmento de Informações dos pacotes de informação.

O design do pacote XTP, conforme Strayer (1995), assegura que todo segmento majoritário e todos os campos de 8 bytes iniciem no limite de 8 bytes, e todos os campos de 4 bytes iniciem no limite de 4 bytes. Os campos de tamanho variável são evitados, e os segmentos de tamanho variável sempre têm um método direto para determinar o seu tamanho.

2.4.2. Mecanismos de Fluxo de Dados

Os mecanismos de fluxo de dados presentes no XTP são o controle de fluxo e o controle de taxa, descritos a seguir.

2.4.2.1. Controle de Fluxo

O volume de saída do XTP, de acordo com Strayer (1995), é regulado por um mecanismo de controle de fluxo de janela deslizante fim a fim, baseado em números de sequência. Um número de sequência é destinado a cada byte de saída do fluxo de dados. A taxa na qual o XTP envia pacotes na rede é regulada por um mecanismo independente, à base de timer, que será visto posteriormente.

Dois campos em pacotes de controle, citando Strayer (1995), são usados nos procedimentos de controle de fluxo, o campo *alloc* e o campo *rseq*. O valor do campo *alloc* representa a borda superior da janela de controle de fluxo, isto é, indica o número de sequência a não ser excedido pelo transmissor. O valor do campo *rseq* representa a borda inferior da janela de controle de fluxo, ele é o valor do último byte contíguo recebido acrescentado em um.

Duas flags (de 1 bit) opcionais modificam o modo como o controle de fluxo é tratado: RES e NOFLOW. O bit *RES* enviado pelo transmissor indica ao receptor o valor *alloc* deve refletir o espaço de buffers tão somente como o

usuário alocou para a associação, isto é chamado de modo reserva. O bit *NOFLOW* indica ao receptor que o controle de fluxo nesta direção será desativado (STRAYER, 1995, p. 65).

2.4.2.2. Controle de Taxa

De acordo com Strayer (1995), o controle de taxa controla a relação consumidor/produtor entre terminais XTP. O controle de taxa está preocupado com a velocidade que os pacotes e os seus conteúdos podem ser processados, ou consumidos, no receptor.

Os parâmetros de diminuição da taxa da produção, na afirmação de Strayer (1995), podem ser realimentados ao transmissor através dos parâmetros de controle de taxa do especificador de tráfego, que tem parâmetros de controle de taxa explícitos. Se os parâmetros de controle de taxa explícitos estiverem disponíveis, os parâmetros default *rate* e *burst* devem ser usados.

A taxa de pacote de saída é regulada por duas variáveis de contexto, *credit* e *burst*, e por um timer de atualização chamado RTIMER.

Os valores de *credit* e *burst* são calculados, de acordo com a partir de *rate* e *burst*. O valor *rate* especifica a taxa de dados máxima em bytes por segundo. O valor *burst* especifica o número máximo de bytes a ser enviado em uma rajada de pacotes. O valor de *rate* dividido pelo valor de *burst* dá o número de transmissões de tamanho de rajada por segundo, ou a taxa na qual a variável *credit* é atualizada. O valor *burst* dividido pelo valor *rate* dá o período de tempo de RTIMER.

A transmissão de dados deve cessar quando *credit* fica em zero ou negativa. Quando o RTIMER expira, a variável interna *credit* é atualizada com o valor de *burst*. Um valor zero para *burst* significa que este contexto não é restringido, e pode transmitir à vontade. Um valor zero para *rate* pára a transmissão de dados completamente.

Para Strayer (1995), o controle de taxa em uma direção ao longo de um caminho XTP é distinto e pode diferenciar-se da taxa no sentido contrário.

2.4.3. Mecanismos de Controle de erro

Conforme Strayer (1995), o controle de erro no XTP é baseado na troca da informação quanto a dados perdidos ou corrompidos e a retransmissão desses dados. Cada pacote é examinado para identificar se houve dano, executando um checksum. Os dados perdidos são detectados e recuperados usando os procedimentos de confirmação e retransmissão.

O controle de erros no fluxo de dados, na afirmação de Strayer (1995), pode ser desligado pelo transmissor setando o bit NOERR (modo sem erro) em todos os pacotes de saída, desta forma um pacote ECNTL nunca será usado.

Na sequência são descritos os principais mecanismos utilizados pelo XTP para controle de erro.

2.4.3.1. Checksum

A função de checksum do XTP, citando Strayer (1995), é uma soma de complemento de um, de 16 bits, sobre todos os pares de octetos participantes (se o número do octeto for ímpar, é acrescentado um octeto de dados com valor zero para fechar o último par). O checksum é executado sobre o pacote inteiro se o bit *NOCHECK* estiver em zero, e somente sobre o cabeçalho se o bit *NOCHECK* estiver setado.

2.4.3.2. Confirmação e Retransmissão

Conforme Sanders e Weaver (2007), quando acontecem erros na transmissão, o XTP deve detectar e começar o processo de retransmissão dos dados com problemas. Um receptor detecta a falta de pacotes, de acordo com Strayer (1995), verificando o seu intervalo de fluxo de pacotes de entrada no espaço de sequência. O receptor registra os dados faltantes guardando o número de sequência do primeiro byte faltante, e opcionalmente guardando uma lista de valores possíveis de dados corretamente recebidos.

Citando Strayer (1995), quando um pacote de controle está para ser enviado, o contexto verifica para ver se algum dado está faltando. Se não houver nenhuma falta de dados, então um pacote CNTL é usado. Este pacote acusa o recebimento de todos os dados cujos números de seqüência são menores do que o valor no campo *rseq*.

Se dados faltantes tiverem sido detectados, na afirmação de Strayer (1995), um pacote ECNTL é enviado. Usando o campo *rseq* neste pacote pode-se confirmar dados da mesma maneira que no pacote CNTL, os campos *nspan* e *spans* são usados para confirmação seletiva dos valores dos dados recebidos. O recebimento de um pacote ECNTL significa que alguns dados foram perdidos. Os campos *rseq*, *nspan*, e *spans* especificam que dados foram perdidos.

O transmissor pode retransmitir dados cujo número de seqüência inicia em *rseq* e vai até o maior número de seqüência enviado por ele. Isto, conforme Strayer (1995), é a retransmissão go-back-N. Este procedimento difere do TCP, onde cada pacote de dados é retransmitido depois do timeout.

De acordo com Strayer (1995), se o transmissor tiver setado o bit FASTNAK em pacotes de saída, que indica “modo de confirmação negativa rápida”, o receptor é orientado a enviar pacotes ECNTL sem ser primeiramente pedido para fazer isso.

2.4.3.3. Handshake de Sincronização

Para controlar erros, o XTP utiliza também o handshake de sincronização que, para Strayer (1995), é uma troca de pacote de controle onde o campo *sync* no pacote de saída deve combinar com o campo *echo* no pacote de entrada. Depois deste procedimento o transmissor conhece o estado do receptor. De Não é permitido o envio de nenhum pacote de dados durante um handshake de sincronização, inclusive dados retransmitidos, a retransmissão pode acontecer assim que o handshake foi concluído.

2.4.4. Mecanismos de Gerenciamento de Conexão

O XTP, assim como o TCP é orientado à conexão. Antes dos dados serem transferidos, uma conexão precisa ser estabelecida entre uma estação emissora e uma estação receptora. Após são negociadas as capacidades para uso da conexão.

O XTP é projetado para as redes de baixa taxa de erro de hoje e assim não emprega muito tempo com mecanismo de handshake. A especificação do número de sequência inicial do fluxo de dados não é necessária, uma vez que o espaço de sequência sempre começa em zero. Por isso, uma conexão é estabelecida implicitamente enviando um pacote com a flag FIRST setada, que já pode conter dados. Se a conexão não for aceita pela entidade receptora todos os dados recebidos são simplesmente descartados, e um pacote "DIAG" é enviado como notificação.

A coleção da informação que compreende o estado do XTP em um sistema é chamada de contexto. Esta informação representa uma instância de uma comunicação ativa entre dois ou mais terminais XTP.

Segundo Strayer (1995), um contexto deve ser criado, ou instanciado, antes do envio ou recebimento de pacotes. Pode haver muitos contextos ativos em um sistema, um para cada conversação ativa. Cada contexto gerencia tanto um fluxo de dados de saída como de entrada.

O agregado de contextos ativos e os fluxos de dados entre eles são chamados de associação.

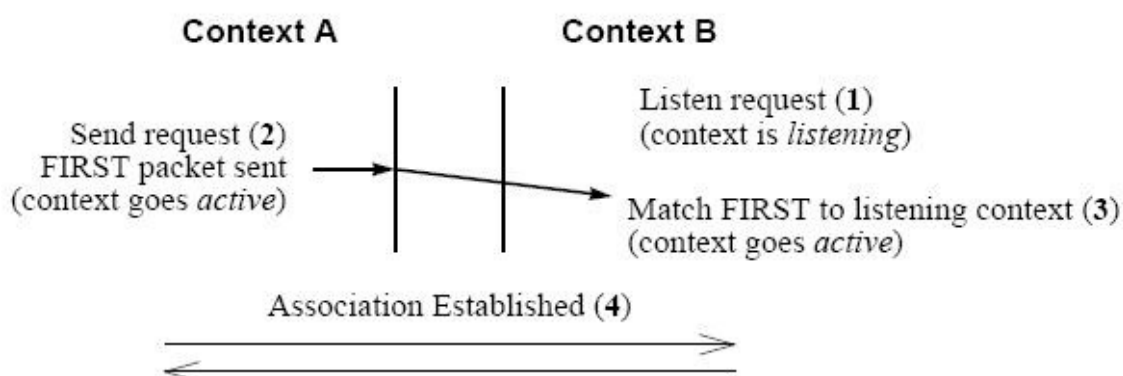


Figura 2.17 - Estabelecimento de Associação XTP (STRAYER, 1995, p. 3)

A **Figura 2.17** mostra o estabelecimento de uma associação (unicast). Para Strayer (1995), um contexto em um sistema está inicialmente em um estado inativo. Um usuário esperando o início de uma associação requisita que o contexto seja colocado no estado de escuta (1). O contexto agora fica na escuta por um pacote FIRST apropriado. O pacote FIRST é o pacote inicial de uma associação, ele contém a informação de endereçamento explícito. O usuário deve fornecer toda a informação necessária do XTP para combinar um pacote FIRST de entrada com o contexto de escuta.

No XTP cada fluxo unidirecional de dados pode ser fechado separadamente usando um handshake de duas vias, alternativamente fechamento gracioso ou abortivo.

Para o fechamento, a flag "WCLOSE" (entidade que quer encerrar seu fluxo de dados de escrita), a flag "RCLOSE" (entidade que encerrou seu fluxo de dados de leitura), e a flag "END" (conexão liberada) nos pacotes de controle são empregadas.

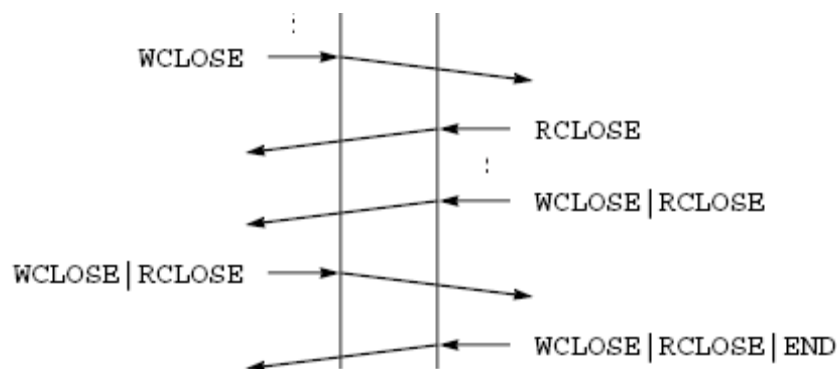


Figura 2.18 - Fechamento Gracioso Completo (STRAYER, 1995, p 60)

Para o fechamento gracioso, existem várias maneiras de realização, conforme citado por Strayer (1995), que podem ser: fechamento gracioso completo e independente, fechamento gracioso abreviado e fechamento forçado. Além do fechamento abortivo citado, o XTP pode fechar uma conexão por inatividade. A **Figura 18** demonstra o fechamento gracioso completo e independente.

2.4.5. Mecanismos de Segurança

O XTP não possui mecanismos de segurança e, como verificado pelo estabelecimento de conexão implícito, é assim como o TCP, sujeito a ataques de negação de serviço.

2.4.6. Características Especiais

Dentre várias características que diferenciam o XTP, citaremos nesta seção a principal delas, o multicast.

2.4.6.1. Multicast

O multicast de camada de transporte, segundo Strayer (1995) é uma característica única no XTP. Ele não existe em outros protocolos de transporte bem conhecidos, como TCP e UDP. As aplicações potenciais do multicast são tão numerosas que o multicast é a característica mais importante e que mais distingue o XTP.

O multicast não é um anexo ao unicast; cada mecanismo usado para comunicações unicast está disponível para o uso também no multicast. O multicast XTP prove um mecanismo poderoso de comunicação em grupo que suporta um serviço de transferência de dados de um para muitos fluxos de dados.

Como afirma Strayer (1995), o multicast XTP é planejado para meios de comunicação que provêem uma facilidade broadcast ou multicast. Este serviço também pode ser estendido para meios de comunicação não multicast através de um serviço de entrega de dados que prove replicação de saída para um grupo de destinos.

Um transmissor multicast pode enviar a um grupo de receptores arbitrariamente grande. Uma vez que isto é um multicast da camada de transporte e não um multicast ou broadcast da camada de enlace, os

procedimentos de controle de fluxo, controle de taxa e controle de erro são aplicados à transmissão de mensagens de tamanho arbitral a grupos de tamanho arbitral.

Citando Strayer (1995), o multicast do XTP prove os mesmos algoritmos e mecanismos de controle que o unicast. A diferença fundamental entre multicast e unicast é que não há nenhum conceito de transferência de dados duplex, como consequência, o multicast XTP é um fluxo de dados simplex de um transmissor a um número arbitrário de receptores. Enquanto que ambos os terminais de uma conversação unicast XTP têm um lado de envio e um de recepção para comunicação duplex, os terminais multicast são unidirecionais. Os receptores não podem enviar dados ao transmissor, por isto, há uma distinção entre o transmissor (possui somente o lado emissor ativo) e os receptores (possuem somente o lado receptor ativo).

O gerenciamento de associação no multicast está intimamente relacionado ao gerenciamento do grupo de receptores. Como em unicast, o estabelecimento de uma associação multicast é o processo de descoberta de receptores. Diferentemente do unicast, o tamanho do grupo de receptores pode crescer ou diminuir durante o tempo de vida da associação.

O XTP não impõe políticas para gerenciar o grupo de receptores, uma vez que esses são aplicações e interfaces específicas, mas o XTP prove os mecanismos de admissão, rejeição, e dispensa de membros.

O multicast refere-se a um transmissor único com múltiplos receptores, que prove procedimentos para transmitir um fluxo de dados na ordem sequencial e sem dados.

Os pacotes multicast obedecem às mesmas regras de sintaxe que pacotes não multicast: o cabeçalho, o Segmento de Controle, e o Segmento de Informação são idênticos. Como afirma Strayer (1995), os pacotes multicast diferenciam-se dos pacotes unicast nos seguintes pontos:

- a) Todos os pacotes em uma associação multicast têm o bit MULTI setado, enquanto os pacotes em uma associação unicast sempre têm este bit em zero.
- b) Os pacotes enviados pelo transmissor multicast podem utilizar o endereço de grupo ou um endereço unicast de um receptor individual.

- c) No estabelecimento de uma associação multicast, os pacotes de controle enviados ao transmissor pelos receptores devem usar o endereço de unicast do transmissor.
- d) Todos os pacotes enviados do transmissor têm o bit RCLOSE setado.

Todos os outros campos e flags de um bit definidos para a transmissão unicast são definidos para multicast.

De acordo com Strayer (1995), os receptores multicast unem-se a uma associação multicast de dois modos. O primeiro método é quando uma escuta de contexto em um endereço multicast recebe um pacote FIRST multicast. O segundo ocorre quando um contexto envia um pacote JOIN ao transmissor multicast de uma associação em progresso, e o transmissor responde com um pacote JOIN, admitindo o contexto ao grupo.

O transmissor mantém a informação sobre todos os receptores ativos na associação. Quando um receptor recebe um pacote de controle, os mesmos procedimentos de associações unicast são aplicados a ele.

Os receptores podem abandonar o grupo enviando um pacote CNTL com o bit END setado. A associação multicast é finalizada somente quando o transmissor multicast envia um pacote com o bit END setado.

3. Estudo Comparativo dos Protocolos de Transporte

Após a realização do estudo das características e dos principais mecanismos dos protocolos TCP, SCTP e XTP, surge a necessidade de fazer um comparativo dos mesmos, levantando assim as vantagens e características únicas de cada protocolo analisado. Além disso, a informação de desempenho dos protocolos em testes empíricos auxilia na identificação do protocolo de transporte mais adequado para as aplicações atuais.

Neste capítulo os protocolos TCP, SCTP e XTP serão comparados em relação aos mecanismos que um protocolo de transporte deve fornecer, citados no Capítulo 2. Posteriormente será apresentada a metodologia utilizada na realização dos testes com estes protocolos, descrevendo os equipamentos e redes utilizados, bem como os aplicativos e sistemas operacionais. Para finalizar são apresentados os testes realizados e seus resultados.

3.1. Comparativo dos Mecanismos do TCP, SCTP e XTP

Conforme o estudo apresentado no Capítulo 2, os protocolos analisados atendem a vários mecanismos e características definidos para a camada de transporte. Dentre os principais está à confiabilidade que, como pode ser observado na **Tabela 3.1**, é garantida pelos três. A **Tabela 3.1** mostra as principais características dos protocolos TCP, SCTP e XTP. Estes mecanismos e características são discutidos a seguir.

Tabela 3.1 - Comparativo Geral

Característica	TCP	SCTP	XTP
Confiabilidade	Sim	Sim	Sim
Orientado à Conexão	Sim	Sim	Sim
Orientação da Transmissão	Octeto	Mensagem	Mensagem
Controle de Fluxo	Sim	Sim	Sim
Controle de Congestionamento	Sim	Sim	Não
Controle de Taxa	Não	Não	Sim

Característica	TCP	SCTP	XTP
Distribuição dos Dados	Ordenada	Ordenada Ordenada Parcial Desordenada	Ordenada
Múltiplos Fluxos	Não	Sim	Não
Multiplos Caminhos	Não	Sim	Não
Multicast	Não	Não	Sim
Pseudo-header para Checksum	Sim	Não	Sim
Permite Conexão Meio-abertas	Sim	Não	Sim
Proteção contra Ataques DoS	Não	Sim	Não

Os mecanismos apresentados no Capítulo 2 são identificados a seguir em cada protocolo analisado.

3.1.1. Mecanismos de mapeamento de unidades de dados

Em relação ao mecanismo de mapeamento de unidades de dados pode-se comparar os protocolos em termos de fragmentação e segmentação, blocking e concatenação, como será apresentado a seguir.

3.1.1.1. Fragmentação e Segmentação

Ao examinar os protocolos TCP, SCTP e XTP, verifica-se que a fragmentação é encontrada somente no SCTP, não estando presente no TCP e no XTP, como pode ser visto na **Tabela 3.2**.

Tabela 3.2 - Mecanismo de Fragmentação

Fragmentação	TCP	SCTP	XTP
	-	X	-

O SCTP usa um número de sequência de transmissão de 32 bits (TSN) que enumera os chunks de dados (uma PDU pode conter vários chunks) e números de sequência de fluxo (SSN) de 16 bits que enumera chunks de dados que pertencem a um fluxo particular. A fragmentação é transparente para a camada superior.

Já a segmentação pode ser encontrada no TCP e no XTP, não estando presente no STCP, como pode ser observado na **Tabela 3.3**.

Tabela 3.3 - Mecanismo de Segmentação

Segmentação	TCP	SCTP	XTP
	X	-	X

No TCP e no XTP a segmentação significa que uma simples TSDU é mapeada para várias TPDU's, para que não seja mais necessário realizar divisões de dados em camadas mais baixas. Para fazer isso, o tamanho máximo do segmento (MSS) é calculado baseado no atual MTU do caminho. Como os segmentos podem alcançar seu destino fora de ordem, deve haver um meio de reordenar os segmentos antes da remontagem. Isto é realizado usando números de sequência.

No TCP cada byte de dados é enumerado usando um número de sequência de 32 bits. Um cabeçalho TCP contém o número de sequência pertencente ao primeiro byte de dados no segmento.

O XTP trabalha de um modo semelhante; cada byte é representado por um número de sequência, mas ao contrário do TCP, ele usa números de sequência de 64 bits.

A segmentação é transparente à camada superior.

3.1.1.2. Blocking e Concatenação

Os protocolos TCP e SCTP possuem o mecanismo de blocking e, ele é opcional, podendo ser desabilitado. Enquanto o XTP não possui tal mecanismo, como pode ser observado na **Tabela 3.4**.

Tabela 3.4 - Mecanismo de Blocking

Blocking	TCP	SCTP	XTP
	[X]	[X]	-

A entidade TCP pode decidir não enviar dados da camada superior imediatamente e tentar reunir uma quantidade maior de dados antes do envio. O objetivo é reduzir o número de segmentos pequenos. Para isso o algoritmo

de Nagle, proposto na RFC 0896, é comumente usado, se não ele pode ser desabilitado.

Em um segmento SCTP são empacotados tantos chunks quanto possível, para reduzir o número de segmentos. Em implementações onde o envio imediato de chunks é atrasado para estimular o empacotamento este mecanismo pode ser desabilitado, mas em caso de congestionamento, o empacotamento sempre ocorrerá.

Já o mecanismo de concatenação não é usado por nenhum dos protocolos analisados neste trabalho. A maioria dos protocolos mais recentes, como o XTP, não faz uso de quaisquer destes mecanismos porque na atualidade, onde cada vez mais a tendência é de redes de alta velocidade, a economia de recursos de rede já não compensa o aumento do atraso e recursos de processamento provocados por esses mecanismos.

3.1.2. Mecanismos de Delimitação de Mensagem

Dos protocolos analisados, apenas o TCP não separa mensagens. O SCTP e o XTP empregam tal mecanismo, como é mostrado na **Tabela 3.5**.

Tabela 3.5 - Mecanismo de Delimitação de Mensagem

Delimitação de Mensagem	TCP	SCTP	XTP
	-	X	X

O XTP tem um flag EOM ("fim de mensagem") no cabeçalho de cada PDU para marcar os limites de uma mensagem de usuário. Um pacote de dados contendo a flag EOM transporta os bytes finais de uma mensagem.

O SCTP usa dois bits no cabeçalho de chunks de dados para indicar os limites da mensagem: um bit de "fragmento iniciando", e um bit de "fragmento terminando".

3.1.3. Mecanismos de Multiplexação

Para fins deste estudo, os mecanismos de multiplexação considerados são de multiplexação de aplicações e de fluxo.

Os três protocolos analisados possuem o mecanismo de multiplexação de aplicações, como pode ser verificado na **Tabela 3.6**.

Tabela 3.6 - Mecanismo de Multiplexação de Aplicação

Multiplexação de Aplicação	TCP	SCTP	XTP
	X	X	X

O TCP e o SCTP usam números de porta de 16 bits como identificadores. Os números de endereço do host e porta formam "sockets". A união do socket fonte e socket destino identifica unicamente um canal de comunicação.

Na fase de estabelecimento de conexão, o XTP usa a mesma técnica se usado com IP como protocolo inferior. Depois, um identificador "key" pode ser empregado ao invés de um socket.

Já a multiplexação de fluxo, como pode ser observado na **Tabela 3.7**, é usada apenas pelo SCTP e, é opcional. No SCTP a camada superior pode especificar o número de fluxos independentes que será suportado durante o tempo que a conexão estiver estabelecida. Mensagens de dados são associadas com números de fluxo (identificador de 16 bits) para identificar o fluxo ao qual o dado pertence.

Tabela 3.7 - Mecanismo de Multiplexação de Fluxo

Multiplexação de Fluxo	TCP	SCTP	XTP
	-	X	-

3.1.4. Mecanismos de Gerenciamento de Caminho

Os mecanismos de gerenciamento de caminho considerados são o heartbeat/keep-alive e o is-alive. Como mostrado na **Tabela 3.8**, o SCTP e o XTP oferecem o mecanismo de keep-alive.

No SCTP a acessibilidade é verificada periodicamente pelo envio de uma requisição heartbeat para a qual o receptor deve responder com uma confirmação de heartbeat se não houver mais dados para transmitir. Além disso, a camada superior é informada imediatamente se uma alteração de estado de um caminho é detectada. Já no XTP, um nó periodicamente requisita um handshake de sincronização com seu host correspondente setando uma SREQ, depois disso, o transmissor conhece o estado do receptor.

Tabela 3.8 - Mecanismo de Heartbeat/Keep-alive

Heartbeat/Keep-alive	TCP	SCTP	XTP
	[X]	X	X

No TCP este mecanismo não é obrigatório, apesar de que muitas implementações oferecem ele. Depois de um tempo de inatividade de pelo menos 2 horas, de acordo com a RFC 0793, um pacote keep-alive é enviado e repetido várias vezes, se não é recebido uma confirmação a conexão é reiniciada.

Pela **Tabela 3.9**, podemos observar que somente o SCTP possui o mecanismo is-alive e, ele é opcional. No SCTP a camada superior pode explicitamente emitir um pedido de status ou disparar um pedido heartbeat para verificar a disponibilidade do destino ao longo de um determinado caminho.

Tabela 3.9 – Mecanismo de Is-alive

Is-alive	TCP	SCTP	XTP
	-	[X]	-

O SCTP, como mostrado na **Tabela 3.10**, possui o mecanismo de troca de caminho e, ele é opcional. Quando a conexão é estabelecida no SCTP, vários endereços de transporte completos que o destino pode ser alcançado podem ser especificados. Os caminhos correspondentes são checados pelo SCTP periodicamente. Se a monitoração detecta que um caminho está inativo, o SCTP troca para um ativo disponível.

Tabela 3.10 - Mecanismo de Troca de Caminho

Troca de Caminho	TCP	SCTP	XTP
	-	[X]	-

3.1.5. Mecanismo de Fluxo de Dados

Os mecanismos de fluxo de dados analisados são controle de fluxo, controle de congestionamento e controle de taxa.

3.1.5.1. Controle de Fluxo

Pela **Tabela 3.11** pode-se observar que os três protocolos possuem mecanismo de controle de fluxo e, no XTP ele é opcional.

Tabela 3.11 - Mecanismo de Controle de Fluxo

Controle de Fluxo	TCP	SCTP	XTP
	X	X	[X]

O TCP usa um número de confirmação de 32 bits como borda inferior da janela de controle de fluxo e um campo *window* de 16 bits para especificar o seu tamanho. Na RFC 1106 a opção "*big-window*" é introduzida para tornar possíveis janelas maiores.

O SCTP oferece um crédito de janela de receptor anunciada de 32 bits representando o espaço de buffer dedicado reservado para esta associação e mantém uma janela que limita os dados que um emissor pode enviar para um destino em particular.

O XTP usa um campo *rseq* de 64 bits na sequência do próximo byte esperado que também serve como borda inferior da janela de controle de fluxo. Como borda superior, um campo *alloc* mantém o número de sequência até que o receptor esteja disposto a aceitar dados. Duas flags de bit opcionais modificam a maneira como o controle de fluxo é controlado, como apresentado na seção 2.4.

3.1.5.2. Controle de Congestionamento

Na **Tabela 3.12**, nota-se que o TCP e o SCTP possuem o mecanismo de controle de congestionamento. Percebe-se também que o XTP possui este recurso, porém, ele é opcional e para funcionar precisa de extensões.

Tabela 3.12 - Mecanismo de Controle de Congestionamento

Controle de Congestionamento	TCP	SCTP	XTP
	X	X	[(X)]

No TCP o controle de congestionamento é realizado de forma a identificar e prevenir congestionamento, como descrito na seção 2.2.

O controle de congestionamento do SCTP é baseado no mecanismo usado no TCP.

Para o XTP o controle de taxa é o mecanismo preferido para inibir congestionamento. Como o mecanismo requer que todos os nos intermediários suportem o XTP, o que não é um caso usual, as implementações podem adicionalmente empregar heurísticas conhecidas do TCP para evitar congestionamento, mas isto não faz parte da especificação do XTP.

3.1.5.3. Controle de Taxa

Pela **Tabela 3.13** pode-se observar que apenas o XTP possui o mecanismo de controle de taxa e ele é opcional.

Tabela 3.13 - Mecanismo de Controle de Taxa

Controle de Taxa	TCP	SCTP	XTP
	-	-	[X]

O XTP prove controle de taxa, por meio do qual um sistema fim ou sistema intermediário pode especificar a largura de banda máxima e tamanho de rajada máxima que ele aceitará em uma conexão. O controle de taxa em uma direção pertencente a um caminho é distinta e pode diferir da taxa na direção oposta.

O mecanismo de controle de taxa pode neutralizar o congestionamento se roteadores com XTP são utilizados. Em outros ambientes ou se bridges são usadas, o mecanismo de controle de taxa do XTP é ineficiente para resolver o problema de um possível congestionamento.

3.1.6. Mecanismos de Controle de Erro

Os principais mecanismos de controle de erro dos protocolos TCP, SCTP e XTP são comparados a seguir.

3.1.6.1. Checksum

Os três protocolos analisados possuem checksum, porém no XTP ele é opcional, podendo ser desabilitado, como pode ser observado na **Tabela 3.14**.

Tabela 3.14 - Mecanismo de Checksum

Checksum	TCP	SCTP	XTP
	X	X	[X]

No TCP o checksum é obrigatório. No SCTP, em versões anteriores a RFC 3286, um algoritmo "Adler 32" de 32 bits (RFC 2960) poderia ser usado. Em versões a partir da RFC 3286, um polinomial CRC-32c é empregado cobrindo toda a PDU. Pacotes com erro no checksum são descartados. O checksum CRC não é computado tão rapidamente quanto o Adler32, mas mostra melhor desempenho na detecção de erros de duplicação de bits e erros de rajada.

No XTP um campo de checksum de 32 bits cobre somente o cabeçalho se o bit NOCHECK estiver setado ou o pacote todo se este estiver em zero. Um pacote recebido que falha na validação do checksum é descartado e nenhuma nova ação é tomada.

3.1.6.2. Detecção de PDU Perdida no Receptor

Como pode ser observado na **Tabela 3.15**, somente o XTP possui o mecanismo de detecção de PDU perdida no receptor, ele é capaz de detectar PDUs perdidas sem tentar retransmiti-las.

Tabela 3.15 - Mecanismo de Detecção de PDU Perdida

Detecção de PDUS Perdidas	TCP	SCTP	XTP
Sem Retransmissão	-	-	[X]

Isto é feito no modo NOERR usando o campo *seq* de 64 bits que representa um número de seqüência do fluxo de dados de saída. Um número de seqüência é associado com todos os bytes de dados naquele fluxo. Uma vez que os pacotes de controle não transportam dados, nenhum número de seqüência é consumido por uma carga de dados de pacotes de controle. As associações sempre começam os fluxos de dados em zero e seguem na seqüência.

3.1.6.3. Confirmação e Retransmissão

Dos três mecanismos de retransmissão citados no capítulo 2, observa-se pela **Tabela 3.16** que a técnica *stop-and-wait* não é implementada em nenhum dos protocolos analisados. O go-back-N está presente em todos, porém no XTP ele é opcional. A retransmissão seletiva também é implementada pelos três protocolos, sendo opcional no TCP e no XTP.

Tabela 3.16 - Mecanismos de Retransmissão

Retransmissão	TCP	SCTP	XTP
stop-and-wait	-	-	-
go-back-N	X	X	[X]
seletiva	[X]	X	[X]

A especificação original do TCP (RFC 0793) define somente confirmação cumulativa. Uma opcional confirmação seletiva foi introduzida na RFC 2018. No comprimento variável, parte dos blocos de dados não contíguos

do cabeçalho TCP podem ser confirmados. Se uma confirmação não é recebida dentro de um intervalo de tempo os dados são retransmitidos. No presente a estratégia de retransmissão é dependente de implementação.

O SCTP tem um comportamento comparável com o TCP estendido com confirmações seletivas: também confirmações cumulativas e seletivas são usadas. A retransmissão é iniciada se uma confirmação não tiver sido recebida em tempo oportuno.

O XTP incorpora confirmação positiva e, quando apropriado, negativa para efeito de retransmissão de pacotes de dados perdidos ou danificados. A retransmissão pode ser go-back-N ou seletiva. Somente se indicado via mensagem de controle que dados existem dados faltantes eles são retransmitidos. O algoritmo de controle de erro pode também ser agressivo: um método de notificação de erro de ação rápida é provido (modo FASTNAK) onde NAKs são enviados imediatamente se um pacote não esperado é recebido.

No XTP o emissor tem que requisitar uma confirmação na forma de um pacote de controle explicitamente. Separando pacotes de dados e de controle permite espaçar pacotes de controle em ambientes com taxa de erro baixa e confirmar um grupo de dados grande com cada pacote de controle.

3.1.6.4. Detecção de Duplicações

Como pode ser observado na **Tabela 3.17**, todos os protocolos analisados possuem detecção de duplicações.

Tabela 3.17 - Mecanismo de Detecção de Duplicações

Detecção de Duplicações	TCP	SCTP	XTP
	X	X	X

O TCP emprega números de sequência de 32 bits que são associados com todos os bytes de dados. O SCTP atribui um número de sequência de transmissão (TSN) de 32 bits para cada fragmento de dados de usuário ou mensagem não fragmentada. O XTP introduz números de sequência de 64 bits, uma vez que nas redes de alta velocidade de hoje números de 32 bits

atachados em cada byte de dados aparentam ser pequenos: os reinícios cíclicos (wrap-around) ocorrendo com frequência são problemáticos.

3.1.6.5. Reordenação de Datagramas

Como pode ser visto na **Tabela 3.18**, tanto o TCP, quanto o SCTP e o XTP possuem o mecanismo de reordenação de datagramas, que utiliza os mesmos números de sequência descritos em mecanismo de detecção de duplicações.

Tabela 3.18 - Mecanismo de Reordenação de Datagramas

Reordenação de Datagramas	TCP	SCTP	XTP
	X	X	X

3.1.6.6. Reordenação de Mensagens

Como se pode observar na **Tabela 3.19**, no SCTP a característica de reordenação de mensagens é opcional, enquanto no XTP é mandatória.

Tabela 3.19 - Mecanismo de Reordenação de Mensagens

Reordenação de Mensagens	TCP	SCTP	XTP
		[X]	X

No SCTP a entrega de mensagens na sequência pode ser desabilitada. Neste modo de operação, ainda que datagramas sejam ordenados dentro de cada mensagem, a ordem da mensagem completa pode mudar.

No XTP a entrega de mensagens fora de sequência não é suportada. Assim, datagramas são não somente ordenados dentro de cada mensagem, mas no fluxo de dados todo.

3.1.7. Mecanismos de Gerenciamento de Conexão

Como se pode observar na **Tabela 3.20**, por serem protocolos orientados a conexão, o TCP, o SCTP e o XTP estabelecem uma conexão entre o nó fonte e o nó destino.

Tabela 3.20 - Mecanismo de Estabelecimento de Conexão

Estabelecimento de Conexão	TCP	SCTP	XTP
	X	X	X

No estabelecimento de conexão confiável do TCP um handshake de três vias clássico é utilizado, como descrito na seção 2.2. O mecanismo utilizado pelo SCTP, descrito na seção 2.3, é o handshake de quatro vias, muito mais robusto e menos vulnerável que o mecanismo utilizado pelo TCP. Já o XTP, conforme descrito na seção 2.4, utiliza o estabelecimento de conexão implícita.

Para a liberação dos fluxos de dados, pode-se observar pela **Tabela 3.21** que todos os protocolos analisados suportam o fechamento gracioso e o abortivo. No TCP e XTP o fechamento dos dois fluxos de dados pode ocorrer independentemente.

Tabela 3.21 - Mecanismo de Liberação de Conexão

Liberação de Conexão	TCP	SCTP	XTP
Graciosa	X	X	X
Abortiva	X	X	X

Quando uma entidade TCP quer fechar um fluxo de dados de saída graciosamente, ele seta a flag "FIN" no cabeçalho do pacote. Um fechamento abortivo é iniciado setando a flag "RST".

O SCTP prevê tanto o fechamento gracioso como o abortivo, mas um estado semi-aberto em que um lado pode continuar enviando dados não existe. Quando qualquer dos terminais realiza um fechamento, a conexão em nó irá parar de aceitar novos dados da camada superior e apenas entregará os dados que já estão na fila.

No XTP, como descrito na seção 2.4, é possível o fechamento abortivo e vários tipos de fechamento gracioso.

3.1.8. Mecanismos de Segurança

Os protocolos TCP e XTP por default não possuem mecanismos de segurança. Já o SCTP possui alguns mecanismos de segurança, como o handshake de quatro vias e a etiqueta de verificação, citados na seção 2.3.

3.1.9. Mecanismo de Multicast

Dos protocolos analisados somente o XTP possui multicast e, é opcional e 1:n, como pode-se observar na **Tabela 3.22**.

Tabela 3.22 - Mecanismos de Multicast

Multicast	TCP	SCTP	XTP
1:n	-	-	[X]
n:1	-	-	-
n:m	-	-	-

O XTP é o único protocolo estudado que pode sob demanda usar seu mecanismo de controle de erro em um ambiente multicast. Evidentemente, existe a restrição de que somente retransmissão go-back-N de pacotes perdidos ou corrompidos pode ser usada, mas que não afeta a funcionalidade básica. A única diferença significativa para o modo normal de operação do XTP é que uma associação multicast é sempre unidirecional.

3.1.10. Mecanismo de Prioridade de Processamento

Como pode ser observado na **Tabela 3.23**, somente o XTP possui o mecanismo de prioridade de processamento e, é opcional. Se o XTP for suportado por nós intermediários, é possível priorizar, mas na prática não está disponível para nós intermediários.

O XTP é o único protocolo analisado que opcionalmente emprega um mecanismo de priorização de processamento de pacotes. O campo "sort" é de 16 bits, este valor em zero indica a prioridade mais baixa.

Tabela 3.23 - Mecanismo de Priorização de Processamento

Prioridade de Processamento	TCP	SCTP	XTP
Emissor e receptor	-	-	[X]
Nós intermediários	-	-	(-)

A implementação XTP deve servir todos os contextos ativos em ordem prioritária. Quando há uma oportunidade para transmitir, o pacote com a maior prioridade é selecionado, consistente com as restrições de controle de taxa e de fluxo de cada contexto. Este procedimento de seleção é repetido em cada oportunidade de transmissão. Contextos não operam em modo ordenação (ou seja, os seus pacotes têm o bit "SORT" em zero), são servidos em ordem FIFO após todos os contextos que estão operando no modo ordenação. As mais sofisticadas políticas de fila podem ser utilizados para implementar o modo ordenação.

O TCP tem apenas uma forma muito simples de priorização chamado modo urgente, descrito na seção 2.2.

3.1.11. Mecanismo de Dados Fora da Banda

Como é mostrado na **Tabela 3.24**, apenas o XTP oferece o mecanismo de envio de dados fora da banda e este é opcional.

Tabela 3.24 - Mecanismo de Dados Fora da banda

Dados fora da banda	TCP	SCTP	XTP
	-	-	[X]

No XTP em pacotes de dados a flag "BTAG" pode ser definida para indicar que os primeiros oito bytes do segmento de dados são usados como campo "BTAG" para utilização especial de aplicações do nível superior. O campo "BTAG" é obscuro para o XTP, o que significa que o XTP transmite o conteúdo do campo, mas não sabe o que tem dentro e nem sabe interpretá-lo.

3.2. Metodologia

A metodologia utilizada para realização do comparativo de desempenho empírico dos protocolos TCP, SCTP e XTP é descrita nesta seção. O ambiente dos testes, os testes realizados, seus resultados e conclusões a cerca dos mesmos são apresentados.

3.2.1. Descrição do Ambiente

O ambiente utilizado para realização dos testes foi simples, contando apenas com duas máquinas e uma rede doméstica de 10/100 Mbps. Além das máquinas um switch foi utilizado, assim como cabos padrão ethernet. A seguir os modelos de equipamento são descritos. Na sequência os sistemas operacionais e módulos adicionais instalados para execução dos testes são apresentados, além aplicativos utilizados.

3.2.1.1. Descrição dos Equipamentos

Os equipamentos utilizados nos testes são:

- M1: Processador Intel Pentium dual-core de 1.73GHz 1M cache;
2G de memória;
Placa de rede Broadcom 10/100.
- M2: Processador Intel Celeron D de 2.53GHz;;
1G de memória;
Placa de rede Via 10/100;
- S: Switch 8 portas.

3.2.1.2. Sistema Operacional

O sistema operacional utilizado para realização dos testes, que serão a seguir apresentados, foi o Linux, distribuição Debian 5.0, com kernel 2.6.26. No entanto realizou-se testes também com a distribuição Fedora 8, OpenSuse 11 e Debian 4.0, respectivamente com kernel 2.6.24, 2.6.27 e 2.6.25.

Para o correto funcionamento dos programas de teste para SCTP, foi necessária a instalação do módulo de kernel LK-SCTP - Linux Kernel SCTP (SOURCEFORGE - SCTP). A versão instalada foi a 1.0.10.

Para o funcionamento dos programas utilizando XTP, tentou-se instalar o módulo de kernel Linux XTP (SOURCEFORGE – XTP), porém não foi obtido sucesso na várias tentativas realizadas. Como este módulo foi lançado em março de 2008, e a sugestão dos autores é de que o kernel 2.6.25 do Linux fosse usado, foi instalado este kernel, porém também sem sucesso.

Entre as tentativas feitas, encaminhou-se e-mail para o autor deste módulo, sem resposta até data de entrega deste trabalho. Tentou-se também, alterar algumas primitivas do módulo para adaptá-lo, obtendo um sucesso parcial, no entanto insuficiente para execução dos testes.

Desta forma, os resultados de testes do protocolo XTP apresentados na sequência, são resultados obtidos em outros trabalhos, devidamente citados no decorrer da explanação dos testes.

3.2.1.3. Aplicativos

Os aplicativos para teste de desempenho foram construídos por Pfützenreuter (2004) na linguagem C e C++ e, permitem variar o tamanho das mensagens transmitidas, o número de fluxos utilizados em SCTP, endereços para associação (*bind*) e o volume total trafegado em *bytes*. Exceto quando explicitamente afirmado o contrário, todos os testes utilizaram os soquetes “estilo TCP”. Os aplicativos utilizados foram:

- `tcp_server`: servidor que utiliza soquetes estilo TCP e permite o uso protocolos TCP e SCTP;

- `tcp_client`: cliente, com as mesmas características do servidor `tcp_server`;
- `tcp_server_lat`: servidor, com as mesmas características do servidor `tcp_server`;
- `tcp_client_lat`: cliente, com as mesmas características do servidor `tcp_server`;

Estes aplicativos serão mais bem apresentados quando os mesmos forem utilizados.

3.2.2. Descrição dos Testes

Os testes realizados são de vazão e latência. Sendo para Pfützenreuter (2004), vazão o volume de dados por unidade de tempo. O teste de vazão consiste de transmitir um volume fixo de bytes (nos testes o volume utilizado é de 100 MBytes) em ambas as direções. E latência, o tempo médio decorrido entre a requisição do cliente e a resposta do servidor.

Assim como Pfützenreuter (2004), foram realizados testes utilizando o protocolo "TCPM" que é um protocolo de aplicação que faz uma separação de mensagens simples.

De acordo com Pfützenreuter (2004), o TCPM foi criado principalmente para assegurar uma maior igualdade nas comparações entre TCP e SCTP, já que SCTP sempre separa mensagens. Além disso, nos testes de latência, é necessário que o servidor tenha meios de determinar onde termina a mensagem do cliente, para que possa respondê-la.

Para os testes de vazão os aplicativos `tcp_server` e `tcp_client` foram utilizados, simulando respectivamente, o servidor e o cliente. Para aproveitar os dados coletados por Viegas (2008) sobre o protocolo XTP, os testes realizados utilizaram mensagens de 5 MB, 278 MB, 1456 MB e 23256 MB. Além disso, realizou-se teste variando o número de fluxos em 10, 100 e 1000.

Os aplicativos não tentam sincronizar de nenhuma forma os dois fluxos, sendo possível que uma direção termine antes da outra, prejudicando o resultado final do teste. Isto é propositado, de acordo com Pfützenreuter

(2004), visto que um bom protocolo de transporte deve permitir uma comunicação full duplex equilibrada, e o teste de vazão testa (indiretamente) esse equilíbrio.

Já para os testes de latência, os aplicativos utilizados foram *tcp_server_lat* e *tcp_client_lat*, respectivamente servidor e cliente, que possuem as mesmas características dos aplicativos usados nos testes de vazão.

De acordo com Viegas (2008), para cada mensagem enviada pelo cliente, exatamente uma é retornada pelo servidor, simulando assim um sistema cliente/servidor típico.

O lado cliente é responsável por computar a latência total entre a remessa da pergunta e o retorno da resposta. O tempo fora desse intervalo não é computado. Cada transação é cronometrada utilizando-se a chamada POSIX *gettimeofday()* e o valor retornado ao final é a média das latências.

3.2.3. Resultado dos Testes

Os testes empíricos realizados com os protocolos TCP, SCTP e XTP são apresentados a seguir. Como mencionado, os testes realizados são de vazão e latência, em interface loopback e rede de 100 Mbps. Os testes em loopback foram realizados na máquina M2, os em rede de 100 Mbps foram realizados em M1 e M2, com uso do switch S e da rede cabeada.

3.2.3.1. Testes em Interface Loopback

Os testes realizados em interface loopback foram de vazão e de latência. Também se verificou a latência com a utilização de vários fluxos de dados.

3.2.3.1.1. Teste de Vazão

No teste de vazão em loopback verifica-se que o desempenho do XTP praticamente supera o TCP. O SCTP apresenta um desempenho muito ruim,

segundo Pfützenreuter (2004) isso se deve ao checksum utilizado (CRC-32c) e também a troca de contexto no destino devido à separação de mensagem.

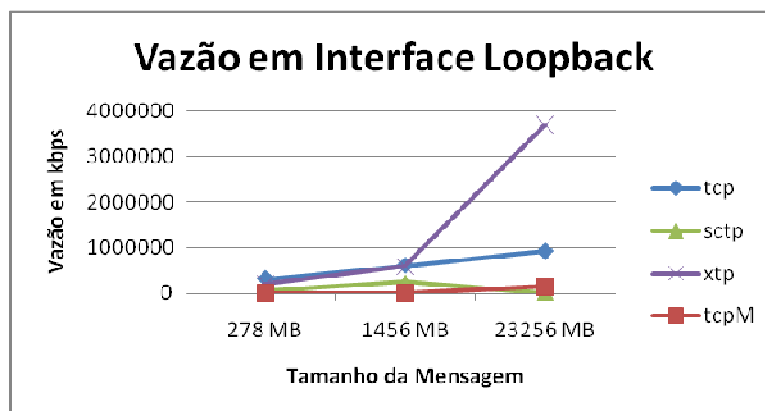


Figura 3.19 - Vazão em Loopback

Os dados utilizados para gerar o gráfico da **Figura 3.19**, são mostrados na **Tabela 3.25**. Os dados do XTP foram produzidos por Viegas (2008).

Tabela 3.25 - Vazão em Loopback

Protocolo/Tamanho Mensagem	278 MB	1456 MB	23256 MB
tcp	310784	605712	912144
sctp	65568	238096	9928
xtp	189895	578785	3696548
tcpM	2192	10896	161528

3.2.3.1.2. Teste de Latência

Os testes de latência em interface loopback foram realizados utilizando o mesmo tamanho de mensagem utilizado por Viegas (2008), para que os resultados alcançados por ele para o XTP possam ser utilizados nos comparativos.

No teste realizado com somente um fluxo, variando o tamanho da mensagem, verifica-se um bom desempenho do XTP, o SCTP também apresenta melhora, em relação ao teste de vazão.

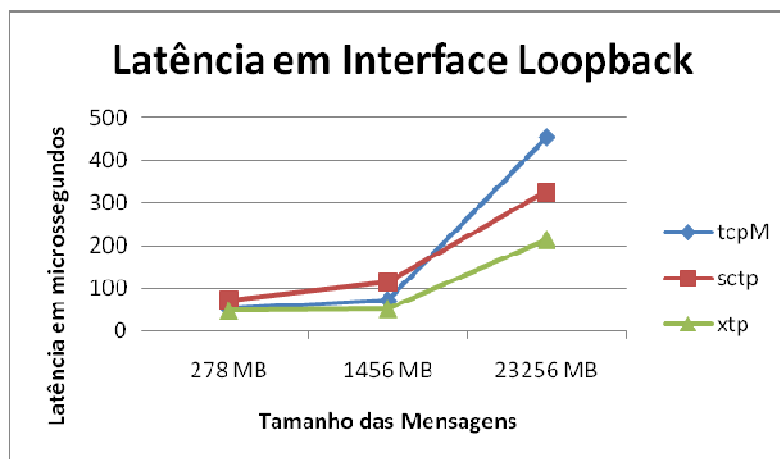


Figura 3.20 - Latência em Loopback

A Figura 3.20 mostra a latência dos protocolos analisados, na interface loopback. Os dados são apresentados na Tabela 3.26.

Tabela 3.26 - Latência em Loopback

Protocolo/Tamanho Mensagem	278 MB	1456 MB	23256 MB
tcpM	51	71	454
sctp	72	115	327
xtp	48	52	215

Os testes de latência com múltiplos fluxos demonstram o que já havia sido apresentado por Pfützenreuter (2004), o desempenho do SCTP fica muito abaixo do TCP. Os dados do XTP não aparecem nestes gráficos porque não foram realizados em nenhum trabalho analisado.

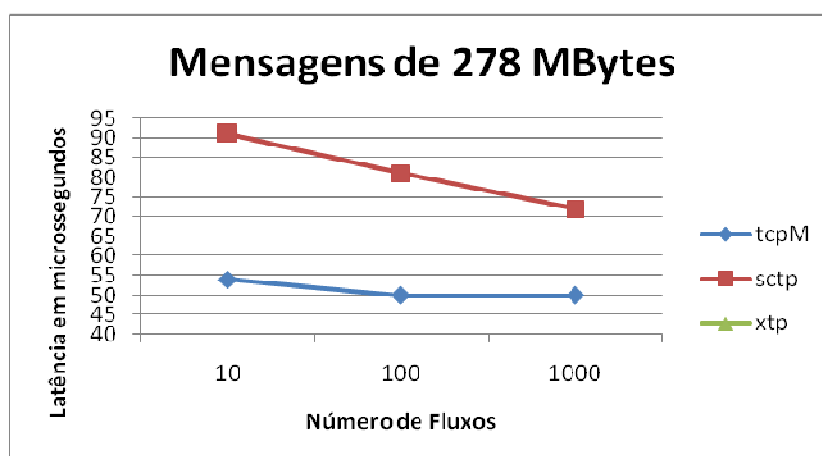


Figura 3.21 - Latência Loopback Multi 278MB

Pelo gráfico da **Figura 3.21**, pode-se perceber a diferença da latência entre o TCP e o SCTP. Enquanto a latência do TCP gira em torno de 50 μ s, a do SCTP está em torno de 80 μ s, cerca de 30% a mais. Os valores coletados e usados para gerar o gráfico são apresentados na **Tabela 27**.

Tabela 3.27 - Latência Loopback Multi 278MB

Mensagem 278MB	Número de Fluxos		
	10	100	1000
Protocolos			
tcpM	54	50	50
sctp	91	81	72
xtp	-	-	-

A análise do gráfico da **Figura 3.22** também revela a diferença significativa entre a latência apresentada pelo SCTP e o TCP. Os dados estão disponíveis na **Tabela 3.28**.

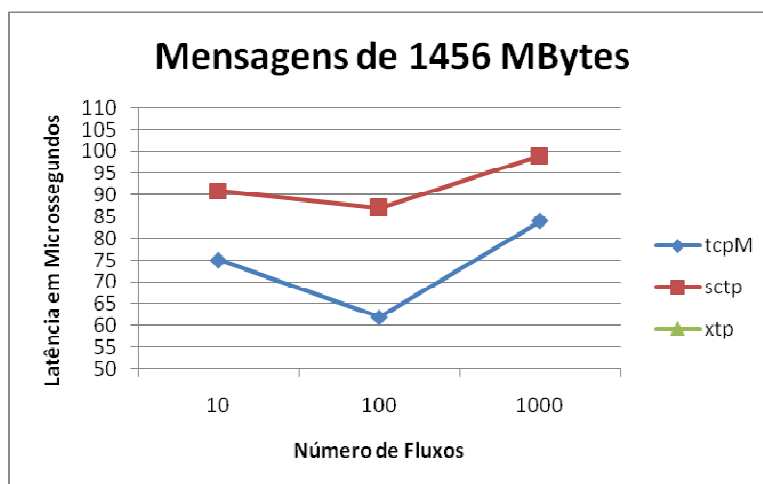


Figura 3.22 - Latência Loopback Multi 1456MB

A menor latência apresentada pelo SCTP, quando 100 fluxos foram utilizados, ainda é maior que a maior latência apresentada pelo TCP.

Tabela 3.28 - Latência Loopback Multi 1456MB

Mensagem 1456MB	Número de Fluxos		
Protocolos	10	100	1000
tcpM	75	62	84
sctp	91	87	99
xtp	-	-	-

Com mensagens de 23256 MB o resultado não foi diferente, a latência do TCP foi muito mais baixa que a do SCTP, como pode ser observado na **Figura 3.23**.

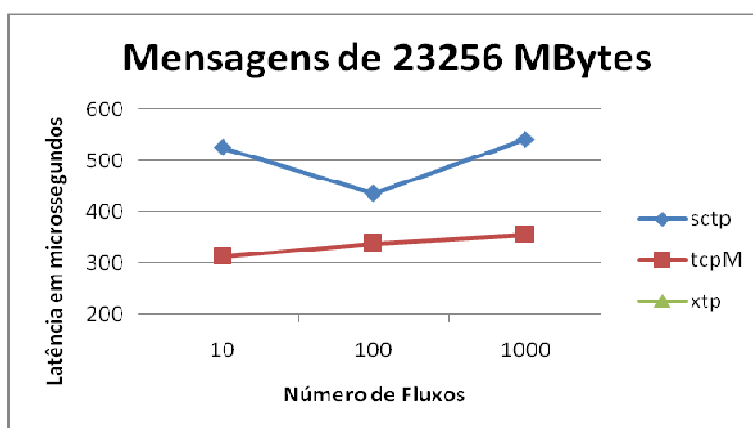


Figura 3.23 - Latência Loopback Multi 23256MB

Os dados usados para gerar o gráfico da **Figura 3.23** são mostrados na **Tabela 3.29**.

Tabela 3.29 - Latência Loopback Multi 23256MB

Mensagem 23256MB	Número de Fluxos		
Protocolos	10	100	1000
sctp	525	436	541
tcpM	313	338	355
xtp	-	-	-

3.2.3.2. Testes em Rede 10/100 Mbps

Os testes realizados em rede de 100 Mbps seguem os mesmos princípios que os realizados em interface loopback. Também foram realizados testes de vazão e desempenho e, foi adotado o mesmo critério para o número de mensagens que Viegas (2008), para que os dados dos testes do protocolo XTP deste, pudessem ser utilizados.

3.2.3.2.1. Teste de Vazão

Pelo gráfico da **Figura 3.24** verifica-se uma melhora do desempenho do SCTP. O XTP tem quase o mesmo desempenho que o SCTP, e os dois juntos ficam muito abaixo ainda do TCP.

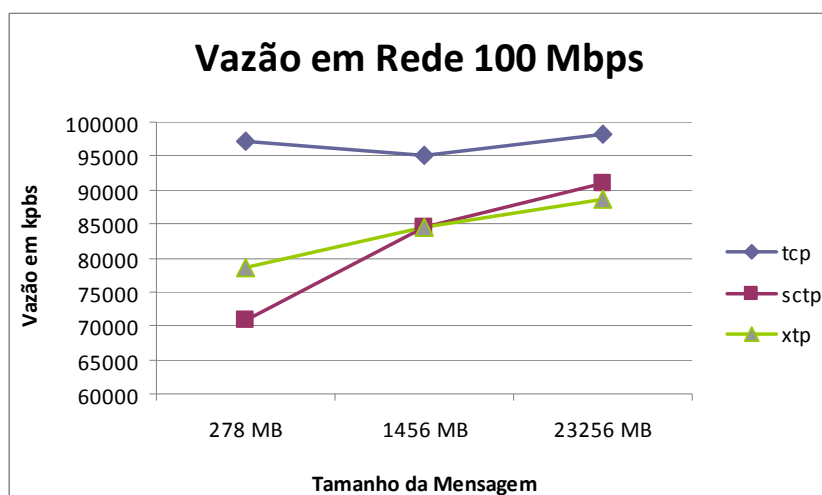


Figura 3.24 - Vazão em Rede 100 Mbps

Os dados do gráfico anterior estão na **Tabela 3.30**.

Tabela 3.30 - Vazão em Rede 100 Mbps

Protocolo/Tamanho Mensagem	278 MB	1456 MB	23256 MB
tcp	97240	95124	98276
sctp	70937	84553	90845
xtp	78585	84585	88585

3.2.3.2.2. Teste de Latência

Os testes de latência foram inicialmente realizados com um fluxo único, com volume de dados default (100 MBytes) e variação no número de mensagens. Após, realizou-se testes com múltiplos fluxos e seguiu-se novamente a divisão de número de mensagens utilizada por Viegas (2008).

No teste com fluxo único, mostrado na **Figura 3.25**, percebe-se um comportamento bem similar em todos os protocolos, mas ainda identifica-se claramente a menor latência do TCP. O SCTP e XTP possuem praticamente o mesmo desempenho.

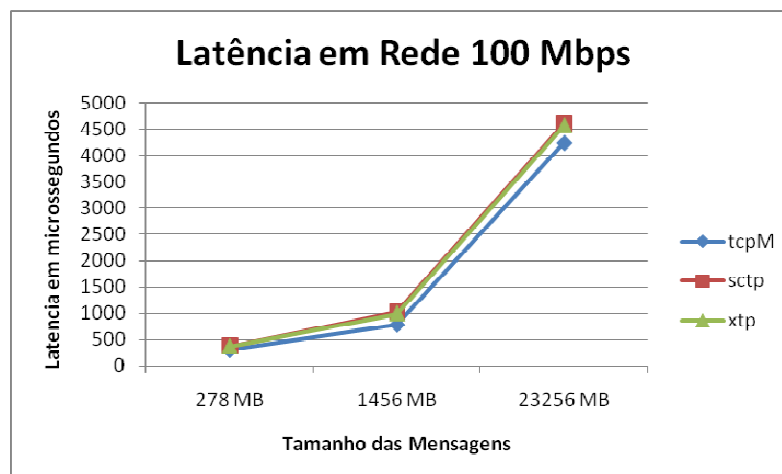


Figura 3.25 - Latência em Rede 100 Mbps

Os dados usados para gerar o gráfico da **Figura 3.25** estão disponíveis na **Tabela 3.31**.

Tabela 3.31 - Latência em Rede 100 Mbps

Protocolo/Tamanho Mensagem	278 MB	1456 MB	23256 MB
tcpM	307	789	4248
sctp	369	1020	4621
xtp	371	997	4582

Para múltiplos fluxos, gerou-se novamente um gráfico para cada número de mensagens. Desta forma, pode-se observar que para mensagens de 278 MB, a latência do TCP é inferior aos outros dois protocolos, girando em

torno de 350 μ s, enquanto no SCTP está em torno de 389 μ s e no XTP 370 μ s, como se pode observar na **Figura 3.26**.

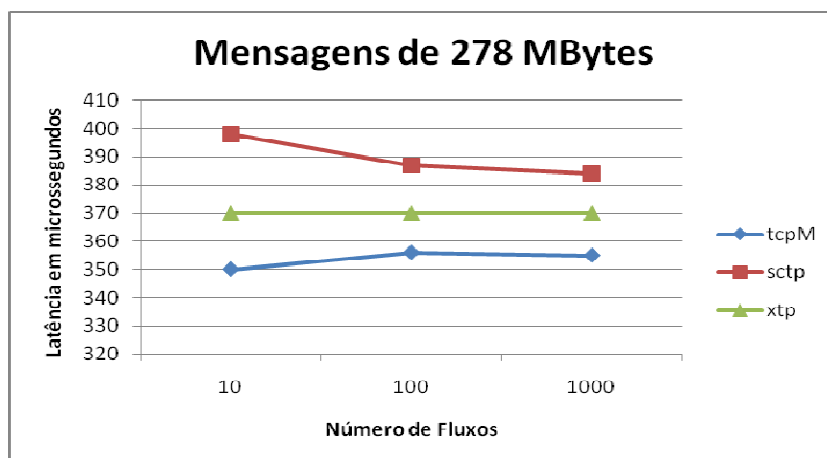


Figura 3.26 - Latência em Rede 100 Mbps Multi 278MB

Os dados do gráfico anterior são apresentados na **Tabela 3.32**.

Tabela 3.32 - Latência em rede 100 Mbps Multi 278MB

Mensagem 278MB	Número de Fluxos		
	10	100	1000
Protocolos			
tcpM	350	356	355
sctp	398	387	384
xtp	370	370	370

Nas mensagens de 1456 MB, o já esperado melhor desempenho do TCP, novamente se confirma, mas entre o SCTP e o XTP observa-se a proximidade dos resultados obtidos, como pode ser verificado na Figura 3.27.

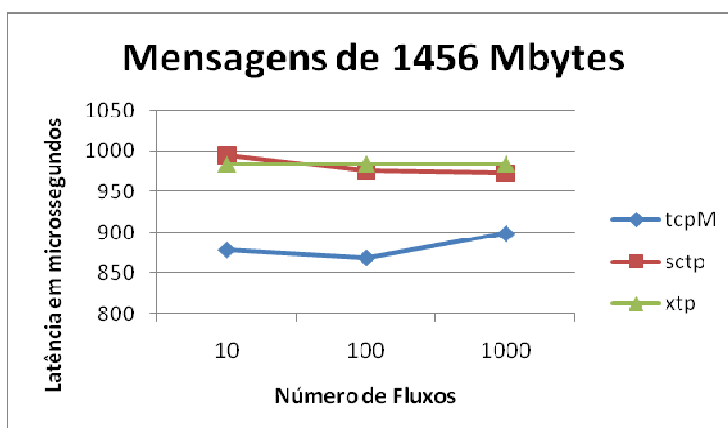


Figura 3.27 - Latência em Rede 100 Mbps Multi 1456MB

Pela **Tabela 3.33** observa-se a similaridade entre os resultados do SCTP e XTP.

Tabela 3.33 - Latência em Rede 100 Mbps Multi 1456MB

Mensagem 1456MB	Número de Fluxos		
Protocolos	10	100	1000
tcpM	879	869	899
sctp	995	976	973
xtp	984	984	984

Já para mensagens de 23256 MB observa-se um péssimo desempenho do SCTP face ao TCP e ao XTP, sendo que o aumento no número de fluxos em nada ajuda a melhorar a performance do SCTP, como pode ser observado na **Figura 3.28**.

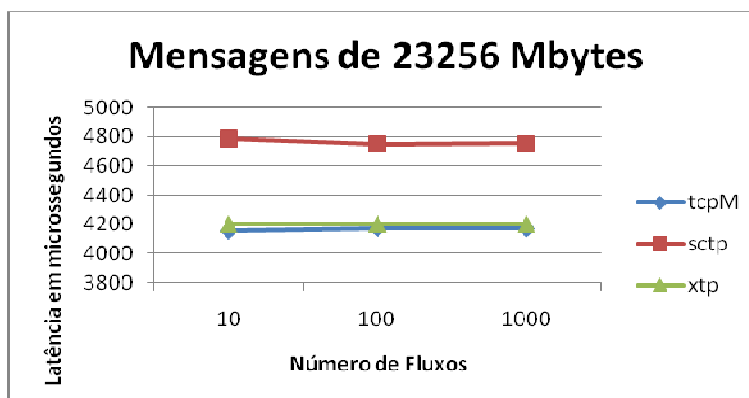


Figura 3.28 - Latência em Rede 100 Mbps Multi 23256MB

Os dados usados para gerar o gráfico anterior estão disponíveis na **Tabela 3.34**.

Tabela 3.34 - Latência em Rede 100 Mbps Multi 23256MB

Mensagem 23256MB	Número de Fluxos		
Protocolos	10	100	1000
tcpM	4156	4172	4169
sctp	4789	4750	4755
xtp	4200	4200	4200

3.2.4. Conclusão dos Testes

Em todos os testes gerados identificou-se claramente que o desempenho do TCP foi melhor que o do SCTP e XTP, em loopback ou em rede de 100 Mbps.

Independente do tamanho da mensagem, a latência do SCTP é sempre maior que apresentada pelo TCP, a do XTP se aproxima mais do TCP em algumas ocasiões.

O desempenho inferior do SCTP nos testes de latência, segundo Viegas (2008), refere-se ao grande número de cópias de memórias executadas pela implementação usada, sabe-se que uma forma “ideal” seria que protocolos de transporte realizassem uma ou até zero cópias de memória, mas sabe-se através do estudo em cima do protocolo SCTP que devido as suas características isso não é possível.

Com a realização dos testes identificou-se a necessidade de fazer testes com aplicações que possam aproveitar o que de melhor é oferecido nestes protocolos, como a característica de múltiplos caminhos e confiabilidade parcial no SCTP e de multicast e parametrização no XTP. O ideal seria realizar testes em ambiente de alto desempenho e com aplicações reais.

4. Conclusão e Trabalhos Futuros

De forma geral, os objetivos definidos para este trabalho foram cumpridos. Foi realizado um extenso estudo bibliográfico sobre os protocolos TCP, SCTP e XTP, levantando suas principais características, mecanismos e modos de operação, de maneira a apresentar subsídios para que os protocolos de alto desempenho analisados possam ser identificados como uma alternativa ao tradicional TCP, quando este não atender as necessidades específicas de alguma aplicação ou rede.

Posteriormente, foi realizada uma comparação entre os principais mecanismos dos protocolos de transporte analisados, identificando o que cada um pode oferecer, além de mostrar se o recurso em questão pode ser desabilitado, para um ganho de desempenho ou para atender alguma necessidade especial de alguma aplicação.

Pelos testes empíricos realizados identificou-se que o desempenho do TCP ainda supera o do SCTP e XTP em um cenário comum, com aplicações que não exigem muito mais de um protocolo de transporte do que as já muito utilizadas e testadas característica do TCP.

Assim sendo, deve-se levar em conta na hora de escolher o protocolo de transporte a se utilizar, não os resultados dos testes empíricos, aqui demonstrados, mas as características da aplicação e da rede que será utilizada e as funcionalidades que o protocolo deve apresentar.

Os resultados apresentam fragilidade e acredita-se que não foram à melhor forma de testar desempenho para protocolos de alto desempenho, porque não demonstram a real capacidade dos protocolos analisados.

Além do que, não foi possível testar efetivamente o protocolo XTP, que apresenta muitas características positivas, mas existem poucas ou quase nenhuma implementação do mesmo disponível e, o principal fórum que discutia este protocolo já não está mais disponível.

Observa-se também, que o protocolo SCTP já está sendo utilizado por diversas aplicações e, pelo crescente número de material disponível na Internet

sobre o mesmo, a tendência é que ele seja melhor aceito e aplicado do que, por exemplo, o XTP.

4.1. Trabalhos Futuros

Como trabalho futuro seria interessante estender os estudos apresentados neste trabalho, considerando-se um maior número de aplicações em um ambiente de produção dos protocolos TCP, SCTP e XTP, buscando mostrar as principais características dos mesmos, de forma a dar subsídios para que a escolha pelo protocolo de transporte beneficie a aplicação e otimize o uso da rede.

Assim, os testes devem procurar ressaltar as principais funcionalidades de cada protocolo, como o mecanismo de confiabilidade parcial do SCTP, ou o SCTP móvel, que aparenta ser bastante interessante, apesar de ainda estar sendo desenvolvido e não estar formalizado na RFC do SCTP. No XTP, o uso do multicast e a parametrização disponíveis devem ser melhor ressaltadas e testadas.

Referências Bibliográficas

ALECRIM, Emerson. **Ataques DoS (Denial of Service) e DDoS (Distributed DoS)**. InfoWester. 09 outubro 2004. Disponível em <<http://www.infowester.com/col091004.php>>. Acesso em: 12 abril 2009.

ATWOOD, J. William et al. **Reliable Multicasting in the Xpress Transport Protocol**. Minneapolis - Eua: Ieee, 1996. Disponível em: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=558148>. Acesso em: 02 maio 2009.

BELLOVIN, Steven M. **Security Problems in the TCP/IP Protocol Suite**. New Jersey: AT&T Laboratory, Abril 1989. Disponível em <<http://www.citi.umich.edu/u/provos/security/ipext.ps.gz>>. Acesso em: 10 abril 2009.

_____. **A Look at "Security Problems in the TCP/IP Protocol Suite"**. New Jersey: AT&T Laboratory, Dezembro 2004. Disponível em <<http://www.cs.columbia.edu/~smb/papers/acsac-ipext.pdf>>. Acesso em: 10 abril 2009.

C²I² Systems. **XPRESS Transport Protocol**. South Africa, 2002. Disponível em: <http://www.cci.co.za/products/pdf/protocols/xtp_a4.pdf>. Acesso em: 02 maio 2009.

CAMPANA FILHO, José Carlos. **Proposta de um Protocolo Leva para Clusters de Computadores**. 2002. 66 f. Monografia (Bacharel) – Curso de Ciência da Computação, UFES, Vitória, 2002. Disponível em: <http://sergio.inf.ufes.br/files/Jose_Campana.projeto_final.pdf>. Acesso em: 28 abril 2009.

CARO JUNIOR, Armando L. et al. **SCTP: A Proposed Standard for Robust Internet Data Transport**. University Of Delaware, USA: IEEE, 2003. 8 p. Disponível em: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=1244536&isnumber=27879>. Acesso em: 02 maio 2009.

CASAD, Joe; WILLSEY, Bob. **Aprenda TCP/IP em 24 horas**. Editora Campus, 1999.

CCIIS - Communications Computer Intelligence Integration Systems. **Xpress Transport Protocol**. África do Sul. Disponível em: <http://www.cci.co.za/products/pdf/protocols/xtp_a4.pdf>. Acesso em: 22 nov. 2008.

CECHIN, Sérgio L. **Protocolo SCTP - Stream Control Transmission Protocol**. Rio Grande do Sul, 2008. Disponível em: <<http://www.inf.ufrgs.br/~cechin/Net/sctp/sctp.html>> Acesso em: 23 de nov. 2008.

CHAMBERS, Chris; DOLSKE, Justin; IYER, Jayaraman. **TCP/IP Security**. Ohio: Department of Computer and Information Science Ohio State University, [200x]. Disponível em <http://www.linuxsecurity.com/resource_files/documentation/tcpip-security.html>. Acesso em 09 de abril de 2009.

COMER, D. E. **Interligação em redes com TCP/IP - Volume I**. Editora Campus, 1998.

COSTA, Daniel G. **SCTP – Uma alternativa aos tradicionais protocolos de transporte da Internet**. Editora Ciencia Moderna, 2005.

_____. **Uma Arquitetura Baseada em SCTP e SIP para Suporte a Aplicações VoIP Móveis e a Especificação Formal do seu Módulo de Controle**. 2006. 79 f. Dissertação (Mestrado) - Curso de Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal, 2006. Disponível em: <<ftp://ftp.ppgeec.ufrn.br/Mestrado/M167.pdf>>. Acesso em: 02 maio 2009.

DAVIE, B.; PETERSON, L. **Computer Networks: A Systems Approach**. Morgan Kaufmann Publishers, 1996.

DÍAZ, Antonio F. et al. **The Lightweight Protocol CLIC: Performance of an MPI implementation on CLIC**. Granada - Spain: University Of Granada, 2001. 8 p. Disponível em: <<http://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=960005&isnumber=20737>>. Acesso em: 28 abril 2009.

_____. **The Lightweight Protocol CLIC on Gigabit Ethernet**. Granada - Spain: University Of Granada, 2003. 8 p. Disponível em: <http://atc.ugr.es/doctorado/apuntes/clic_cac_2003.pdf>. Acesso em: 28 abr. 2009.

FARREL, Adrian. **A internet e seus protocolos – Uma análise comparativa**. Editora Campus, 2005.

FILEPPETTI, Marco A. **CCNA 4.1 – Guia Completo de Estudo**. Editora Visual Books, 2008.

FRÖHLICH, Antônio Augusto Medeiros et al. **Um Framework Meta-Programado Para a Implementação de Protocolos Leves de Comunicação**. Santiago - Chile: Proceedings of the 32th Latin-American Conference on Informatics. UFSC, 2006. Disponível em: <<http://www.lisha.ufsc.br/~guto/publications/clei2006c.pdf>>. Acesso em: 28 abr. 2009.

GUHA, B; MUKHERJEE, B. **Network security via reverse engineering of TCP code: Vulnerability analysis and proposed solutions**. California: IEEE Network, agosto 1997. Disponível em <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=598458>. Acesso em 10 abril 2009.

HENRICI, Dirk. **A Universal Scheme for the Classification of Network Services**. 2002. 121 f. Tese (Doutorado) - Universität Kaiserslautern, Kaiserslautern, 2002. Disponível em: <<http://dSPACE.icsy.de:12000/dSPACE/bitstream/123456789/33/1/DPArchiv.0087.pdf>>. Acesso em: 04 maio 2009.

JACINTO, Daniele Santini. **Protocolos de transporte para suporte à comunicação multimídia**. São Carlos: Universidade Federal de São Carlos, 2005. 29 p. Disponível em: <www.dc.ufscar.br/~zorzo/download/CCO123/Cap6.pdf>. Acesso em: 02 maio 2009.

JAVVIN TECHNOLOGIES. **Network Protocols Handbook**. 2. ed. Saratoga - Usa: Javvin Technologies, 2005. 359 p. Disponível em: <<http://rapidlibrary.com/>>. Acesso em: 03 maio 2009.

JIN, Hai et al. **Lightweight Real-Time Network Communication Protocol for Commodity Cluster Systems**. China: Springer Berlin / Heidelberg, 2005. 10 p. Disponível em: <www.springerlink.com/index/f3xu610640356275.pdf>. Acesso em: 02 maio 2009.

JONCHERAY, Laurent. **Simple Active Attack Against TCP**. Utah: Fifth USENIX UNIX Security Symposium, junho 1995. Disponível em <https://db.usenix.org/publications/library/proceedings/security95/full_papers/joncheray.ps>. Acesso em: 10 abril 2009.

LI, Jia-ru; HA, Sungwon; BHARGHAVAN, Vaduvur. **HPF: A Transport Protocol For Supporting Heterogeneous Packet Flows in the Internet**. Illinois - Usa: University Of Illinois At Urbana-champaign, 1999. 8 p. Disponível em: <ieeexplore.ieee.org/iel4/6063/16207/00751388.pdf?arnumber=751388>. Acesso em: 02 maio 2009.

MORRIS, Robert T. **A Weakness in the 4.2BSD UNIX TCP/IP Software**. New Jersey: AT&T Bell Laboratory, 1985. Disponível em <<http://pdos.csail.mit.edu/~rtm/papers/117.pdf>>. Acesso em: 10 abril 2009.

NUNES, Marcelo D.; DUARTE, Otto C. M. B. **Análise de Mecanismos Para Protocolos de Alto Desempenho**. VI Simpósio Brasileiro de Arquitetura de Computadores e Processamento de Alto Desempenho. Caxambu, agosto 1994. Disponível em: <<http://www.gta.ufrj.br/ftp/gta/TechReports/NuDu94b.ps.gz>>. Acesso em: 26 abril 2009.

ORTIZ, Cleber Machado; GRANVILLE, Lisandro Zambenedetti; TAROUÇO, Liane Margarida Rockenbach. **Aplicação que usa Protocolo de Perfil Leve para Transferência de Arquivos**. Blumenau: Seminário de Computação (XII SEMINCO). 14 p.. UFRGS, 2003. Disponível em: <<http://www.inf.furb.br/seminco/2003/artigos/106-vf.pdf>>. Acesso em: 28 abril 2009.

PAULO, Elisabete et al. **Reserva de Recursos e Qualidade de Serviço em Redes de Computadores de Débito Elevado**. Fundação para a Computação Científica Nacional. Lisboa, março de 1997. Disponível em: <<http://eden.dei.uc.pt/~eduardo/Publications.htm>>. Acesso em: 26 abril 2009.

PFÜTZENREUTER, Elvís. **Aplicabilidade e desempenho do protocolo de transporte SCTP**. 2004. 122 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Ufsc, Florianópolis, 2004. Disponível em: <<http://epx.com.br/mestrado/>>. Acesso em: 13 maio 2009.

PARZIALE, Lydia et al. **TCP/IP Tutorial and Technical Overview**. USA: Redbooks IBM, 2006. 1004 p. Disponível em: <www.redbooks.ibm.com/redbooks/pdfs/gg243376.pdf>. Acesso em: 03 maio 2009.

Quintero, Nestor Felipe Maya. **Middleware para os Protocolos SCTP e HIP, PUC**. Rio de Janeiro, 2007.

RFC 0793. **Transmission Control Protocol**. 1981. Disponível em: <<http://www.faqs.org/rfcs/rfc793.html>>. Acesso em: 14 maio 2009.

RFC 0813. **Window and Acknowledgement Strategy in TCP**. 1982. Disponível em: <<http://www.faqs.org/rfcs/rfc813.html>>. Acesso em: 14 maio 2009.

RFC 0879. **TCP maximum segment size and related topics**. 1983. Disponível em: <<http://www.faqs.org/rfcs/rfc879.html>>. Acesso em: 14 maio 2009.

RFC 0896. **Congestion control in IP/TCP internetworks**. 1984. Disponível em: <<http://www.faqs.org/rfcs/rfc896.html>>. Acesso em: 14 maio 2009.

RFC 1106. **TCP big window and NAK options**. 1989. Disponível em: <<http://www.faqs.org/rfcs/rfc1106.html>>. Acesso em: 14 maio 2009.

RFC 1146. **TCP alternate checksum options**. 1990. Disponível em: <<http://www.faqs.org/rfcs/rfc1146.html>>. Acesso em: 14 maio 2009.

RFC 2581. **TCP Congestion Control**. 1999. Disponível em: <<http://www.faqs.org/rfcs/rfc2581.html>>. Acesso em: 14 maio 2009.

RFC 2960. **Stream Control Transmission Protocol**. 2000. Disponível em: <<http://www.faqs.org/rfcs/rfc2960.html>>. Acesso em: 14 maio 2009.

RFC 2988. **Computing TCP's Retransmission Timer**. 2000. Disponível em: <<http://www.faqs.org/rfcs/rfc2988.html>>. Acesso em: 14 maio 2009.

RFC 3554. **On the Use of Stream Control Transmission Protocol (SCTP) with IPsec**. 2003. Disponível em: <<http://www.faqs.org/rfcs/rfc3554.html>>. Acesso em: 14 maio 2009.

RFC 3758. **Stream Control Transmission Protocol (SCTP) Partial Reliability Extension**. 2004. Disponível em: <<http://www.faqs.org/rfcs/rfc3758.html>>. Acesso em: 14 maio 2009.

RIBEIRO, Cesar Henrique Pereira. **Novos Protocolos de Streaming MMTP e SCTP**. Universidade Federal Fluminense, 2006. Disponível em: <www.midiacom.uff.br/~debora/fsmm/trab-2006-2/apres_streaming.pdf>. Acesso em: 28 abril 2009.

RIBEIRO, Nuno M. **Estudo Comparativo de protocolos de nível de transporte em redes rápidas: XTP versus TCP e TP4**. PRODEP, março 1993. Disponível em: <<http://cerem.ufp.pt/~nrbeiro/publicacoes/ntp.pdf>>. Acesso em: 26 abril 2009.

SANDERS, Robert M.; WEAVER, Alfred C. **The Xpress Transfer Protocol (XTP) - Tutorial**. Virginia, 2007. Disponível em: <<http://www.cs.virginia.edu/papers/p67-sanders.pdf>>

SANTOS, Éverton Machado Dos; RIBEIRO, Glaucia da Silva; MINICZ, Márcio de Freitas. **SCTP: Princípios e Funcionamento**. RTI: *Redes, Telecom e Instalações*, São Paulo, n. 100, ano IX, p.72-81, setembro 2008. Mensal.

SCTP Programmer's Guide. Hewlett-Packard Development Company, setembro 2007. Disponível em <http://docs.hp.com/en/5992-0620/index.html>. Acesso em: 14 maio 2009.

STEVENS, W. Richard. **TCP/IP ILLUSTRATED: The Protocols**. Addison Wesley, 1994. 600 p.

Stevens, W. Richard; Wright, Gary R. **TCP/IP Illustrated: TCP for Transactions, Http, Nntp, and the Unix Domain Protocols**. Addison-Wesley Reading, Massachusetts, 1996.

SOURCEFORGE - SCTP. **Linux SCTP**. Módulo de Kernel Linux para SCTP. Disponível em: <<http://lksctp.sourceforge.net/>>. Acesso em: 13 maio 2009.

SOURCEFORGE - XTP. **Linux XTP**. Módulo de Kernel Linux para XTP. Disponível em: <<http://sourceforge.net/projects/linux-xtp/>>. Acesso em: 13 maio 2009.

STRAYER, W. Timothy (editor chefe). **Xpress Transport Protocol Specification: Revision 4.0**. XTP-95-20. Santa Barbara: XTP Forum, 01 mar 1995. Disponível em <<http://www.ir.bbn.com/~strayer/pubs.html>>. Acesso em 08 abr 2009.

SOUZA JÚNIOR, Jaci Alves de. **Implementação de um protocolo de redes leve para o “cluster” de computadores do Departamento de Informática**. Vitória: UFES, 2006. 20 p. Disponível em: <http://sergio.inf.ufes.br/files/Jaci_Souza.relatorio_pibic.pdf>. Acesso em: 28 abr. 2009.

STEWART, Randall; AMER, Paul D. **Why is SCTP needed given TCP and UDP are widely available?** Virginia - Usa: Internet Society, 2004. 5 p. Disponível em: <<http://www.isoc.org/briefings/017/>>. Acesso em: 02 maio 2009.

TANENBAUM, Andrew S. **Redes de Computadores**. 4ª Edição. Editora Campus, 2003.

TANTAWY, Ahmed N. **High Performance Networks – Technology and Protocols**. Klumer Academic Publishers, 1994.

TKOTZ, Viktoria. **Ataques na camada 4 (TCP e UDP)**. Aldeia Numa Boa, maio 2007. Disponível em: <<http://www.numaboa.com/informatica/seguranca/165-exploits/718-camada4>>. Acesso em: 26 abril 2009.

TRCEK, Denis; KOVAC, Damjan. **Formal apparatus for measurement of lightweight protocols**. Slovenia: Elsevier, 2008. Disponível em: <<http://portal.acm.org/citation.cfm?id=1461075>>. Acesso em: 02 maio 2009.

VIEGAS, Diogo Ruviano. **Estudo Experimental dos Protocolos TCP, SCTP E XTP**. 2008. 95 f. Dissertação (Mestrado) - Curso de Ciência da Computação, UFSC, Florianópolis, 2008.

ZHAO, Feng; SU, Jinshu; CHENG, Xiaomei. **OpenRouter: A TCP-Based Lightweight Protocol for Control Plane and Forwarding Plane Communication**. Hunan - China: Springer Berlin / Heidelberg, 2005. 10 p. Disponível em: <www.springerlink.com/index/50j8p0ethkyrgy4h.pdf>. Acesso em: 02 maio 2009.

Apêndice

Estudo Comparativo entre Protocolos de Transporte TCP, SCTP e XTP

Solange Sonaglio¹

¹Centro Tecnológico - Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-970 – Florianópolis – SC – Brasil

solange@inf.ufsc.br

Abstract. *In an era where the networks have more and more flow and capacity of transmission, where the equipments of tip and of interconnection improve in terms of hardware and software to each day, the transport protocols also specify if to be adapted and to gain performance. The protocol TCP (Transmission Control Protocol) was developed in an era where the networks were of low speed and performance, presenting many limitations, principally in security. So, to provide for needs specific, other transport protocols appeared, like the SCTP (Stream Control Protocol) and the XTP (Xpress Transport Protocol), which are protocols projected for nets of high performance, with low rate of error and great width of band.*

Resumo. *Em uma era onde as redes possuem cada vez mais vazão e capacidade de transmissão, onde os equipamentos de ponta e de interconexão melhoram em termos de hardware e software a cada dia, os protocolos de transporte também precisam se adaptar e ganhar desempenho. O protocolo TCP (Transmission Control Protocol) foi desenvolvido em uma era onde as redes eram de baixa velocidade e desempenho, apresentando muitas limitações, principalmente em segurança. Assim, para suprir necessidades específicas, outros protocolos de transporte surgiram como o SCTP (Stream Control Protocol) e o XTP (Xpress Transport Protocol), que são protocolos projetados para redes de alto desempenho, com baixa taxa de erro e grande largura de banda.*

1. Introdução

O tráfego de serviços IP tem crescido de uma forma exponencial nos últimos tempos, aumentando a necessidade de melhorar e otimizar a transmissão dos dados nas redes IP. Neste cenário, os protocolos de transporte em uso hoje, como o TCP, apresentam limitações quanto ao desempenho e segurança.

Outros protocolos, voltados para prover às redes um alto desempenho, têm sido propostos. Exemplos são os protocolos SCTP (Stream Control Transmission Protocol) e o XTP (Xpress Transport Protocol) que têm o perfil de interesse no desenvolvimento deste trabalho, que visa identificar se os protocolos citados têm características de desempenho que os qualifica a serem utilizados como uma alternativa ao TCP.

2. Proposta de Trabalho

Inicialmente os protocolos de transporte TCP, SCTP e XTP serão apresentados brevemente. O desempenho dos protocolos em testes empíricos pode auxiliar na identificação do protocolo de transporte mais adequado para as aplicações atuais.

2.1. TCP

Segmentos ou mensagens são trocados para estabelecer conexões, transferir dados, enviar reconhecimentos e fechar conexões. O método de controle de fluxo utilizado pelo TCP é chamado de janela deslizante e baseia-se no envio do tamanho da janela de destino determinada pelo lado TCP destino. O TCP usa controle de congestionamento fim-a-fim, significando que o remetente limita ou aumenta a taxa de entrega de dados para conexão em função do congestionamento percebido por ele.

O TCP emprega um checksum de complemento de um, de 16 bits, cobrindo tanto o pseudo-cabeçalho IP, quanto os dados. Ele fornece um serviço de transmissão confiável, onde para cada segmento transmitido o host destino envia um segmento de confirmação. Uma vez que informação de estado é mantida, é dito orientado à conexão.

O processo normal de estabelecer uma conexão entre dois terminais TCP envolve três passos, chamado de handshake de três vias. Para o TCP o encerramento de uma conexão full duplex deve ser feito explicitamente pelos dois terminais envolvidos. O TCP ainda fornece um mecanismo para que dados presentes em determinado segmento sejam processados imediatamente no receptor, mesmo se o fluxo normal de dados foi parado pelo mecanismo de controle de fluxo.

2.2. SCTP

Algumas aplicações atuais necessitam de um protocolo de transporte que possua características tanto do TCP quanto do UDP, além de prover outros recursos. O surgimento do SCTP veio auxiliar neste ponto, visto que o mesmo é similar tanto ao TCP como ao UDP, mas com recursos adicionais.

O SCTP apresenta todas as características de um protocolo de transporte confiável, assegurando que os dados sejam transmitidos na rede sem erros, sem duplicação e em sequência. Ele, assim como o TCP, é orientado a conexão, por isso a transmissão dos dados é confiável e uma conexão precisa ser estabelecida. Os mecanismos de fluxo de dados disponíveis no SCTP são o controle de fluxo e o controle de congestionamento. O SCTP implementa um algoritmo de soma de verificação mais robusto que o do TCP, o CRC-32c, um algoritmo que usa 32 bits para a verificação. As confirmações no SCTP utilizam a confirmação seletiva. A retransmissão seletiva acontece quando o receptor detecta intervalos na numeração sequencial das mensagens de dados.

SCTP difere-se do TCP por realizar uma associação que possui um contexto e significado mais amplo que a conexão TCP. A associação SCTP corresponde a um número arbitrário de fluxos simplex (unidirecionais). O estabelecimento de associações se dá através da troca de quatro mensagens, procedimento denominado handshake de quatro vias, isso faz com que o estabelecimento da associação seja mais seguro, uma vez que os recursos são alocados após o recebimento da terceira mensagem. Assim são evitados ataques do tipo DoS, comuns no TCP por alocar recursos logo no recebimento da primeira mensagem.

A característica de múltiplos caminhos permite que uma associação tenha mais de um endereço IP em cada extremidade, tornando-a menos vulnerável a eventuais

problemas de rede. Para torná-lo mais flexível em ambientes onde a confiabilidade dos dados não é requisito primário, como aplicações em tempo real, foi especificada uma extensão, onde um nó pode informar ao outro que mensagens pedidas não serão retransmitidas. Esta extensão é conhecida como PR-SCTP (Partial Reliability SCTP).

2.3. XTP

Os mecanismos de fluxo de dados presentes no XTP são o controle de fluxo e o controle de taxa. O controle de erro no XTP é baseado na troca da informação quanto a dados perdidos ou corrompidos e a retransmissão desses dados e pode ser desligado pelo transmissor. A função de checksum é uma soma de complemento de um, de 16 bits, sobre todos os pares de octetos participantes. A retransmissão é feita por go-back-N ou confirmação negativa rápida (FASTNAK). Também se utiliza de um handshake de sincronização para controle de erro.

O XTP, assim como o TCP é orientado à conexão. O XTP é projetado para as redes de baixa taxa de erro de hoje e assim não emprega muito tempo com mecanismo de handshake. Por isso, uma conexão é estabelecida implicitamente enviando um pacote com a flag FIRST setada, que já pode conter dados. No XTP cada fluxo unidirecional de dados pode ser fechado separadamente usando um handshake de duas vias, alternativamente fechamento coordenado ou abortivo. Não possui mecanismos de segurança e, como verificado pelo estabelecimento de conexão implícito é, assim como o TCP, sujeito a ataques de negação de serviço.

O multicast de camada de transporte é uma característica única no XTP, que não existe em outros protocolos de transporte bem conhecidos, como TCP e UDP. O multicast XTP prove um mecanismo poderoso de comunicação em grupo que suporta um serviço de transferência de dados de um para muitos fluxos de dados.

3. Comparativo entre os protocolos TCP, SCTP e XTP

A Tabela 1 mostra as principais características dos protocolos TCP, SCTP e XTP.

Tabela 1. Comparativo das principais características

Característica	TCP	SCTP	XTP
Confiabilidade	Sim	Sim	Sim
Orientado à Conexão	Sim	Sim	Sim
Orientação da Transmissão	Octeto	Mensagem	Mensagem
Controle de Fluxo	Sim	Sim	Sim
Controle de Conestionamento	Sim	Sim	Não
Controle de Taxa	Não	Não	Sim
Distribuição dos Dados	Ordenada	Ordenada Ordenada Parcial Desordenada	Ordenada
Múltiplos Fluxos	Não	Sim	Não
Múltiplos Caminhos	Não	Sim	Não
Multicast	Não	Não	Sim
Pseudo-header para Checksum	Sim	Não	Sim
Permite Conexão Meio-abertas	Sim	Não	Sim
Proteção contra Ataques DoS	Não	Sim	Não

4. Resultados Experimentais

A metodologia utilizada para realização do comparativo de desempenho empírico dos protocolos TCP, SCTP e XTP é descrita nesta seção. O ambiente utilizado para realização dos testes foi simples, contando apenas com duas máquinas e uma rede doméstica de 10/100 Mbps, com sistema operacional Linux, distribuição Debian 5.0, com kernel 2.6.26. O módulo para SCTP foi instalado (LK-SCTP - Linux Kernel SCTP). Para o XTP tentou-se instalar o módulo de kernel Linux XTP, porém não foi

obtido sucesso na várias tentativas realizadas. Com isso os resultados apresentados para o XTP são resultados obtidos em outros trabalhos. Os aplicativos para teste de desempenho foram construídos na linguagem C e C++ e, permite variar o tamanho das mensagens transmitidas, o número de fluxos utilizados em SCTP, endereços para associação e o volume total trafegado em bytes. Os aplicativos utilizados foram: tcp_server, tcp_client, tcp_server_lat e tcp_client_lat.

Os testes realizados são de vazão e latência, em interface loopback e rede de 100 Mbps. Sendo vazão o volume de dados por unidade de tempo. O teste de vazão consiste de transmitir um volume fixo de bytes em ambas as direções. E latência, o tempo médio decorrido entre a requisição do cliente e a resposta do servidor. Foram realizados testes utilizando o protocolo "TCPM" que é um protocolo de aplicação que faz uma separação de mensagens simples. Para os testes de vazão os aplicativos tcp_server e tcp_client foram utilizados. Os testes realizados utilizaram mensagens de 5 MB, 278 MB, 1456 MB e 23256 MB. Além disso, realizou-se teste variando o número de fluxos em 10, 100 e 1000. Já para os testes de latência, os aplicativos utilizados foram tcp_server_lat e tcp_client_lat. Para cada mensagem enviada pelo cliente, exatamente uma é retornada pelo servidor.

No teste de vazão em loopback, mostrado na figura 1, verifica-se que o desempenho do XTP praticamente supera o TCP. O SCTP apresenta um desempenho muito ruim, isso se deve ao checksum utilizado (CRC-32c) e também a troca de contexto no destino devido à separação de mensagem. Os testes de latência em interface loopback realizados com somente um fluxo, como é apresentado na figura 2, variando o tamanho da mensagem, verifica-se um bom desempenho do XTP, o SCTP também apresenta melhora, em relação ao teste de vazão.

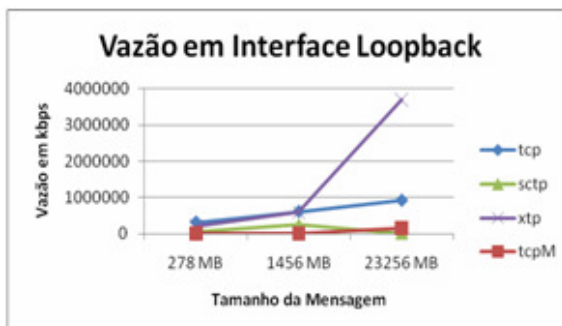


Figura 1. Vazão em loopback

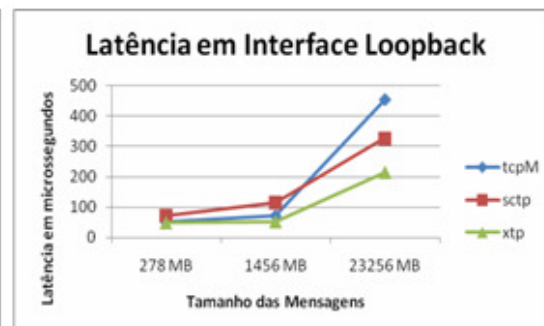


Figura 2. Latência em loopback

A análise do gráfico da figura 3 também revela a diferença significativa entre a latência apresentada pelo SCTP e o TCP.

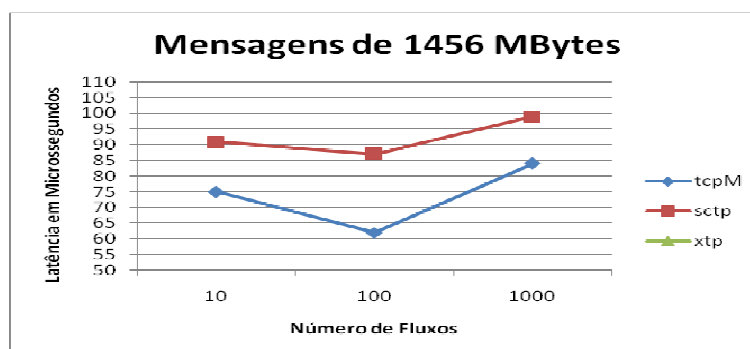


Figura 3. Latência em múltiplos fluxos - mensagens de 1456MB (loopback)

Os testes realizados em rede de 100 Mbps seguem os mesmos princípios que os realizados em interface loopback. Os testes de latência foram inicialmente realizados com um fluxo único, com volume de dados default (100 MBytes) e variação no número de mensagens. Após, realizou-se testes com múltiplos fluxos. Pelo gráfico da figura 4 verifica-se uma melhora do desempenho do SCTP no teste de vazão. O XTP tem quase o mesmo desempenho que o SCTP.

No teste de latência com fluxo único, mostrado na figura 5, percebe-se um comportamento bem similar em todos os protocolos, mas ainda identifica-se claramente a menor latência do TCP. O SCTP e XTP possuem praticamente o mesmo desempenho.

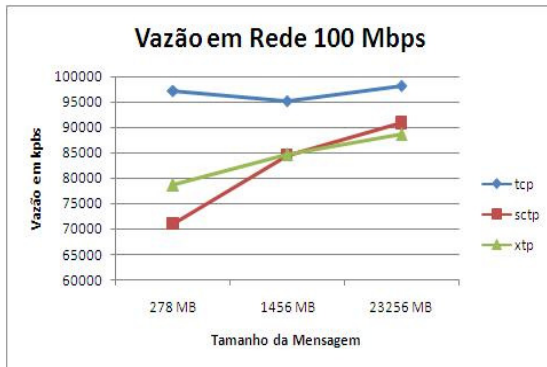


Figura 4. Vazão em rede 100Mbps

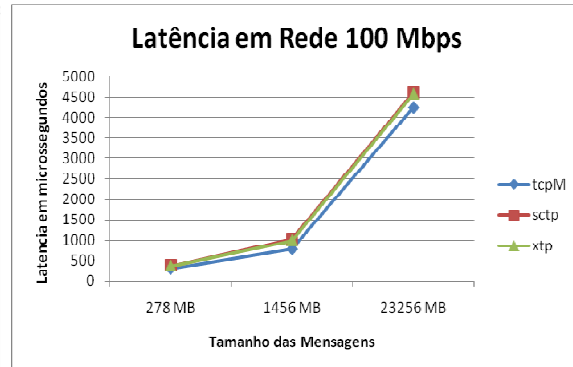


Figura 5. Latência em rede 100Mbps

Nas mensagens de 1456 MB, o já esperado melhor desempenho do TCP, novamente se confirma, mas entre o SCTP e o XTP observa-se a proximidade dos resultados obtidos, como pode ser verificado na figura 6.

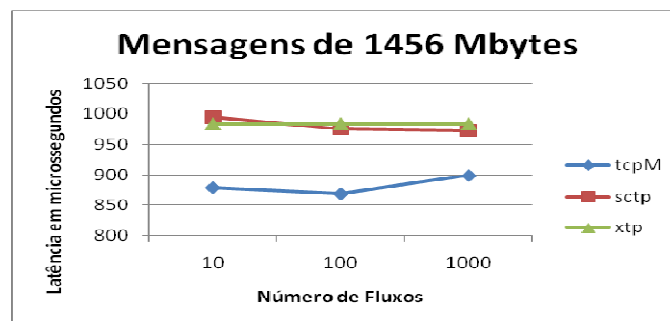


Figura 629. Latência em múltiplos fluxos - mensagens de 1456MB (rede 100Mbps)

5. Conclusões

De forma geral, os objetivos definidos para este trabalho foram cumpridos. Foi realizado um extenso estudo bibliográfico sobre os protocolos TCP, SCTP e XTP. Levantou-se suas principais características, mecanismos e modos de operação, de maneira a apresentar subsídios para que os protocolos de alto desempenho analisados possam ser identificados como uma alternativa ao tradicional TCP, quando este não atender as necessidades específicas de alguma aplicação ou rede. Na sequência, foi realizada uma comparação entre os principais mecanismos dos protocolos de transporte analisados, identificando o que cada um pode oferecer. Pelos testes empíricos realizados identificou-se que o desempenho do TCP ainda supera o do SCTP e XTP em um cenário comum, com aplicações que não exigem muito mais de um protocolo de transporte do que as características muito utilizadas e testadas do TCP.

Como trabalho futuro seria interessante estender os estudos apresentados neste trabalho, considerando-se um maior número de aplicações em um ambiente de produção, buscando mostrar as principais características dos protocolos, de forma a dar subsídios para que a escolha pelo protocolo beneficie a aplicação e otimize o uso da rede.

Referências

- COSTA, Daniel G. SCTP – Uma alternativa aos tradicionais protocolos de transporte da Internet. Editora Ciência Moderna, 2005.
- HENRICI, Dirk. A Universal Scheme for the Classification of Network Services. 2002. 121 f. Tese (Doutorado) - Universität Kaiserslautern, Kaiserslautern, 2002. Disponível em: <<http://dspace.icsy.de:12000/dspace/bitstream/123456789/33/1/DPArchiv.0087.pdf>>. Acesso em: 04 maio 2009.
- PFÜTZENREUTER, Elvis. Aplicabilidade e desempenho do protocolo de transporte Sctp. 2004. 122 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Ufsc, Florianópolis, 2004. Disponível em: <<http://epx.com.br/mestrado/>>. Acesso em: 13 maio 2009.
- SANTOS, Éverton Machado Dos; RIBEIRO, Glaucia da Silva; MINICZ, Márcio de Freitas. Sctp: Princípios e Funcionamento. RTI: Redes, Telecom e Instalações, São Paulo, n. 100, ano IX, p.72-81, setembro 2008. Mensal.
- STRAYER, W. Timothy (editor chefe). Xpress Transport Protocol Specification: Revision 4.0. XTP-95-20. Santa Barbara: XTP Forum, 01 mar 1995. Disponível em <<http://www.ir.bbn.com/~strayer/pubs.html>>. Acesso em 08 abr 2009.
- VIEGAS, Diogo Ruviaro. Estudo Experimental dos Protocolos TCP, Sctp E Xtp. 2008. 95 f. Dissertação (Mestrado) - Curso de Ciência da Computação, UFSC, Florianópolis, 2008.