UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Diogo Ruviaro Viegas

UM ESTUDO EXPERIMENTAL DOS PROTOCOLOS TCP, SCTP E XTP

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Mário Antônio Ribeiro Dantas, Dr.

Florianópolis, março de 2008.

UM ESTUDO EXPERIMENTAL DOS PROTOCOLOS TCP, SCTP E XTP

Diogo Ruviaro Viegas

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Mário Antônio Ribeiro Dantas, Dr.

Banca Examinadora

_____

Mário Antônio Ribeiro Dantas, Dr.

_____

Roberto Willrich, Dr.

_____

Carlos Becker Westphall, Dr.

_____

Carlos Barros Montez, Dr.

AGRADECIMENTOS

A Deus, por iluminar-me sempre.

A minha esposa, Melissa que, pelo amor, carinho, compreensão, paciência, sabedoria e que em todos os momentos esteve ao meu lado me dando forças e fazendo acreditar que eu poderia chegar aonde desejasse.

Aos meus pais, pelo amor e carinho incondicional em todas as fases da minha vida.

A minha família, que não me deixou desanimar e torceu para que o sonho virasse realidade.

A minha filha, Bruna, que, durante os momentos que pensei em desistir, a lembrança do seu sorriso me dava forças para continuar até ao final.

Ao meu orientador Mario Dantas, por suas palavras de incentivo, por sua sabedoria, por sua dedicação, sou eternamente grato e principalmente por não me deixar desanimar.

Ao Elvis Pfützenreuter por todo apoio, resoluções das minhas dúvidas e todo material para a realização deste trabalho.

Aos colegas da minha empresa, que com suas idéias, conselhos e sugestões me impulsionaram nesta longa caminhada.

Aos colegas de operadoras que não mediram esforços para mesmo a distância acrescentar com suas experiências informações fundamentais neste trabalho.

A todos aqueles que diretamente e indiretamente estiveram do meu lado durante todo esse caminho e que hoje comemoram a sua conclusão.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ADSL - Asymetrical Digital Subscriber Line.

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.

ATM - Asyncronous Transfer Mode.

AUC - Authentication Center.

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.

BSS - Base Service Set.

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 somatótio de 32 bits, com polinômio divisor diferente do CRC-32.

CSMA/CD - Carrier Sense Multiple Access with Collision Detection.

DCCP - Datagram Congestion Control Protocol.

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

DNS - Domain Name System.

EIR - Equipment Identity Register.

FDDI - Fiber Distributed Data Interface.

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

GGSN - Gateway GPRS Support Node.

GPRS - General Packet Radio Service.

GSM - Global system for Mobile Communications.

HOL - Head-of-Line Blocking - Problema em alguns protocolos confirmados que causa atraso na entrega dos dados.

HTTP - HyperText Transfer Protocol.

HLR - Home Location Register.

IANA - Internet Assigned Numbers Authority.

ICMP - Internet Control Message Protocol.

IETF - Internet Engineering Task Force.

IPSEC - Internet Protocol SECurity.

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

IPTables - Arquitetura de firewall do sistema operacional Linux, versão 2.4 em diante.

IPv4 - Internet Protocol version 4.

IPv6 - Internet Protocol version 4.

ISDN - Integrated Services Digital Network. Padrão de telefonia inteiramente digital.

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

ITU - International Telecommunication Union.

ITU-T - International Telecommunication Union – Telecommunication. Setor de padrões da ITU.

Kernel - Núcleo do sistema operacional.

Kernel level - Código que roda dentro do kernel.

LAN – Local Area Network.

LK-SCTP - Linux Kernel SCTP.

MAC - Midia Access Controle

M2UA - MTP-2 User Adaptation.

M2PA - MTP-2 Peer-to-Peer Adaptation.

M3UA - MTP-3 User Adaptation.

MSC - Mobile Switch Center.

MTU - Maximum Transmission Unit.

MTP - Message Transfer Part. A subpilha inferior da pilha de rede SS7. Subdividida em MTP-1, MTP-2 e MTP-3.

NAT - Network Address Translation.

NCP - Network Control Protocol.

NFS - Network File System.

Octeto - Byte com invariavelmente 8 bits.

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

PDP - Packet Data Control.

PEI - Protocol Engines Incorporated.

PMTU - Path Maximum Transmission Unit.

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

PTN – Public Telephone Network

RFC - Request For Comment.

RPC - Remote Procedure Call.

RSVP - Resource reSerVation Protocol.

RTP - Real Time Protocol.

RTO - Retransmission TimeOut. Tempo que o transmissor espera para receber uma confirmação do receptor. Após esse tempo, o pacote é considerado perdido e retransmitido.

RTT - Round-Trip Time. Tempo decorrido entre a transmissão do pacote e o recebimento da respectiva confirmação. É essencialmente a soma das latências de ida e de volta da rede.

SCCP - Signaling Conection Control Part. Camada de aplicação da pilha SS7.

SCTP - Stream Control Transmission Protocol.

SGSN - Serving GPRS Support Node.

SIGTRAN - Comitê Signaling Transport.

SMS - Short Message System.

SNA - Systems Network Architecture.

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.

SSH - Secure Shell Protocol.

SSL - Secure Sockets Layer.

SSN - Stream Sequence Number.

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

TCB - Transmission Control Block.

TCP - Transmission Control Protocol.

TCP/IP - Transmission Control Protocol/Internet Protocol. Sigla pela qual a pilha de protocolos da Internet é vulgarmente conhecida.

TCPM - No contexto desta dissertação, protocolo de aplicação criado para testes de desempenho.

Timestamp - Marca de tempo.

TLS - vide SSL

TLV - Type, Length and Value. Tipo de estrutura de dados de tamanho variável.

TSN - Transmission Sequence Number.

UDP - User Datagram Protocol.

UNIX - Família de sistemas operacionais.

UNIX socket - vide soquete UNIX.

UNIXM - No contexto desta dissertação, protocolo de aplicação criado para testes de desempenho.

U-SCTP - Unreliable SCTP.

VLSI - Very Large Scale Integration.

VPN - Virtual Private Network.

VoIP - Voice over IP.

VLR - Visitor Location Register.

XML - eXtensible Markup Language.

XTP - eXpress Transport Protocol.

WAN – Wide Area Network.

RESUMO

Nas redes de dados das operadoras de telefonia celular é fundamental que o protocolo utilizado na comunicação entre terminais seja confiável e seguro e que possa prover o máximo de garantias a integridade dos dados.

Os protocolos usualmente utilizados na comunidade científica fazem parte da pilha TCP/IP. O protocolo UDP não acrescenta confiabilidade à rede e apenas implementa a multiplexação. Por outro lado, o protocolo TCP tem a vantagem de ser um protocolo confiável possuindo conexões ponto-a-ponto. Na rede de telefonia celular, considera-se nas maiores das vezes o protocolo TCP e deseja-se que as vantagens de UDP existam no TCP. Desta forma, nesta dissertação optou-se pelo estudo dos protocolos SCTP e XTP por possuírem tais características em comum e acrescentar alguns diferenciais.

Entre os recursos que mais se destacam no SCTP estão a transmissão de mensagens indivisíveis, múltiplos fluxos de mensagens por conexão, variação da confiabilidade das mensagens, entre outras. Por outro lado o XTP tem como características trabalhar em aplicações de alto desempenho, e por possuir similaridades com o TCP.

A proposta deste trabalho de pesquisa é de prover um estudo empírico utilizando-se dos protocolos TCP, SCTP e XTP. O estudo caracteriza-se por considerar os recursos do SCTP e XTP, a fim de demonstrar em uma rede real interna de uma operadora de telefonia móvel o diferencial dos mesmos suportando protocolos de aplicação e simulando diversas condições de rede.

ABSTRACT

In datas networks of the mobile phone companies is fundamental that the protocol used in communication between terminals be reliable, safe, robust end must contains enough characteristics that insure the data´s integrity.

The protocols most used in scientific community and integrant part of TCP/IP battery The UDP protocol doesn´t increase reliability to networks and just implements the multiplexing, in the other hand the TCP protocol has the advantage to be a reliable protocol with peer to peer connections. In telephone's network, is used, in mostly of times, the TCP protocol and is desired that the advantages of UDP exist in TCP. Thou, it was discovered that SCTP can posses these characteristics in common and increase some differences.

Among the most values resources in SCTP are the transmissions of indivisible messages, multiplies messages flows using connection,, variation of messages reliability, among others. In other hand the XTP has as characteristics, to work in high performance application, and because of it has similarities with the TCP.

The propose of this work is provide a empiric study using the TCP, SCTP and XTP. The study is characterized because considers the resources of the SCTP and XTP, for the purpose of indicate, in a real internal network, the differential of the mobile phone company, supporting application´s protocols and simulating many conditions of network.

1. INTRODUÇÃO

O protocolo TCP (*Transmission Control Protocol*) é conhecido como uma evolução do nível de transporte da proposta existente na Arpanet, o NCP. Dentre suas várias utilizações, serve para administrar os constantes problemas de uma comunicação confiável em interfaces que não são confiáveis (UNICERT.COM,2007). Um exemplo clássico é da interligação de redes de datagramas ou redes de pacotes via rádio.

O fato da arquitetura TCP/IP ter sido aceita como um conjunto de protocolos padrão se deve a alguns fatores como: o desejo em protocolos de transporte entre as redes é algo relativamente recente e durante todo o seu tempo de maturação, a tecnologia avançou de tal maneira que o TCP/IP pode ser colocado em qualquer tipo de equipamento sem algum custo considerável.

O modelo de interconexão existente em sistemas abertos, como o OSI da ISO, é seguido por protocolos hierarquizados e possuem a característica de trabalharam em pares onde cada camada de protocolo contém uma "conversa" com a camada igual no outro lado da conexão.

Por outro lado, alguns protocolos têm sido estudados, visando garantir um melhor desempenho em redes comerciais com uma abordagem leve quando comparado com o convencional protocolo TCP. Um exemplo é o protocolo SCTP que chama a atenção desde o inicio para a junção das pilhas de rede SS7 e TCP/IP, na troca de mensagens de sinalização telefônica.

Outras camadas de adaptação disponíveis são M2UA (MORNEAULT et al, 2002) e M2PA (GEORGE,2003), que em conjunto com TCP/IP substituem as camadas MTP-1 e MTP-2, mas mantêm a camada MTP-3 do SS7.

Conforme (PFÜTZENREUTER, 2004) comenta que a sinalização telefônica é o fluxo de informações administrativas, como mensagens SMS, tarifação, serviços ao usuário, sinal de ocupada etc, transmitida por uma rede de comutação de pacotes. A transmissão do conteúdo (voz) faz uso de uma rede de comutação de circuitos separada. Nessa função criticamente

importante, o SCTP já é largamente utilizado, e cada vez mais na medida em que as empresas telefônicas migram a rede de sinalização inteiramente para TCP/IP.

Assim, sabe-se também que o SCTP tem um grande potencial para funcionar fora do ambiente de telecomunicações, podendo ser usado com todas as características dos protocolos padrões de transporte mais alguns adicionais em ambiente aonde atuam protocolos de aplicação da Internet.

Com a motivação de reunir os protocolos TCP, SCTP e XTP num ambiente real como as operadoras de celular, fazendo com que aplicações de alto desempenho consigam através de testes aqui descritos, verificar para determinados cenários qual protocolo usar. As operadoras de celulares em âmbito mundial estão atravessando um período de intensa atividade que tem como principal objetivo possibilitar a convergência de serviços. Esta convergência engloba os serviços de telecomunicações bem como o acesso aos serviços de internet possibilitando que qualquer conteúdo trafegue em qualquer rede.

Tem havido um movimento considerável em diversos grupos de estudo com o objetivo de integrar redes de circuitos comutados com as redes IP.

Em telefonia, conforme RUSSELL e ARIAS-RODRIGUEZ (RUSSEL, 2002) é muito comum a troca de mensagens atômicas entre sistemas; é essencial um serviço de datagrama confirmado, ou de mensagens atômicas confirmadas.

Isto deverá permitir que os serviços disponíveis nestas redes sejam trazidos para um ambiente de rede IP e permitir a aquelas redes o uso diversificado de plataformas IP em suas soluções de acesso ou *backbone*.

O objetivo principal do SIGTRAN do IETF é permitir o transporte da sinalização considerando os requisitos funcionais e de desempenho das redes de comutação de circuitos. Assim, para funcionar com uma rede de comutação de circuitos, as redes IP necessitam carregar mensagens de sinalização, seja de assinantes digitais (DSS) seja do SSNº7, entre nós IP, tais como *Signalling Gateways, Media Gateway Controller, Media Gateway ou Bases de Dados IP*.

A primeira idéia, segundo ARIAS-RODRIGUEZ (RUSSEL, 2002), foi simplesmente adotar o TCP para o transporte de mensagens, por ser tradicional e bem conhecido, e tentar melhorar seus pontos fracos.

A necessidade de novos blocos funcionais ficou evidente com o estudo dos requisitos para sinalização, pois os blocos com funcionalidade equivalentes existentes (TCP e UDP*)* não atendiam às especificações de segurança e qualidade de serviço imposto aos Sistemas de Sinalização do ITU-T. As principais limitações do TCP dizem respeito à segurança, já que o TCP é vulnerável a fraude, e aos mecanismos de seqüenciamento empregados. O TCP fornece uma transferência confiável e entrega na ordem correta dos dados através de um mecanismo de reconhecimento e bloqueio de transmissão, porém este mecanismo o torna inviável para aplicações em tempo real.

O novo conjunto de blocos funcionais consiste de 3 subcamadas:

- Protocolo padrão IP;
- Uma subcamada de transporte comum de sinalização, suportando um conjunto comum de funções de transporte confiável;
- Uma subcamada de adaptação que suporta primitivas específicas requeridas por uma aplicação da sinalização particular.

A subcamada de transporte comum foi denominada *Protocolo de Transmissão de Controle de Vazão - Streaming Control Transmission Protocol - SCTP*. Embora a arquitetura SIGTRAN tenha sido o principal elemento motivador no desenvolvimento do SCTP, espera-se que outras aplicações venham a se beneficiar de sua estrutura.

O *Transporte de Sinalização* (*Signaling Transport - SIGTRAN)* é a união de padrões definidos pelo *Internet Engineering Task Force* (IETF). O objetivo desta união de padrões foi de gerar um modelo de arquitetura para o transporte de sinalização sobre redes IP.

As subcamadas de adaptação específicas foram desenhadas de modo a permitir flexibilidade na aplicação dos servidores ou elementos de rede IP. O cenário utilizado para o estudo comparativo foi em cima da rede de telefonia de uma operadora, aonde trafegam dados GPRS.

A Rede GSM é uma rede otimizada para voz que é a sua principal aplicação. No inicio, seus objetivos eram de produzir na rede móvel os serviços de dados disponíveis na rede fixa, isso tudo através da ISDN. Os canais do GSM, através da sua estrutura flexível juntamente com o protocolo SS7, ajudaram o inicio dos serviços SMS (short message service), FAX e principalmente transporte de dados.

Com o crescimento e utilização das aplicações de dados, apareceu a necessidade de desenvolver soluções que fizessem com o transporte de dados ocorresse a taxa maior. Ao se estabelecer uma conexão na rede GSM, utiliza-se um "slot" de tempo com taxa de até 9,6 kbit/s.

No GPRS (General Packet Radio Service) pacotes de dados são encaminhados por múltiplos slots de tempo que acabam sendo alocados conforme a demanda dos pacotes enviados e recebidos.

GPRS possui algumas características que deverão ser ressaltadas aqui como:

- Taxa de transporte de dados máxima de 26 a 40 kbit/s
- Conexão de dados sem necessidade de se estabelecer um circuito telefônico, o que permite a cobrança por utilização e não por tempo de conexão e faz com que o serviço esteja sempre disponível para o usuário (always on).
- Implantação implica em pequenas modificações na infra-estrutura instalada, o que facilita a sua adoção pelos operadores de GSM.
- Padronizado para transporte de dados definidos pelos protocolos IP e X.25.

A figura 1, mostra como é a arquitetura de uma rede GSM.

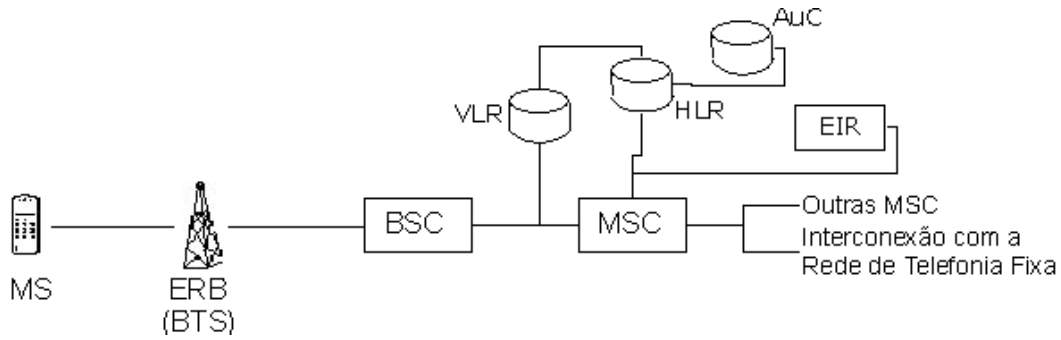

Figura 1: Arquitetura da rede GSM (TELECO.COM, 2008)

Na figura 1, algumas siglas tem que ser explicadas para um maior entendimento quando se fala em GPRS (TELECO.COM, 2008).

- MS é uma estação móvel, no qual está carregado com um SIM Card.
- BSS é o sistema responsável pela comunicação móvel em determinado local. Constituído por ERBs e BTS que juntas constituem uma célula e uma BSC que monitora e controla as BTSs.
- MSC que seria uma central de comutação e controle que a função de comutação e sinalização para as estações móveis.
- HLR que seria um registro de assinantes locais de um sistema celular.
- VLR que seria um registro de assinantes visitantes (roaming) em um sistema celular.
- AUC como centro de autenticação do assinante.
- EIR como um registro de identidade do equipamento.

Assim para poder-se adicionar GPRS em uma rede GSM, teremos uma nova estrutura da arquitetura como mostra a figura abaixo:

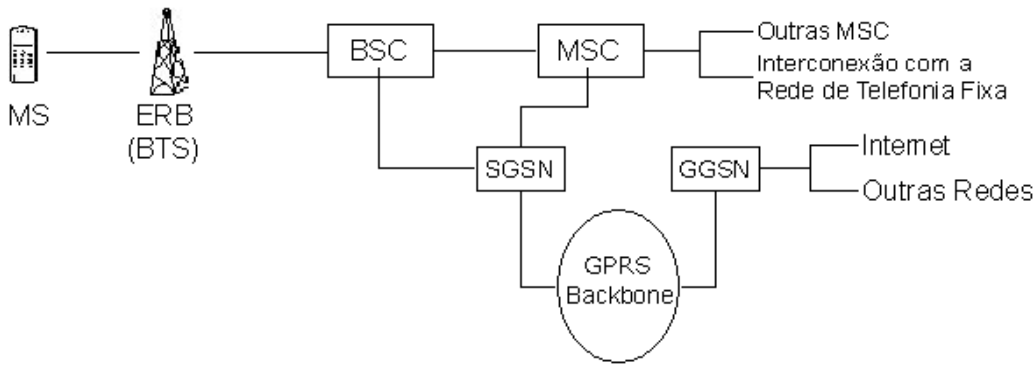

Figura 2: Arquitetura da rede GSM com GPRS (TELECO.COM, 2008)

Os novos elementos são:

- SGSN que tem como papel principal manter a conexão quando um usuário passe uma área de cobertura de uma célula para outra.
- GGSN que permite a conexão a internet e outras redes de dados.

Para poder utilizar o GPRS necessita-se um terminal que suporte esse serviço. A conexão entre ambos é realizada através dos seguintes passos:

- O terminal GPRS deve ser reconhecido pela rede como acontece com o terminal GSM de voz. Cria-se então uma ligação lógica entre o terminal e o SGSN. Assim o terminal está registrado e autenticado na rede.
- Consegue-se após isso um endereço IP estabelecendo uma conexão GPRS através da ativação do PDP (Packet Data Protocol). Este endereço IP é informado pela operadora e outro operador de acordo como está configurada a rede.
- O Terminal GPRS está pronto para enviar e receber pacotes. Assumindo assim alguns estados como: Ilde, Ready ou stand-by.

1.1 - Objetivo geral

O objetivo geral deste trabalho é de comparar protocolos de leves como SCTP e XTP fornecendo uma alternativa para aplicações de alto desempenho para redes de dados de operadoras celulares e sistemas distribuídos. O protocolo SCTP foi criado recentemente (2000), e é utilizado para as comparações com o TCP, em redes de telecomunicações (RFC 2960).

1.2 - Objetivos específicos

- Apresentação dos protocolos usados para o estudo comparativo;
- Comparação entre protocolos leves de alto desempenho;
- Alteração de aplicativos básicos para os testes de vazão e latência;
- Utilização de protocolos de aplicação adaptados para o SCTP;
- Criação de cenários típicos de rede, pela variação de banda, latência e perda de pacotes;
- Criação de ambiente multicaminhos, testes de funcionalidade e resistência a falhas de rede;
- Testes de desempenho nos ambientes criados.

1.3 - Metodologia empregada

A metodologia empregada neste trabalho foi uma extensa pesquisa bibliográfica, com a preocupação em detalhar os seguintes tópicos:

- o protocolo TCP;
- o protocolo SCTP;
- o protocolo XTP;
- algoritmos de checksum utilizados nos protocolos SCTP;
- comparativos entre protocolos;
- redes internas de operadoras de celular;

- Rede GSM e GPRS;

Esta dissertação está organizada em 6 capítulos. O capítulo 2 aborda sobre protocolos de transporte partindo para um estudo em cima do protocolo TCP, mostrando algumas características deste protocolo que serão colocadas em analise com os demais protocolos estudados. Por outro lado, no capítulo 3 o protocolo SCTP é apresentado com suas principais virtudes que chamaram a atenção para que fosse colocado neste estudo comparativo. O capitulo 4 é caracterizado pelo protocolo XTP que possui similaridades com os protocolos TCP e XTP e foi considerado de fundamental importância na comparação realizada. Os resultados experimentais e ambientes onde os testes foram realizados estão no capitulo 5. Finalizando, o capitulo 6 apresentam conclusões e trabalhos futuros.

2. PROTOCOLOS DE TRANSPORTE

2.1 – Definição

Confome (DANTAS,2002) comenta que os protocolos são um conjunto de regras que determinam como deverá ocorrer a comunicação entre duas estações numa rede de comunicação e como os erros devem ser detectados e tratados.

Salienta-se também que aspectos como sintaxe e semantica devem ser incorporados aos protocolos, visto que o formato dos dados é orientado pela sintaxe de um protocolo bem como os níveis de sinais que devem ser considerados pelo protocolo.

Algumas facilidades citadas por (DANTAS, 2002) que um protocolo pode suportar, tais como:

- A comunicação entre um computador e o meio físico de uma rede de computadores.
- O acesso a uma rede de comunicação através de um computador.
- O transporte de dados entre os computadores.

Os protocolos possuem algumas caracteristicas em comum que permitem com que seja feita uma classificação em cima deles. Dependendo do tipo de comunicação que é efetuada entre duas estações pode-se determinar a complexidade de um protocolo, caso de uma rede ponto-a-ponto aonde a comunicação é direta. Por outro lado, em redes comutadas existe a preocupação da gerência de uma comunicação indireta, assim existe a necessidade de um protoloco mais complexo.

Protocolos têm como característica também serem simétricos ou assimétricos. No caso dos simétricos a comunicação será fim-a-fim. Diferente dos simétricos os assimétricos caracterizam-se determinar como será feita a comunicação por um dos elementos envolvidos.

Os protocolos de transporte têm como objetivos conectar dois programas. Você pode ter em um mesmo computador vários programas trabalhando com a rede simultaneamente, por exemplo, um navegador Web e um leitor de e-mail. Da mesma forma, um mesmo computador pode estar

rodando ao mesmo tempo um servidor web e um servidor POP3. Os protocolos de transporte (UDP e TCP) atribuem a cada programa um número de porta, que é anexado a cada pacote de modo que o TCP/IP saiba para qual programa entregar cada mensagem recebida pela rede.

Dentro os protocolos de transporte, os mais conhecidos são o TCP e UDP.

2.2 - Protocolo TCP

O modelo TCP/IP surgiu como sendo um modelo mais simples e específico para o padrão da Internet. O modelo TCP/IP chamada a atenção pela máxima flexibilidade que acontece na camada de aplicação para os desenvolvedores de software.

O protocolo TCP é hoje o mais utilizado e é o padrão para troca de informações na Internet, 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, pois tiveram no seu desenvolvimento objetos de oferecer confiabilidade trabalhando com vários ambientes de rede, com diferentes ferramentas e com muitos aplicativos.

No TCP, um pacote de dados é tratado como uma seqüência de bits dividida em octetos. A aplicação entrega ao protocolo de transporte os dados no formato de pacote e o protocolo pode segmentar o pacote na melhor foram que for para a transmissão.

Os serviços orientados para conexão agrupam três fases. Na fase inicial, criação da conexão, um único caminho entre o terminal origem e o terminal destino é criado. Assim, os recursos são reservados para garantir um nível consistente de serviço. Na próxima fase, a de transferência de dados, os mesmos são encaminhados em seqüência pela conexão estabelecida, encerrando junto ao terminal destino na ordem em que foram enviados. A ultima fase, de encerramento da conexão, serve para encerrar a conexão entre o terminal origem e o terminal destino quando não é mais necessária.

O processo de troca de números de seqüência na fase inicial é de suma importância, pois ela garante que os dados perdidos devido a problemas de transmissão possam ser recuperados.

Primeiramente, o terminal origem inicia uma conexão pelo envio de um pacote indicando o número seqüencial inicial com um bit no cabeçalho definindo assim um pedido de conexão. Desta maneira, o terminal destino recebe o pacote e grava um numero de seqüência "a", devolvendo com uma confirmação "a" + 1 e inclui o seu próprio numero seqüencial inicial "b". O número de confirmação "a" + 1 é que o terminal destino recebeu todos os octetos até "a", inclusive, e que está esperando "a" + 1 em seguida.

Este tipo de confirmação positiva (ACK) é um processo comum utilizado por muitos protocolos para poder fornece confiabilidade na transmissão. Com o ACK, o terminal origem encaminha um pacote, ligando o temporizador e espera por uma confirmação antes de poder enviar o pacote seguinte.

Na janela de transmissão, o seu tamanho indica o número de dados que poderá ser transmitido antes de ser recebida uma confirmação do terminal destino. Ao aumentar o tamanho da janela do receptor, aumenta-se a quantidade de dados que o terminal origem pode encaminhar sem a necessidade de confirmar o recebimento. Após um terminal transmitir a quantidade de pacotes da janela calculada do terminal destino, o terminal de origem precisa receber uma confirmação de que todos os segmentos chegaram ao destino antes de encaminhar mais dados. Desta maneira, o TCP utiliza um processo conhecido como confirmação de espera, nada mais é que o número da confirmação se refere ao segmento que é esperado em seguida. O processo de janela é um esquema de controle de fluxo que faz com o que o terminal de origem receba uma confirmação do terminal destino após transmitir um numero de dados. Assim sendo, tomamos o exemplo de uma janela de tamanho três, no qual o terminal origem pode enviar até três segmentos ao terminal destino, aguardando assim uma confirmação. Caso o terminal destino receba os três segmentos, ele deverá enviar uma confirmação ao terminal de origem, que poderá encaminhar assim mais três segmentos, pode ser que por algum motivo, o terminal destino não receba os três segmentos, então ele não enviará uma confirmação (COMER, 1998). Devido a não receber a

confirmação, o terminal origem compreenderá que os segmentos deverão ser encaminhados novamente e que a taxa de transmissão deve ser diminuída.

2.2.1 – Transmissão Confiável

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.

Para detecção de segmentos perdidos, TCP fornece dois mecanismos de detecção, um através de temporizador, definido como RTO, sendo 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 ACK, então o protocolo assume que o segmento foi perdido e inicia-se uma retransmissão do segmento perdido (COMER, 1998).

Figura 3: Transmissão confiável TCP

Por outro lado, a detecção de perdas poderem ser feita através de ACK duplicados, partindo como exemplo o terminal origem enviados os segmentos 1,2,3 e 4 sendo que o segmento 1 seja perdido. Desta maneira, quando o segmento 2 chegar, o terminal destino envia um ACK com o

número de seqüência do segmento 1. No momento em que receber o segmento 3, ele envia um ACK com o numero de seqüência do segmento 1, e assim por diante. Neste caso, quando vários ACK duplicados são recebidos com o mesmo número de seqüência, o terminal origem valida que o segmento foi perdido ou danificado e realiza o processo de retransmissão.

2.2.2 - Controle de fluxo

O processo de controle de fluxo neste protocolo, baseia-se no envio do tamanho da janela de destino determinada pelo lado TCP destino. O protocolo utiliza um processo especifico de janela deslizante para contornar o controle de fluxo a fim de garantir uma transmissão eficiente. Com a utilização do processo de janela deslizante, é possível o envio de uma quantidade de segmentos ao mesmo temo antes que uma confirmação de recebimento retorne, aumentando consideravelmente o desempenho da rede. O tamanho da janela é a quantidade de segmentos que o lado origem TCP é permito enviar sem a obrigação de esperar as confirmações do lado destino.

Neste processo de janela deslizante, o protocolo permite que o tamanho da janela seja variável com o tempo, sendo que em cada ACK gerado pelo terminal destino, um novo tamanho de janela de advertência é definido, determinando assim quantos novos segmentos o receptor estará esperando.

O processo de controle de fluxo é muito importante em trocas de dados na rede, chamando a atenção onde comunicação atravessa várias redes e roteadores com capacidades diferentes. O fluxo de dados apresenta dois problemas, onde o primeiro está ligado ao TCP quando o mesmo necessita de um controle fim-a-fim, necessitando de um controle de fluxo entre o terminal origem e o terminal destino. O problema ocorre quando o terminal destino tem um desempenho menor que o terminal origem, desta maneira, o terminal destino deve regular o fluxo de dados com o objetivo de garantir uma transmissão confiável e evitando a perda de segmentos. Assim sendo, fazendo com que o terminar origem não envie mais dados do que o terminal destino possa processar. Finalizando os problemas, o ultimo acontece quando o controle de fluxos deverá ser realizado nos sistemas ditos “intermediários” como os roteadores, fazendo com que o terminal origem não envie uma quantidade de dados maior do que o roteador possa suportar. Caracteriza-

se desta maneira o controle de congestionamento, desta maneira, tem-se que a janela de congestionamento é o controle de fluxo criado pelo terminal origem, enquanto o tamanho da janela anunciada de advertência é o controle de fluxo imposto pelo receptor (PETERSON, 1996).

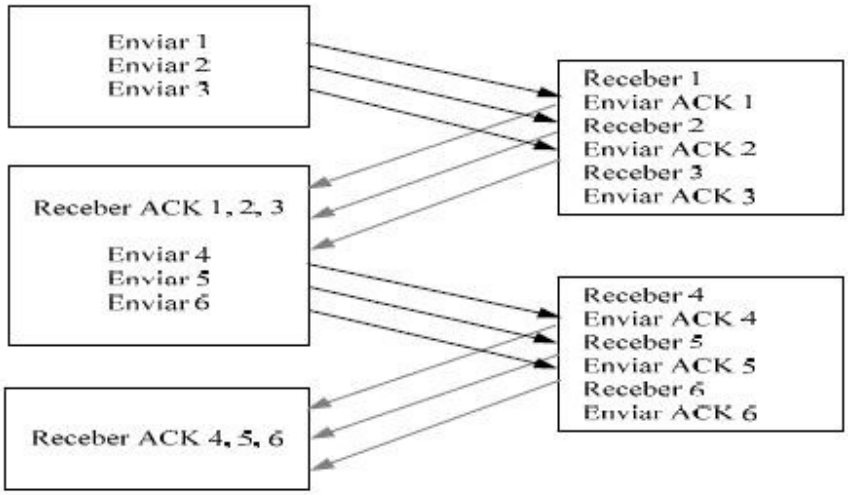

Figura 4: Controle de fluxo TCP

2.2.3 – Confirmação e retransmissão

TCP foi projetado para encaminhar segmentos de tamanho variável, além de retransmitir segmentos que carregam mais dados do que o original, as mensagens de confirmação não são ligadas a datagramas e nem a segmentos. Ao contrário disso, as confirmações estão ligadas ao número de seqüência que mostra o numero de bytes que foram recebidos. O terminal destino agrupa os segmentos de dados que chegam e monta uma cópia do fluxo enviado. Devido aos segmentos serem enviados em datagramas IP, eles podem ser perdidos ou serem entregues fora de ordem. Assim, o terminal destino utiliza os números de seqüência para poder reordenar os segmentos, sempre validando o maior prefixo no fluxo de dados recebido (JARDINI, 2000).

O processo de confirmação é chamado de cumulativo devido mostrar quanto do fluxo de dados foi acumulado no terminal destino. O problema ocorre quando o terminal origem recebe apenas

uma informação sobre a posição do fluxo de dados que o terminal destino recebeu sem erros, não sendo, portanto, informado sobre as transmissões que ocorreram sem problemas.

Caso o terminal origem utilize o processo de retransmitir apenas o primeiro segmento que não foi reconhecido, ele deverá esperar que a confirmação deste segmento chegue para então decidir quanto mais pode enviar.

3 – PROTOCOLO SCTP

O SCTP *(Stream Control Transmission Protocol)* é um protocolo de transporte da pilha TCP/IP.

O SCTP teve como principal função na sua fase inicial de desenvolvimento, ser usado para troca de mensagens telefônicas na internet, auxiliando e facilitando o transporte de protocolos como o SS7. Entretanto, as características do SCTP fazem com que o seu uso seja interessante para aplicações comuns na Internet, fazendo com que sejam incorporados os protocolos de transporte da pilha TCP/IP.

Entre as principais vantagens do SCTP destaca-se a de poder gerar funções flexíveis a diversas aplicações e que não estão presentes no UDP ou TCP.

A definição formal do protocolo está na RFC 2960 (STEWART et al, 2000). No entanto, esta RFC tem diversos erros e omissões. O SCTP Implementer's Guide (STEWART et al., 2003c) contém a listagem das diversas correções que serão futuramente incorporadas à RFC original.
A RFC 2960 contém a primeira especificação do protocolo SCTP, datada em outubro de 2000. Durante os últimos anos, novas RFC foram geradas para este protocolo, mostrando que cada vez mais a sua usabilidade faz com que o protocolo torne-se mais robusto.

3.1 - Características principais

O protocolo apresenta todas as características de um protocolo de transporte confiável, porém com funcionalidades a mais que o TCP. Dentre estas funcionalidades, destacamos:

- Entrega confirmada de dados, não duplicados.
- Fragmentação de acordo com MTU encontrado durante o caminho.
- Entrega seqüencial de dados em múltiplos fluxos.
- Empacotamento de múltiplas mensagens em um único pacote.
- Tolerância à falhas da rede.

Da mesma maneira que o TCP, o SCTP fornece um serviço confiável de transporte, garantindo que os dados sejam entregues sem erro e não os duplicando e sempre em seqüência. O SCTP

consegue descobrir quando os dados são duplicados ou corrompidos, descartados, fazendo a retransmissão quando necessário.

Devido o SCTP ser orientando a conexão, existe a confiança na transmissão dos dados. Para entender esse processo, deve-se lembrar que a conexão necessita ficar ativa durante a comunicação, devendo ser iniciada antes de se enviar qualquer dado. Este processo ocorre também no TCP, mas SCTP difere-se por realizar uma associação que possui um contexto e significado mais amplo que a conexão TCP.

O SCTP tem como característica ser unicast (ONG, 1999) embora os terminais envolvidos na comunicação sejam multicast, IPV6. Outra característica comum ao TCP é o fato de SCTP também ser *rate adaptative*, garantindo a esse protocolo adaptações às variações da rede, como por exemplos os congestionamentos que aparecem. Algoritmos de controle de congestionamento foram reformulados pelo SCTP, visto que suas funcionalidades já estavam testadas em congestionamentos na Internet.

3.2 - Associação em SCTP

Uma associação, como TCP, possui a conexão fim-a-fim, sendo controlado por protocolos de transporte. A diferença entre a associação SCTP e uma conexão TCP é que SCTP possui um número qualquer de fluxos unidirecionais, combinado durante o inicio da associação, enquanto TCP é constituído por um único fluxo full-duplex. Assim, a multiplicidade de fluxos torna-se uma característica do SCTP, garantindo uma maior eficiência na comunicação (COSTA,2005).

Assim, cada fluxo SCTP gerado é um canal de comunicação unidirecional, podendo até nesse caso acontecer de haver um número desigual de fluxos em cada direção na comunicação. Havendo a necessidade de criar-se uma conexão TCP por uma associação SCTP, cria-se um fluxo SCTP em cada direção.

Um dos grandes problemas encontrados no protocolo TCP é com relação ao fato de o mesmo permitir que um lado continue mandando dados mesmo quando o outro lado da conexão deixa de existir, fazendo com que as conexões fiquem "meio abertas", ou seja, apenas um lado está conectado e sinalizando que está enviando dados. No SCTP os terminais param de aceitar novos dados da camada de aplicação quando um dos participantes abandona a conexão.
Na figura abaixo, encontra-se uma representação do modelo de múltiplos fluxos das associações SCTP comparando com as conexões TCP.

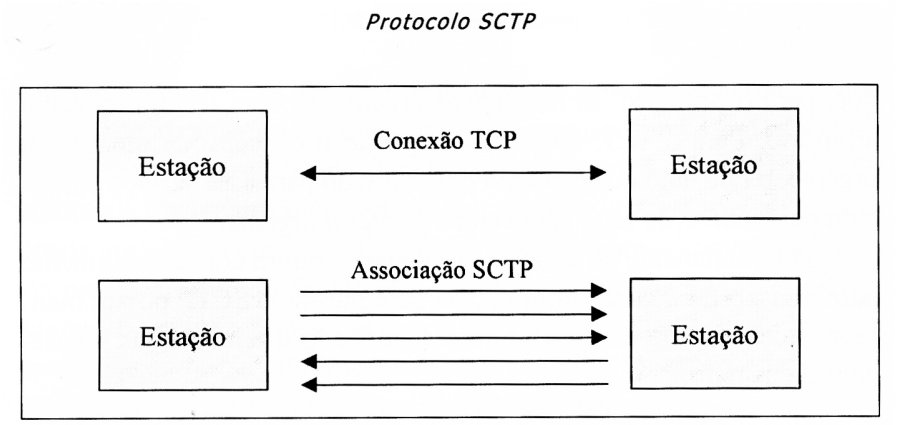

Figura 5: Modelos das Associações (COSTA, 2005)

3.3 - Orientação a mensagens

Mesmo os dois protocolos, SCTP e TCP, parecendo-se nas suas características, chama-se a atenção para a orientação das mensagens. No TCP a orientação é por octetos, não preservando qualquer estrutura de dados dentro da transmissão, já no SCTP a orientação é por mensagens, sendo os blocos tratados como independentes, com significado próprio.

O TCP garante a confiabilidade na entrega, o UDP a atomicidade e o STCP garantem os dois serviços, visto que cada mensagem é tratada atomicamente de uma forma independente, lembrando um serviço de datagrama confiável.

O processo de orientação a mensagens controladas pelo SCTP faz com que a mensagem de texto “teste” seja tratada como uma mensagem de texto e não como o TCP trata, ou seja, uma seqüência de octetos, recebendo assim uma identificação de mensagem. Mesmo que exista alguma restrição por parte de MTU, acontece um tratamento com relação a cada fragmento da mensagem gerada devido à restrição. Assim, SCTP fornece números que identificam as mensagens enquanto TCP gera números que identificam os octetos.

Esta diferença tem um papel significativo no desenvolvimento de aplicações, para quais as aplicações que utilizam o TCP faz com que cada dado de usuário receba alguma marcação especial. Para TCP os dados são apenas octetos que podem ser transmitidos de qualquer forma e com outros octetos de outras informações. Em Costa (COSTA, 2005) a transmissão deste protocolo faz com que uma mensagem de texto e um dado de controle possam ser enviados dentro de um mesmo segmento e se as aplicações não colocassem marcações apropriadas, a aplicação que recebesse os dados não saberia identificar a mensagem recebida uma vez que diversas informações poderiam ter sido misturadas.

Para aplicações desenvolvidas que necessitam confiabilidade ao nível de transporte, a orientação a mensagens gerada pelo SCTP facilita o seu trabalho. As mensagens de texto e controle são enviadas pelo SCTP como mensagens de texto e de controle, pois o SCTP não mistura as informações de usuário com significados diferentes. No caso de mensagens urgentes, que são aquelas processadas mesmo que seu contexto esteja fora de ordem em relação à ordenação, o SCTP permite que tenham um tamanho arbitrário, ao contrário do TCP que limita por causa da indicação feita pelo ponteiro urgente.

3.4 - Múltiplos Caminhos

O multihoming (múltiplos caminhos) faz com que um terminal possa utilizar mais de um endereço IP numa associação, mesmo sem um roteador. Garantindo assim uma comunicação mais confiável caso haja problemas de rede.

Em um terminal com um único endereço de rede, a falha de acesso a uma LAN pode fazer com que o terminal destino fique isolado e uma falha no núcleo pode causar uma indisponibilidade temporária, resolvida apenas quando os protocolos de roteamento IP ultrapassem o ponto com problema. Assim, SCTP utiliza outros acessos para reforçar a comunicação.

SCTP utiliza um par de endereços primários e um número qualquer de alternativos. O campo "endereço de origem" dos datagramas pertencesse à associação e originados pelo terminal utilizam o endereço primário. O campo "endereço de destino" possui o endereço primário do terminal destino da associação. Desta maneira, um caminho primário é constituído de dois endereços primários da associação.

Devido a problemas de rede, o endereço primário pode tornar-se inalcançável, fazendo com que os endereços alternativos sejam utilizados pelo par de comunicações para mensagens SCTP. Isso faz com que a probabilidade se alcançar o ponto remoto aumente, devido a alguma falha no caminho primário. Pode acontecer de o problema no caminho continuar, assim todas as mensagens são enviadas pelos endereços alternativos e um novo endereço primário é então estabelecido para essa estação.

Durante o inicio da associação acontece à troca de listas de endereços primários e alternativos entre os terminais SCTP. Em determinada associação, apenas uma porta é utilizada para todos estes endereços e estação que fornece apenas um endereço primário, não possuem endereço alternativos para a comunicação.

Outra característica importante nesta seção é que uma estação SCTP pode considerar uma estação indisponível devido aos endereços primários e alternativos serem considerados inalcançáveis. Para finalizar esta seção ressalta-se que as portas utilizadas pelo SCTP são as

mesmas que o TCP e UDP utilizam e não ocorrem “conflitos”, pois o protocolo de transporte é utilizado para identificar uma comunicação particular.

3.5 - Múltiplos fluxos

No início de uma associação SCTP os usuários de serviços destes protocolos definem a quantidade de fluxos de transmissão e recepção por eles suportados, assim no momento que é estabelecida a associação é calculado um número comum aos dois terminais visto que cada um determinou o número por eles suportado. Assim, em uma associação SCTP, múltiplos fluxos são utilizados para comunicações entre os dois terminais. Cada mensagem de usuário possui além dos dados de usuário, um determinado fluxo da associação, recebendo assim um número de fluxo. Uma mensagem de usuário pode estar associada a qualquer um fluxo de transmissão definido no inicio da associação. A mensagem recebe também um identificador relacionado à posição no fluxo e esse número de seqüência no fluxo juntamente com o número do fluxo forma a identificação de mensagens em relação à ordenação.

A idéia de múltiplos fluxos de transmissão e recepção nada mais é do que um conceito lógico fim-a-fim adotado para ordenação de mensagens e para o protocolo SCTP a maneira de transmissão e a fila de ordenação é definida em cada fluxo adquirindo-se assim uma maior eficiência caso ocorram perdas ou erros nas mensagens, visto que no SCTP a perda de uma mensagem em um fluxo não afeta os demais da associação, o que não acontece no TCP que possui apenas uma maneira de ordenar e retransmitir.

O processo de retransmissão apenas no fluxo afetado é garantido pelos algoritmos de ordenação que são aplicados em cima do fluxo com problema, permitindo desta maneira que o terminal receptor continue distribuindo mensagens para a camada de aplicação enquanto armazena em buffer as mensagens do fluxo com problema até a retransmissão. Este processo de retransmissão é seletivo fazendo com que os fluxos não afetados não entrem no processo de retransmissão, ao contrário do TCP. Os benefícios são vários e entre eles a economia nos recursos de rede

decorrentes da não retransmissão dos fluxos com mensagens já recebidas, garantindo ao final uma melhoria no desempenho.

A vantagem dos múltiplos fluxos aparece principalmente nas paginas *web*. Na mesma associação SCTP cada documento, página web pode ser transmitido por fluxos específicos, visto que esses documentos ou paginas são de diferentes tipos e tamanhos. Antes de entregar a camada de aplicação, as mensagens de determinado fluxo são ordenadas seguindo os campos de seqüência no fluxo. Importante ressaltar que o SCTP possibilita a não utilização desse recurso, o que se conceitua como mensagem urgente. Neste tipo de mensagem não está associado qualquer fluxo lógico, lembrando que essa opção de não ordenação é utilizada em sinalização telefônica na internet.

Os fluxos lógicos também não associam mensagens de controle, visto que se trata de mensagens pequenas e o seu reconhecimento é através de mensagens de controle específicas ou temporizadores.

3.6 - Mensagens SCTP

As mensagens de usuário e de controle são transmitidas em pacotes SCTP, onde cada pacote contém um cabeçalho comum, juntamente com uma ou mais mensagens. Lembrando que este pacote é passado para a camada de rede, devido ao desejo de eficiência de comunicação, várias mensagens de controle e de usuários podem ser colocadas no mesmo pacote de acordo com o que o usuário escolhe.

Durante o processamento de um pacote SCTP, as mensagens que estão dentro do pacote são quebradas na ordem que foram recebidas, assim, desta maneira algumas mensagens podem aparecer antes de outras. No processo de congestionamento, por exemplo, o SCTP pode agrupar em um único pacote várias mensagens sem o desejo dos usuários, ressaltando que mensagens de controle não podem sofrer esse processo de agrupamento. As mensagens mais conhecidas são:

- INIT

- INIT ACK
- SHUTDOWN COMPLETE.

A figura 6 mostra o cabeçalho SCTP, aonde cada campo somando neste cabeçalho chega ao valor de 12 octetos e esses campos são comuns a todas as mensagens de controle e usuários SCTP. Dividindo este cabeçalho percebe-se que os 4 primeiros octetos são reservados para porta de origem e destino aonde valores adicionados a esses campos são utilizados no processo de múltiplos fluxos, processo também utilizado no TCP. Para a identificação da associação, utilizam-se estas portas e os endereços IP de origem e destino.

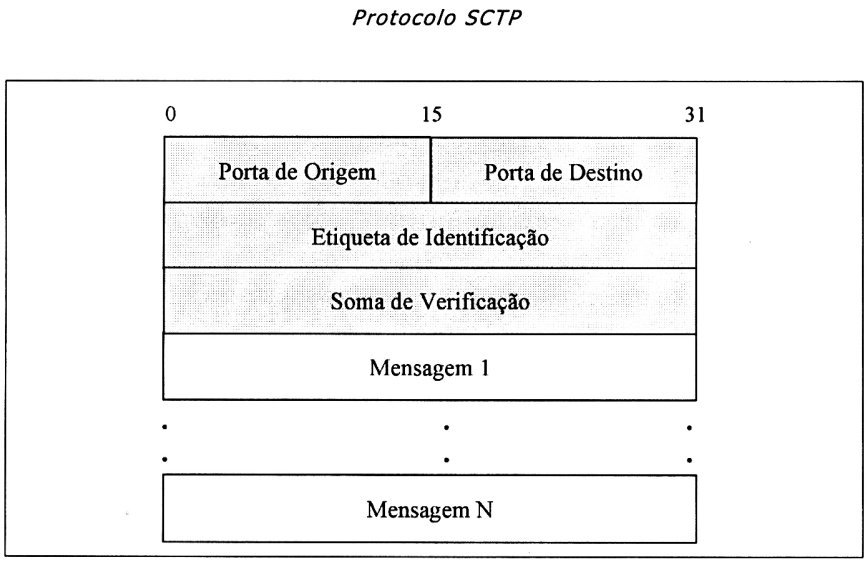

Figura 6: Formato dos Pacotes (COSTA, 2005)

Outro campo em destaque é a etiqueta de identificação, aonde se encontra este campo em todas as áreas de dados dos datagramas IP cuja finalidade é verificar a autenticidade do pacote confirmando o seu emissor. Para permitir a validação, realiza-se uma troca de valores de identificação que são criadas pelos terminais que estabelecem a associação e esses valores devem ser inalterados durante todo o processo de comunicação não havendo qualquer atualização deles.
O campo etiqueta possui dois valores, um para cada direção da associação. Acontece um cálculo de etiqueta feita por cada estação e informada à estação par evitando-se assim que seja de

conhecimento geral o valor criado da etiqueta e não permitindo o envio de mensagens de fora desta associação para dentro dela.

Outro campo interessante é o de soma de verificação de tamanho quatro octetos, maior que o do TCP que é de dois octetos, permitindo assim um algoritmo mais robusto e fortalecendo o controle de erro.

As mensagens SCTP são colocadas após o cabeçalho, assim como o cabeçalho segue um tamanho e formato possuindo uma parte de tamanho 4 octetos e uma parte variável. Nesta última parte, existem vários parâmetros que estão podem ser obrigatórios e opcionais.

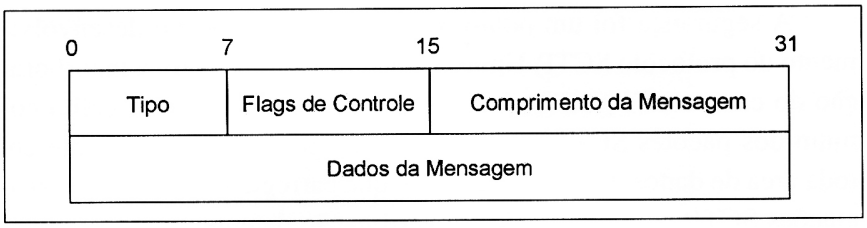

Figura 7: Formato das mensagens (COSTA, 2005)

Devido à grande variedade de tipos de mensagens, criou-se o campo tipo de mensagem com tamanho de um octeto e podendo possuir valores de 0 a 254. As mensagens com o valor zero para este campo são mensagens de usuário. Para mensagens com valor entre 1 e 14 são mensagens de controle. O campo *flags* de controle de um octeto permite um controle adicional às mensagens utilizando para isso seus bits como indicadores de alguma opção. O campo comprimento de mensagem, como o próprio nome já diz, determina o tamanho da mensagem incluindo os quatro octetos do cabeçalho padrão. Cada mensagem de quatro octetos inalterados e dois que indicam o tamanho da mensagem, permitindo que as mensagens sejam analisadas separadamente.

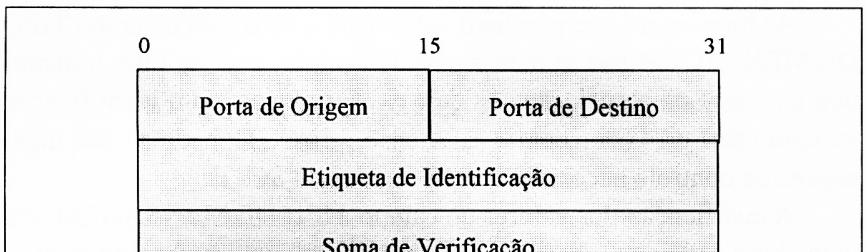

Figura 8: Exemplo de encapsulamento de mensagens (COSTA, 2005)

3.7 – Associações internas

A associação SCTP é gerada no processo *four-way handshake*, onde a alocação de recursos pelo servidor é realizada somente após a troca da terceira mensagem *handshake*, com a sua validação feita corretamente. Assim a parte de segurança é colocada em questão no TCP por não realizar esse processo que o SCTP faz, que assegura que a abertura da associação foi feita da parte certa.

O TCP por realizar a troca de apenas três mensagens acaba sendo mais vulnerável. A associação começar com um terminal enviando um pedido init e espera por uma mensagem *init ack*, a associação será estabelecida corretamente se os terminais usarem todos os recursos.

Assim, depois que o envio e recebimento da quarta mensagem é executa, a associação está corretamente estabelecida. As mensagens de controle *cookie* echo e *cookie ack* podem estar no mesmo pacote que mensagem de dados. Importante ressaltar que o não recebimento do *cookie ack* faz com que as mensagens de dados dos usuários sejam descartadas. O terminal que está como servidor, possui o estado closed quando se inicia o processo de estabelecimento da associação, o processo inicia-se após o terminal receber o *init*. Após receber esta mensagem, começa a troca de mensagens de controle até que se estabeleça a conexão fazendo com que uma

tabela hash seja criada armazenando essas variáveis de uma forma segura. O *cookie* acaba armazenando todos esses valores, juntamente com o MAC.

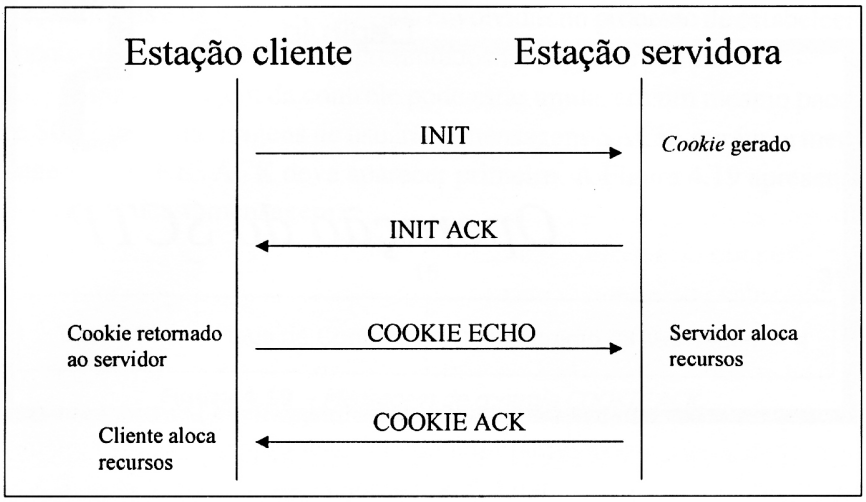

Figura 9: Estabelecendo comunicação (COSTA, 2005)

O papel do temporizador é importante neste processo de inicio de associação, pois realiza o processo de retransmissão caso não o terminal que enviou uma mensagem não receba a resposta. Caso a mensagem enviada não receba uma resposta, será descartada. Para que o processo de retransmissão não seja exaustivo, o SCTP determina um número aleatório que se for alcançado faz com que o endereço primário seja descartado e um dos endereços alternativos seja utilizado. Após receber o cookie echo, o terminal verifica o MAC para valida a mensagem recebida, assim assegura-se que quem enviou o cookie é realmente o outro terminal da associação. Caso a validação esteja correta é enviada uma mensagem cookie ack entrando no estado established. A associação não está completamente pronta, pois o terminal que enviou a primeira mensagem de controle ainda não aloucou os recursos necessários. Os estados *closed* e *established* são comuns aos dois terminais. A segurança na troca de mensagens faz com que no protocolo SCTP a troca de mensagens de controle sejam feitas sempre para o endereço primário. O terminal pode desejar

o encerrar a associação mesmo sem estar no estado closed. O encerramento através de uma mensagem *shutdown*.

O terminal que deseja terminar a associação recebe a confirmação das mensagens enviadas sem enviar mais alguma e envia um *shutdown*. O terminal que recebe o shutdown confirma com um shutdown *ack* após receber a confirmação das mensagens enviadas. Assim a estação que recebeu o *shutdown* ack responde com o shutdown complete fazendo com que o terminal entre no estado closed. O terminal que recebeu a mensagem shutdown complete, apaga as variáveis alocadas entrando no estado closed também. Entretanto, existe a mensagem abort que faz com que o terminal que envia esta mensagem libere todos os recursos e o lado que recebe apenas encerra a associação.

4 – PROTOCOLO XTP

XTP *(XPress Transport Protocol)* é um protocolo confiável, funciona em tempo real, atua na camada de transporte e foi desenvolvido por um grupo de pesquisadores e desenvolvedores coordenados pelo Protocol Engines Incorporated (PEI) (WEAVER, 2007). Atuais protocolos da camada de transporte como TCP, não foram criados para a próxima geração com alta velocidade de transmissão interconectada a redes confiáveis como FDDI e de largura de banda gigabit (WEAVER,2007). Ao contrário de todos os protocolos da camada de transporte, percebe-se que o XTP foi desenvolvido para atuar sobre um *chip set* VLSI. De uma maneira mais simples, o protocolo combina a camada de transporte com a camada de rede e utiliza a alta velocidade alcançada possível com uma implementação VLSI. XTP é capaz de fornecer transmissão de dados end-to-end em alta velocidade sem comprometer confiabilidade e funcionalidade.

As redes de computadores estão se caracterizando pela alta confiabilidade e pelas elevadas taxas de transmissão de dados, tradicionais protocolos da camada de transporte, como o TCP, no qual foi desenvolvido com pouca confiança na hora de atuar em redes interconectadas, podem ser fracos para ambientes emergentes. Embora eles contenham muitas características importantes, tais como detecção a erro, a retransmissão de dados, acabam sendo deficientes em alguns aspectos, eles não possuem uma taxa de controle e retransmissão seletiva, seus cabeçalhos são extensos e complexos devido ao seu tamanho. As taxas de transmissão acabam não sendo tão validas e limitam a escabilidade do protocolo.

XTP fornece uma transmissão de dados confiáveis em um ambiente de redes interconectadas, com um processamento em tempo real do protocolo, ou seja, o tempo de processamento dos pacotes enviados ou recebidos não é maior do que o tempo de transmissão. XTP contém controle de erros, fluxo de transmissão semelhante aos encontrados na camada de transporte com a adição da capacidade de *multicast.* O tempo de gerência é minimizado, visto que no protocolo só existe um temporizador que é utilizado no fechamento da conexão (WEAVER, 2007). XTP é desenvolvido para trabalhar nas máquinas em paralelo, pois o controle de erros, controle da taxa de transmissão, criação do contexto podem ser executados em paralelo. Considera-se o protocolo XTP como sendo um protocolo leve por várias razões, entre elas por possuir um algoritmo

simples e flexível, o cabeçalho dos pacotes é limitado e possui informações suficientes para validar e orientar os mesmos. O núcleo do protocolo contém 4 campos de tamanho fixo tais como:

- Key
- Route
- Seq
- Word

4.1 - Tipos de PDU

XTP utiliza dois formatos de quadros, um para controle de pacotes e outro para informações de pacotes.

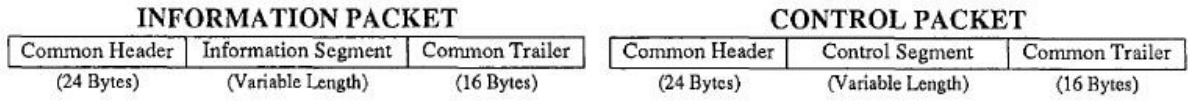

Figura 10: Formatação dos quadros (WEAVER, 2007)

Ambos os quadros, compartilham o campo *header* e o campo *trailer*, cada um com tamanho constante. Cada pacote XTP contém um segmento de tamanho variável entre o *header* e *trailer*, cujo tipo de segmento determina o tipo de pacote. Os campos mais importantes estão dentro de 8 bytes para poderem ser acessados rapidamente. O campo *header* especifica o tipo de pacote e especifica qual parte do fluxo de dados está incluso o segmento de informação da mensagem. Alguns modos opcionais, como a validação da desativação de erros ou transmissão multicast é indicada no cabeçalho do pacote no campo *control flags*. O segmento *trailer* possui dois campos *checksum* que identifica quantos dados foram entregues na aplicação cliente que está recebendo e também contém uma flag. Estas flags geralmente controlam a mudança dos estados, por exemplo, o fechamento de uma conexão de transmissão de dados ou requisição de dados de confirmação. O segmento *information* contém dados do usuário a serem transmitidos e é usado para passar endereços e outro dado qualquer quando solicitado. Cada pacote contém um

subconjunto de dados a serem transferidos. No XTP não há qualquer restrição quanto ao limite de número de bytes incluídos em cada pacote de dados. Para cada aplicação, este limite é conhecido como maximun transmission unit (MTU).

O segmento *control*, contem valores dos parâmetros de controle a erros, fluxo e taxas. Este segmento contém também campos usados para sincronizar uma retransmissão e recebimento quando necessário (WEAVER, 2007).

4.2 - Multi-Packet Handshaking

A seqüência de troca de múltiplos pacotes no XTP fornece aos usuários das aplicações um circuito virtual na camada de transporte e um serviço de datagramas também na camada de transporte. A conexão deverá ocorrer com a troca de 3 pacotes.
A seqüência de troca de múltiplos pacotes no XTP fornece aos usuários das aplicações um circuito virtual na camada de transporte e um serviço de datagramas também na camada de transporte. A conexão deverá ocorrer com a troca de 3 pacotes.

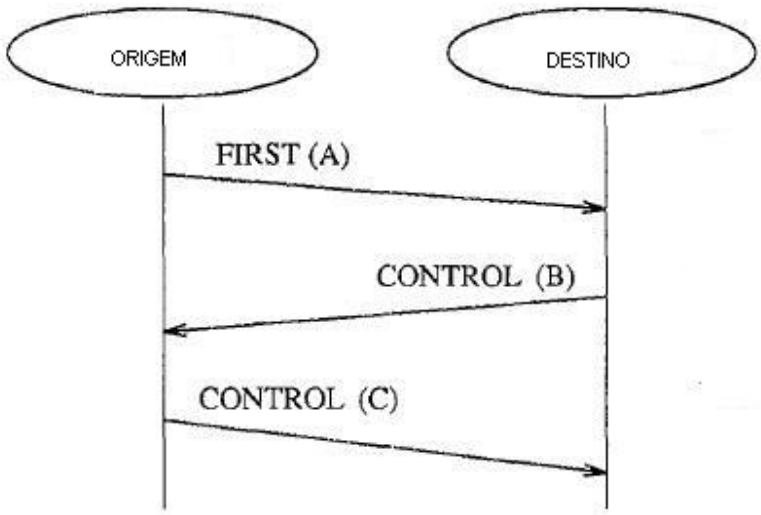

Figura 11: Modo de conexão Handshake (WEAVER, 2007)

A figura 11 mostra como o XTP pode realizar uma conexão confiável com a troca de apenas 3 pacotes, o terminal inicia a transmissão e o destino recebe e verifica que o terminal que iniciou a transmissão pretende realizar uma conexão. Caso o destino concorde é estabelecida uma conexão. Assim os pacotes são colocados numa fila para esperar que comece a transmissão.

Segundo (WEAVER, 2007) no XTP a responsabilidade em detectar quem perdeu o *ack* é assumida pela remetente. O remetente pede o *ack* pelo bit *status request* dentro do cabeçalho do trailer. Um temporizador (*wtimer*) deve ser utilizado pelo remetente para determinar se o receptor não conseguiu responder a um request do remetente. Se o temporizador expirar antes que o *ack* chegar, o remetente assume que o *ack* foi perdido, e envia um request para um pacote

de controle do ack recebido. Por outro lado, quando terminar, a origem confirma a finalização do contexto, a fim de que o receptor possa ter certeza que o contexto está fechado. Caso este último pacote fique danificado ou perdido, o receptor acaba fechando a conexão.

Ao contrário do TCP, onde cada pacote de dados seria retransmitido após o timeout, no XTP apenas um pacote CNTL contendo o SREQ será enviado. O destino retornará o pacote CNTL que dados os dados dos pacotes, se existirem, serão retransmitidos. Ao utilizar retransmissão seletiva, XTP evita retransmitir dados, que já foi recebido corretamente.

A flag END é enviada no pacote final para informar que o destino fechará ou encerrará a sua conexão, garantindo assim que nenhum outro pacote poderá ser transmitido. Note que quando um dado de confirmação é solicitado no XTP, como no primeiro pacote da Figura 2 (pacote A), a confirmação não é necessariamente fornecida imediatamente. Em alguns casos, o receptor atrasa a confirmação até que todos os dados recebidos antes da SREQ sejam processados. Incluindo também os dados do pacote com SREQ.

4.3 - Controle a erros

Quando acontecem erros na transmissão, XTP deve detectar os erros e começar o processo de retransmissão dos dados com problemas. XTP utiliza dois checksums do pacote para verificar a integridade dos pacotes recebidos pela rede.

Prefere-se colocar os checksums nos últimos bytes do frame XTP de modo a que o cálculo checksum possa estar de acordo com os pacotes de transmissão ou recepção. Se os checksums forem colocados no inicio do pacote, todo o pacote terá de ser acessado para calcular o checksum antes que comece a transmissão do mesmo. Assim, existirão duas varreduras sobre os dados, uma para o checksum, e uma copiar os bytes para a rede (WEAVER, 2007).

Quando cada checksum indica que o pacote recebido contém informações erradas, o receptor assume o pacote está ilegível e descarta. Caso a origem seja conhecida, o receptor poderá

informar imediatamente à origem que o pacote foi corrompido na transmissão permitindo que a origem começar a retransmissão.

Geralmente, esta informação está disponível no campo *KEY* que identifica exclusivamente o processo originário cliente no nó que transmitiu o pacote. Contudo, o destino não pode supor que o campo *KEY* está correto, uma vez que o erro pode ter ocorrido em qualquer lugar dentro do pacote incluindo o próprio campo *key*. Assim, o receptor sempre descarta pacotes com erros.

4.4 - Mecanismos de endereçamento

XTP foi desenvolvido para interagir com várias camadas de rede. Em cada caso, os pacotes são encapsulados dentro de PDUs da camada subjacente. Dentro da camada de XTP, cada fim de uma conexão, XTP deve ser capaz de identificar seus pares.

Ao invés de incluir todos os dados relevantes de endereçamento em cada pacote, o cachê do XTP endereça dados contidos no primeiro pacote tanto no terminal que envia quanto no que recebe e usa o campo *key* como um índice dentro do cachê para acessar os endereços conforme necessário. Assim, este primeiro pacote é uma informação especial. Os próximos pacotes contêm somente o campo *key*, resultando em pacotes menores porque o campo *key* é codificado em poucos bytes. O campo *key* é incluído em cada pacote.

O *key* é gerado pelo nó de inicialização da conexão e inclui o primeiro pacote transmitido ao terminal destino. Incluídos neste primeiro pacote, os dados de endereçamento usados para identificar o destinatário. Estes endereços estão contidos em uma lista para ser comparada com o filtro de endereçamento do terminal destino, no modo multicast, mas que um destinatário está direcionado para cada pacote. O destinatário apropriado percebe a chegada do primeiro pacote e salva o identificador *key*, o MAC e também o VIA em um banco de dados associado com o registro. Os próximos pacotes não precisam conter o destino na rede, contudo que a tripla (*key*, *mav, via*) possa ser consultada.

4.5 - Multicast

XTP define um modo multicast de operação aonde um terminal de origem pode realizar um broadcast da mesma stream ou um datagrama seqüencial para múltiplos terminais destino simultaneamente.

O modo multicast do XTP é similar na operação para um simples modo receptor em muitos aspectos. A questão da transmissão do primeiro pacote e subseqüentes pacotes de dados. SREQ é usado para solicitar pacotes CNTL. O controle de erro é suportado usando a retransmissão go-back; a retransmissão seletiva não é suportada. As conexões multicast e alocação de espaço no buffer e cada terminal destino poderem variar de tamanhos. Essencialmente, a transmissão de dados procede ao ritmo mais lento ao terminal destino.

Como o multicast envolve um grande número de terminais destino, o terminal origem deverá enviar sinalizar o pacote rejeitado para todos os terminais destino. Para atenuar, este efeito, os terminais destino requerem atualizar-se do envio do pacote rejeitado por multicast quando cientes que o terminal origem foi devidamente notificado. Os terminais destino monitoram a rede para que outros pacotes rejeitados durante o tempo em que o pacote está sendo preparado e aguardado para transmissão. Caso outro pacote rejeitado chegar, com destino para o terminal origem no mesmo multicast, o terminal destino compara a sua RSEQ com o pacote recém chegado. O campo RSEQ é significativo, pois é o próximo byte que os terminais destino por multicast deverão aceitar (WEAVER, 2007).

4.6 - Ordenação de bytes

Uma característica interessante dos pacotes que usam XTP diz respeito à ordenação que os bytes estão organizados em uma palavra para vários computadores. Esta ordenação afeta a seqüência em que os bytes são colocados na rede. Bytes dentro de uma palavra podem ser organizados do mais alto para o mais baixo endereço, ou do mais baixo para o mais alto endereço. Equipamentos de diferentes fabricantes suportam diferentes ordenações de bytes. Desde que não existe padronização, XTP fornece uma maneira natural para suportar ambas as ordenações.

O problema é codificar dentro de cada pacote uma indicação de qual byte ordenado foi usado pelo terminal origem para preparar o pacote, e de tal forma que o terminal destino adere para cada byte ordenado poder determinar a correta ordenação dos bytes. Isto foi resolvido no XTP usando duas flags, A posição das duas flags foi escolhida para que eles mapeiem dentro de si mesmo se a ordenação veio incorreta.

5 – ESTUDO COMPARATIVO ENTRE PROTOCOLOS TCP, SCTP e XTP

Após realizar um estudo dos protocolos TCP, SCTP e XTP existe a necessidade de fazer um breve comparativo com suas vantagens e características únicas a cada protocolo. Junto a essa análise, é possível juntar informações e baseados nos testes definir qual protocolo a ser usado para esse cenário, principalmente quando se fala em aplicações de alto desempenho aonde a transmissão dos dados, recuperação de pacotes, confiabilidade são fundamentais para o seu processo.

Tabela 1: Tabela comparativa entre protocolos TCP, SCTP, XTP e UDP.

Característica	**SCTP**	**TCP**	**XTP**	**UDP**
Confiabilidade	sim	sim	sim	não
Orientado a Conexão	sim	sim	sim	não
Orientação da Transmissão	msg	octeto	msg	msg
Controle de fluxo	sim	sim	não	não
Controle de congestionamento	sim	sim	sim	não
Tolerância a falhas	sim	não	sim	não
Distribuição de dados	Parcal. Ordenada	ordenada	ordenada	desorden.
Múltiplos fluxos	sim	não	não	Não
Full duplex	Sim	Sim	Sim	Sim
Entrega ordenada de dados	Sim	Sim	Sim	Não
ACKs seletivos	Sim	Opcional	Sim	Não
Pseudo-header para Checksum	Não	Sim	Sim	sim
Permite conexões half-closed	Sim	Sim	Sim	não
Proteção contra ataques SYN flooding	Sim	Não	Não	Não

5.1 – Descrição do Ambiente

Um switch que receberá dados GPRS na rede da operadora será uma das máquinas a serem analisas visando encontrar perdas, latência e demais parâmetros de desempenho dos protocolos.

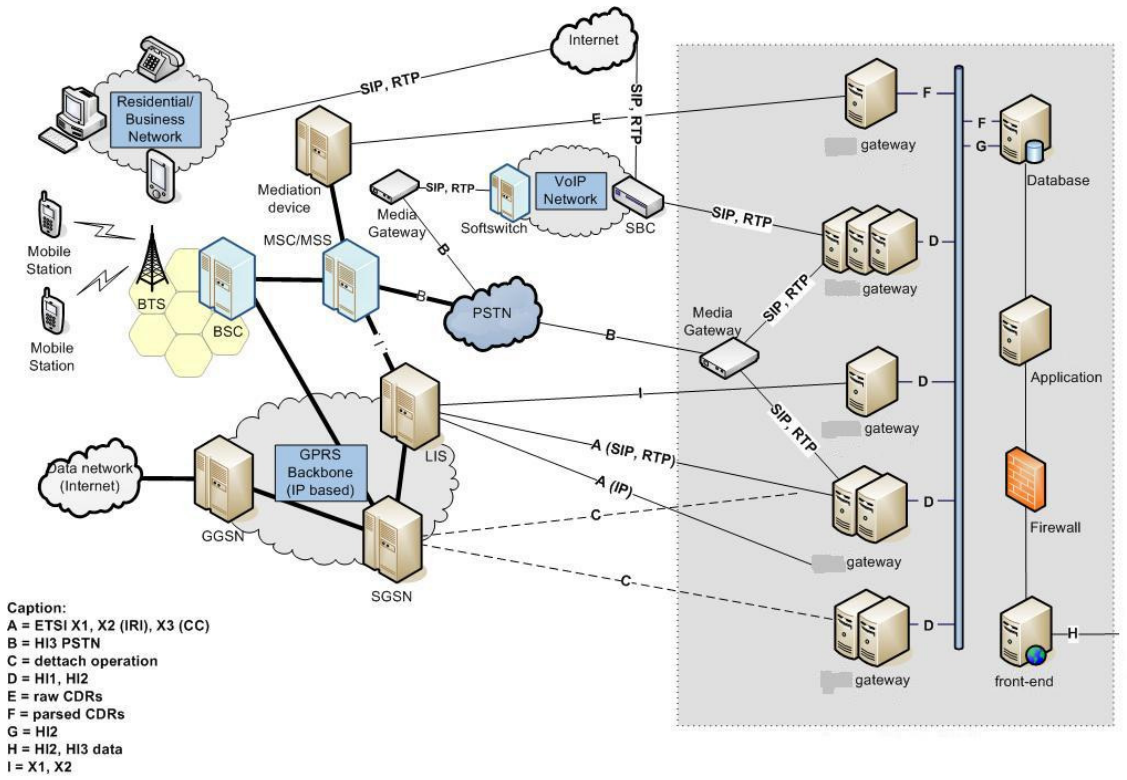

Figura 12: Rede GPRS integrada cenário de testes

Pela figura 12 se percebe algumas máquinas que recebem o trafego GPRS e repassam para as demais máquinas que se localizam em outras redes. Assim, os testes possibilitam uma comparação entre as redes, roteadores e switchs nos quais os dados GPRS passam.

Na figura 13, temos uma visão das máquinas a serem analisadas após o recebimento dos dados GPRS do switch principal. Notam-se as diferentes redes que se localizam as máquinas.

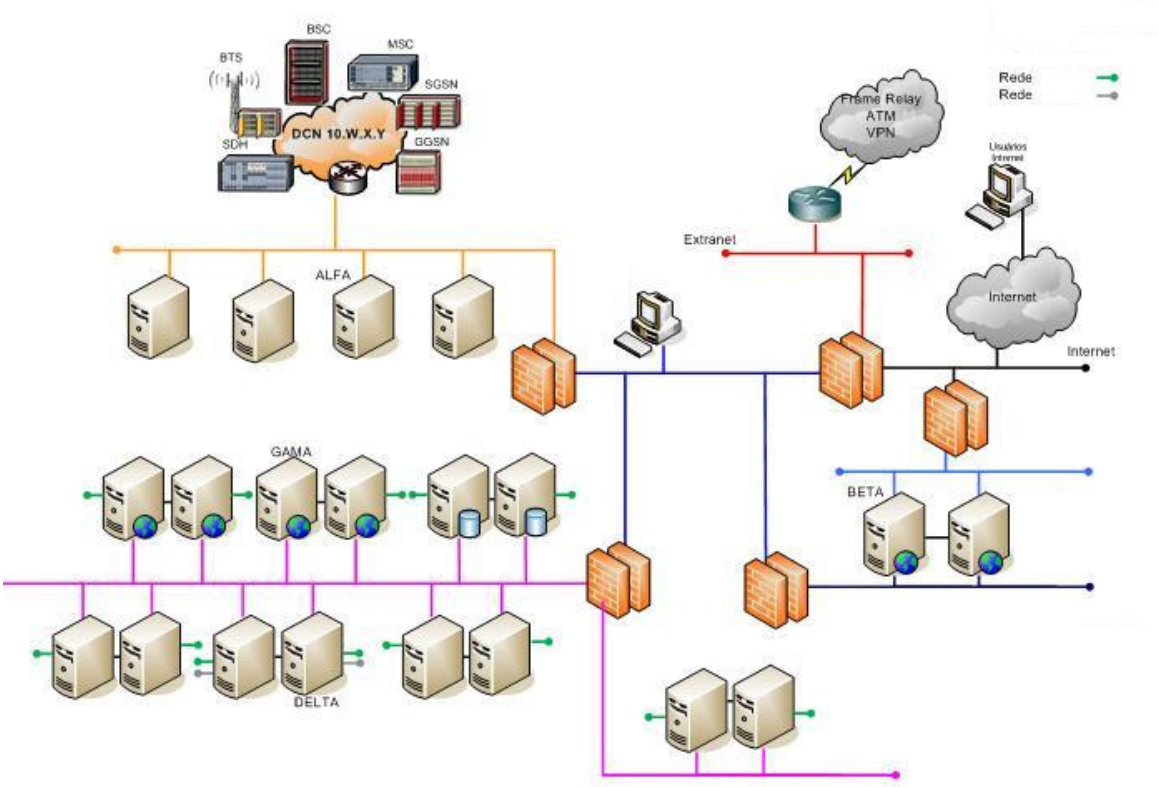

Figura 13: Visão interna do cenário de testes

Para testes deste nível, em operadoras de telefonia móvel, a disponibilização das máquinas devem ser em redes separadas, principalmente para testes de captura, vazão, latência e perda de pacotes. As comunicações entre máquinas que contenham algum firewall entre elas tiveram as regras solicitadas à equipe de rede e segurança da operadora para que os testes fossem gerados e analises fossem feitas.

5.1.1 - Descrição das máquinas

As máquinas utilizadas para os testes de latência, vazão, etc. fazem parte da plataforma que se encontra na operadora.

Os nomes são fictícios para evitar comparações com o ambiente real, mas todas as informações técnicas são verdadeiras e baseadas nelas que os testes obtiveram os resultados descritos posteriormente.

- Máquina ALFA

- Processador Intel Xeon 5130 dual core de 2.0 Ghz
- 4 GB de memória DDR-2
- 2 interfaces de rede 10/100/1000
- Placa de rede Intel 1000PT Dual Port PCI-e
- Red Hat Enterprise Linux ES 4.0

- Máquina BETA

- Processador Intel Xeon 5130 dual core de 2.0 Ghz
- 4 GB de memória DDR-2
- 2 interfaces de rede 10/100/1000
- Placa de rede Intel 1000PT Dual Port PCI-e
- Red Hat Enterprise Linux ES 4.0

- Máquina GAMA

- Processador Intel Xeon 5130 dual core de 2.0 Ghz
- 4 GB de memória DDR-2
- 2 interfaces de rede 10/100/1000
- Placa de rede Intel 1000PT Dual Port PCI-e
- Red Hat Enterprise Linux ES 4.0

- Máquina DELTA

- 2 processadores Intel Xeon 7140M Dual Core de 3.4 GHz com 2 MB de cache L2 (2x1MB) e 16 MB de cache L3 (FSB 800MHz)
- Processadores com tecnologia EM64T
- 8 GB de memória DDR-2 400 MHz (8x1GB)
- 2 placas de rede 10/100/1000 integradas
- Placa de rede Intel 100 MT Dual Port PCI
- Red Hat Enterprise Linux AS 4.0

5.2 - Cenário de Rede

5.2.1 - Rede Loopback

Utilizou-se cada um dos computadores para gerar testes de vazão e latência na mesma rede com tamanho de mensagens variáveis.

A máquina ALFA, BETA e GAMA foram utilizadas para esses testes.

5.2.2 - Rede Ethernet 100Mbps

Foi utilizada uma rede Ethernet 100Mbps, sem a introdução de nenhuma perda ou latência artificial. Os testes variaram o tamanho das mensagens trocadas entre os participantes, de 1 a 23246 *bytes*.

A máquina ALFA foi utilizada como servidor e a máquina BETA como cliente.

5.2.3 - Aplicativos construídos para o teste

Os aplicativos utilizados nos testes deste capítulo foram escritos por (PFÜTZENREUTER, 2004), nas linguagens C e C++.

Os aplicativos descritos por (PFÜTZENREUTER, 2004) permitem variar o tamanho das mensagens transmitidas, 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".

Segundo (PFÜTZENREUTER, 2004), por padrão, os aplicativos trafegam 100 Mbytes, em cada direção, por rodada.

5.2.4 - Testes de vazão

Os aplicativos construídos para o teste de vazão enviam certo volume de *bytes*, divididos em mensagens individuais. O envio acontece nas duas direções; se o volume total configurado for e.g. 10 MBytes, o volume total trafegado em ambas as direções será 20MBytes. Os aplicativos não tentam sincronizar de qualquer forma os dois fluxos e é perfeitamente possível que uma direção termine antes que outra, prejudicando o resultado final do teste. Isto é propositado – um bom protocolo de transporte deve permitir uma comunicação *full-duplex* equilibrada, e o teste de vazão testa (indiretamente) esse equilíbrio.

5.2.5 - Lista de utilitários de teste de vazão

tcp_server: Servidor que utiliza soquetes estilo TCP. Permite usar os protocolos TCP e SCTP.

tcp_client: Lado cliente do teste, com as mesmas características de *tcp_server*.

Os testes de vazão podem ser executados tanto com TCP como com SCTP. Nos cenários de *loopback* e rede *100Mbps*, ambas as possibilidades foram utilizadas para efeitos de comparação.

5.2.6 - Testes de latência

Os aplicativos para teste de latência, da mesma forma que os testes de vazão, trafegam um dado volume em ambas as direções, dividido em mensagens de tamanho fixo. A diferença é que, 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.

A chamada *gettimeofday()* tem precisão de segundos em computadores e sistemas suficientemente recentes. O tempo consumido pela chamada foi considerado negligenciável frente às latências a serem mensuradas.

Os aplicativos criados para o teste foram: *tcp_server_lat* e *tcp_client_lat*. Eles compartilham das mesmas características dos aplicativos correspondentes para medida de vazão.

5.2.7 - Sistema operacional e implementação do SCTP

O sistema operacional utilizado foi o Linux, versão de *kernel* 2.6.9-55. Foi o *kernel* estável mais recente disponível à época dos testes. O protocolo SCTP sofreu atualizações importantes até a versão 2.6.8, como o suporte a PR–SCTP. A implementação do SCTP utilizada é a LK–SCTP *(Linux Kernel SCTP)*, incluída nas séries 2.4 e 2.6 do *kernel* do Linux. Exceto onde explicitamente afirmado o contrário, o sistema operacional, o TCP e o LK–SCTP foram utilizados com suas configurações *default*, sem qualquer otimização de parâmetros. Presume-se que a maioria dos usuários use um sistema da mesma forma, sem qualquer otimização, e que uma configuração *default* sub-ótima seria simplesmente percebida como um problema inerente.

A importância dos aplicativos *user-level* é reduzida nos testes deste trabalho, pois o desempenho depende majoritariamente do *kernel*. O compilador C foi usado com a distribuição versão 3.4.

A implementação começou a ser desenvolvida em 2001 por (PFÜTZENREUTER, 2004), e já foi objeto de vários testes independentes. Possui inclusive sua própria biblioteca de testes de regressão, que verificam, a cada lançamento de nova versão, se todos os aspectos do protocolo estão funcionando corretamente. Porém, a maturidade será atingida apenas quando o protocolo for largamente usado por aplicações reais.

5.3. Resultados Experimentais

5.3.1 - Vazão em Interface de *loopback*

Os resultados através do gráfico e tabela abaixo acabam seguindo o mesmo principio que (PFÜTZENREUTER, 2004) declara, aonde o cálculo somatório CRC-32c e a separação de mensagens forçando a troca de contexto no destino para cada mensagem fazem com que SCTP fique abaixo de TCP.

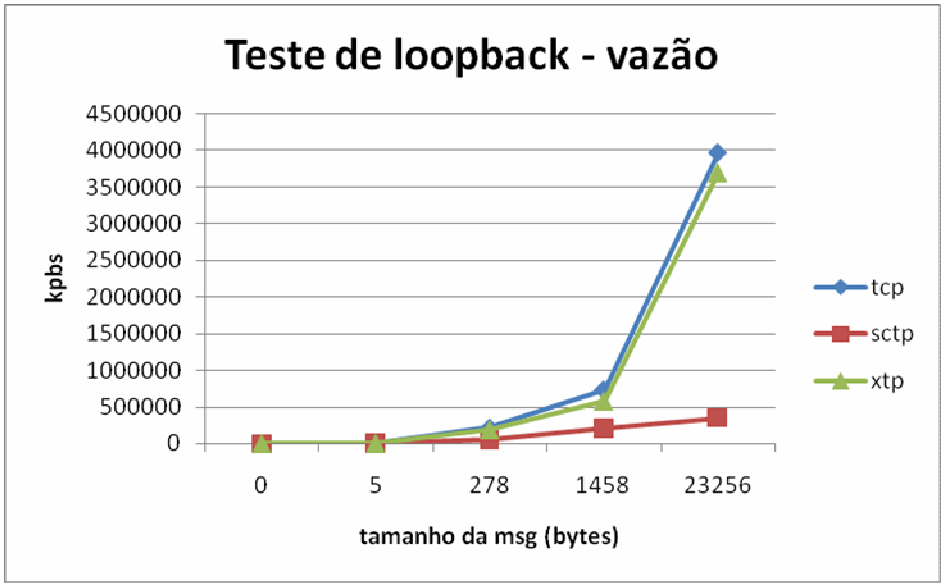

Figura 14: Teste de *loopback* – Vazão

Tabela 2: Tabela de *loopback* - Vazão

Protocolo / tam. da msg	0	5	278	1458	23256
tcp	0	4408	214008	732656	3970768
sctp	0	1024	58672	213744	351200
xtp	0	3965	189895	578785	3696548

Contudo quando os testes foram mais “nivelados”, fazendo com que TCP tenha a separação de mensagens, os resultados fizeram com que os protocolos tivessem seus resultados parecidos. O

teste de desligar o CRC-32c foi feito como sugestão por parte de (PFÜTZENREUTER, 2004) e comprovou-se um pequeno ganho. Assim, acredita-se que essa característica não influencia no desempenho de SCTP.

5.3.2 - Latência em Interface de *loopback*

O fato de SCTP realizar várias cópias em memória acaba prejudicando no seu desempenho com relação aos outros protocolos. XTP, neste caso teve um desempenho parecido com TCP o que nos faz entender que o desempenho para ambos os protocolos em cenários assim, não dá ganho a nenhum dos dois.

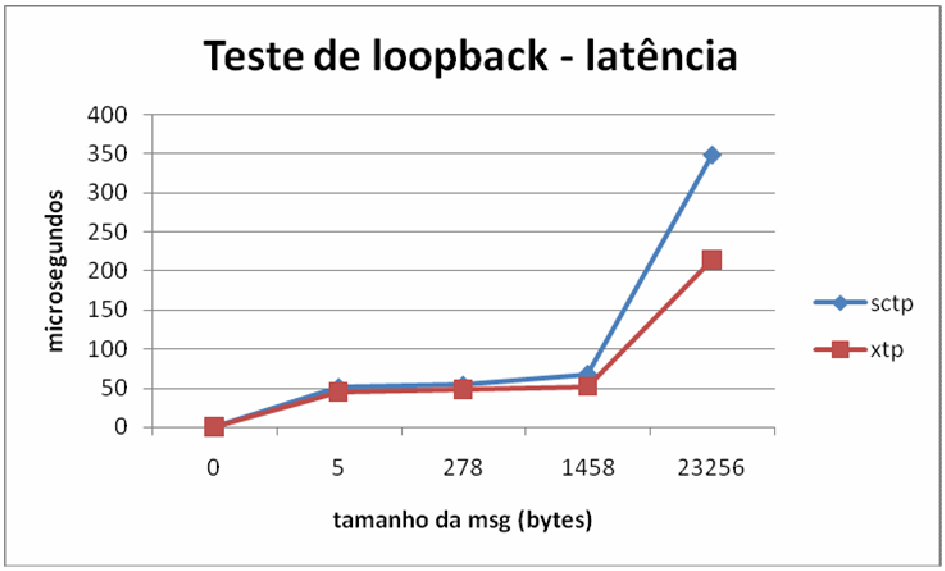

Figura 15: Teste de *loopback* – Latência

Tabela 3: Tabela de *loopback* – Latência (microsegundos)

Protocolo / tam. da msg	0	5	278	1458	23256
sctp	0	51	54	68	349
xtp	0	45	48	52	215

Conforme dados apresentados na Tabela 3 a variação no número de fluxos não apresentou ganho ao SCTP como se imaginava, as mensagens utilizadas foram de tamanho variado em bytes.

(KANG & FIELDS, 2003) avisam que o desempenho de SCTP é melhor quando múltiplos fluxos são criados, o que não foi comprovado neste teste. Este ganho está ligado ao fato de múltiplos fluxos são criados recebendo dados de forma desordenada, contrariando os testes com TCP que recebem dados desta maneira.

Para que os testes sejam justos, buscou-se considerar todas as principais características dos protocolos.

5.3.3 - Vazão rede 100 Mbps

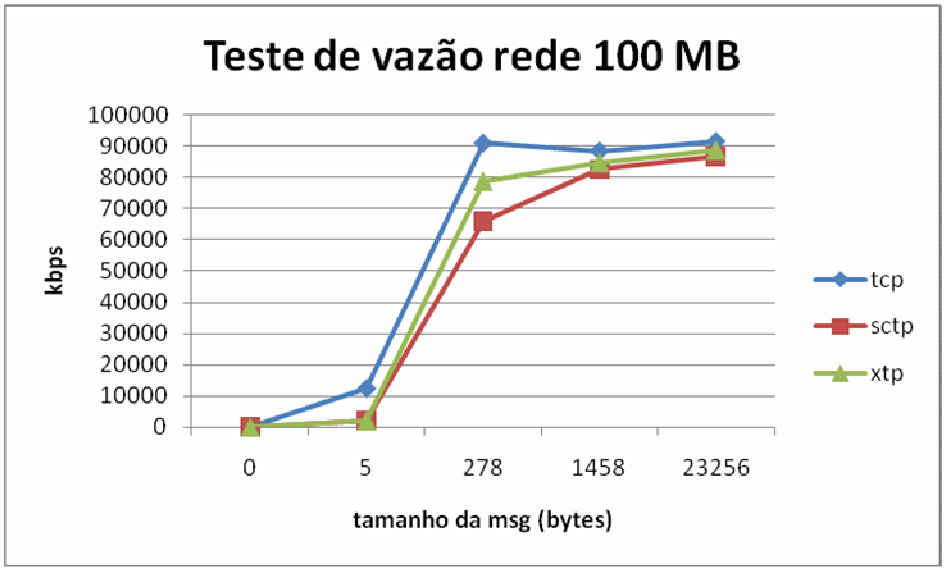

Figura 16: Teste de Rede 100Mbps

Tabela 4: Tabela de Rede 100Mbps

Protocolo / tam. da msg	0	5	278	1458	23256
tcp	0	12424	90832	88312	91392
sctp	0	2152	65920	82664	86864
xtp	0	2058	78585	84585	88585

Nos testes de vazão já utilizando uma rede de 100 Mbps, percebe-se uma melhor de SCTP quando o tamanho da mensagem aumenta. Para mensagens de 23256 bytes, SCTP praticamente tem a mesma vazão que XTP. Acredita-se ainda que LK-SCTP acomoda uma mensagem por datagrama, sendo que poderia estar acomodando várias, chama-se isso de *message bundling*.

5.3.4 - Latência rede 100Mbps

Pelos resultados da latência a seguir, a latência de SCTP é um pouco maior que TCP, fato que se deve aos mesmos motivos colocados nos testes de latência em loopback que naquele caso acabaram sendo mascarados pela latência da rede.

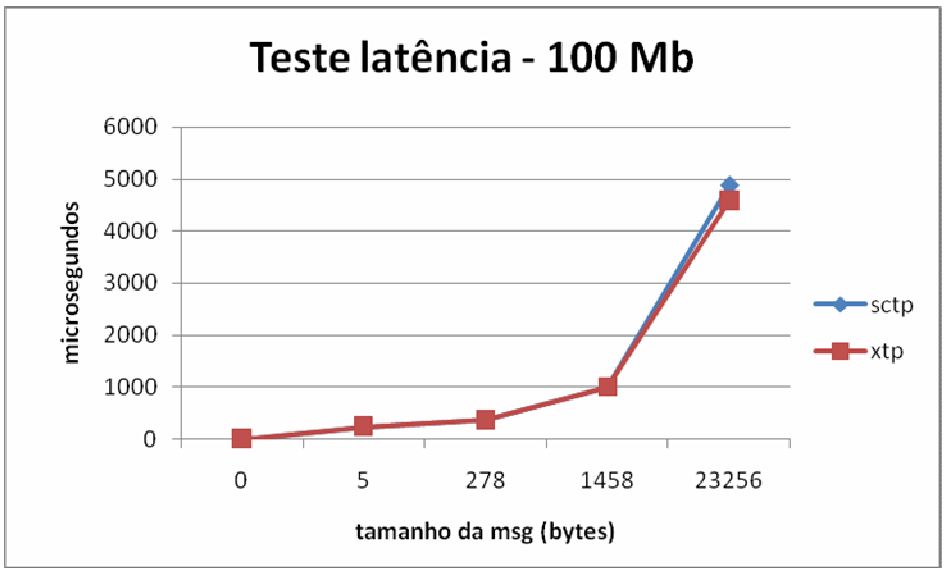

Figura 17: Teste de Rede 100Mbps - Latência

Tabela 5: Tabela de Rede 100Mbps - Latência

Protocolo / tam. da msg	0	5	278	1458	23256
sctp	0	247	372	996	4875
xtp	0	246	371	997	4582

5.3.5 - Vazão rede 100 Mbps – Vários fluxos

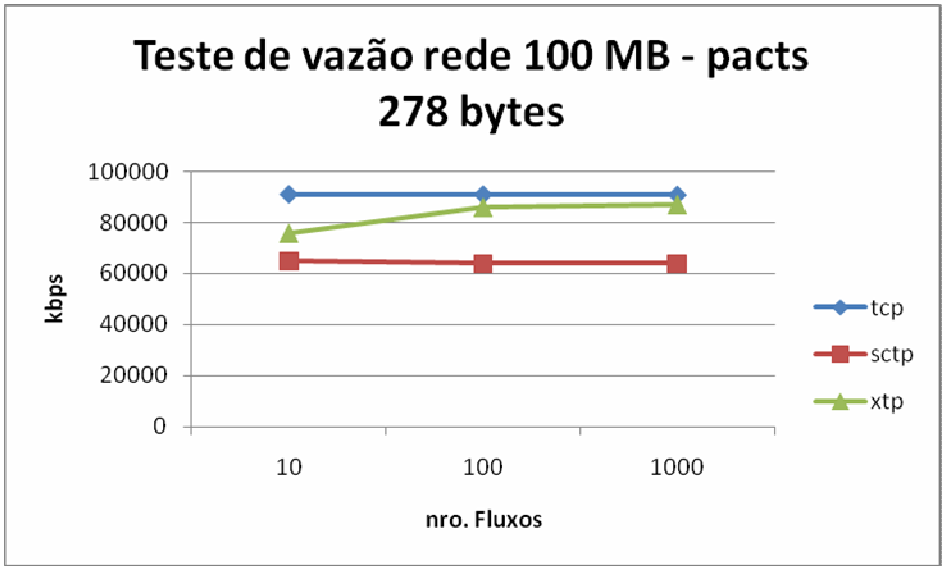

Figura 18: Teste de Rede 100Mbps - Vazão – Vários fluxos

Tabela 6: Tabela de Rede 100Mbps - Vazão – Vários fluxos

Protocolo / Nro. Fluxos	10	100	1000
tcp	91152	91008	90888
sctp	64896	63888	63848
xtp	75895	85856	86958

Os testes de vazão mostraram uma diferença considerável contra SCTP. Esperava-se que com o aumento de fluxos numa rede real com máquinas distintas a vazão fosse a favor de SCTP, mas deve-se levar em conta também a questão de segurança que SCTP abrange. Assim, mensagens com tamanho 257 bytes têm uma melhor vazão com TCP.

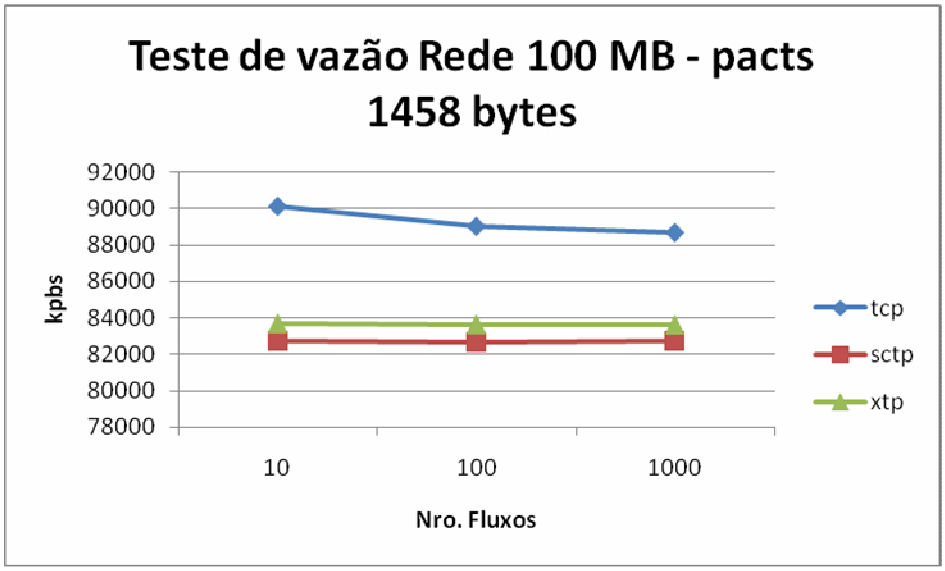

Figura 19: Teste de Rede 100Mbps - Vazão – Vários fluxos

Tabela 7: Tabela de Rede 100Mbps - Vazão – Vários fluxos

Protocolo / Nro. de Fluxos	10	100	1000
tcp	90104	89032	88696
sctp	82704	82664	82720
xtp	83695	83654	83625

A medida que o tamanho da mensagem aumenta, diminui a diferença entre a vazão de TCP e SCTP, devido aos controles de segurança que SCTP faz, o seu desempenho ainda é baixo, mas pelo fato de TCP realizar uma conexão única para o envio dos pacotes, esperava-se que pelo menos a mesma vazão que o teste anterior fosse mantida.

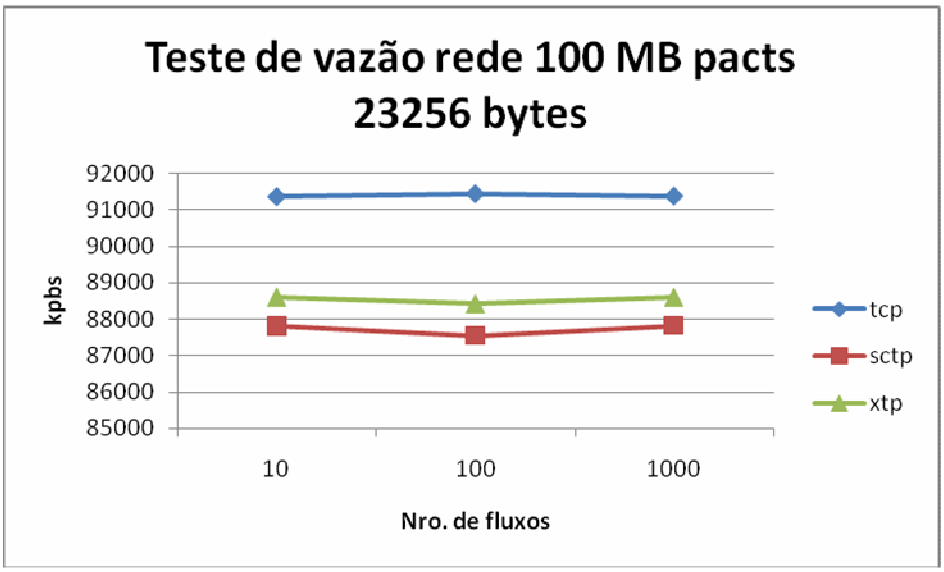

Figura 20: Teste de Rede 100Mbps - Vazão – Vários fluxos

Tabela 8: Tabela de Rede 100Mbps - Vazão – Vários fluxos

Protocolo / Nro. de fluxos	10	100	1000
tcp	91368	91448	91384
sctp	87808	87552	87816
xtp	88596	88412	88592

Com o ultimo teste de vazão para rede de 100 Mbps, enviou-se pacotes com tamanho de 23256 bytes, surpreendendo pois TCP e TCPM tiveram uma excelente vazão em comparação com os demais protocolos, reafirmando a teoria que a segurança nos outros protocolos prejudicou o desejo de um melhor desempenho

5.3.6 - Latência rede 100 Mbps – Vários fluxos

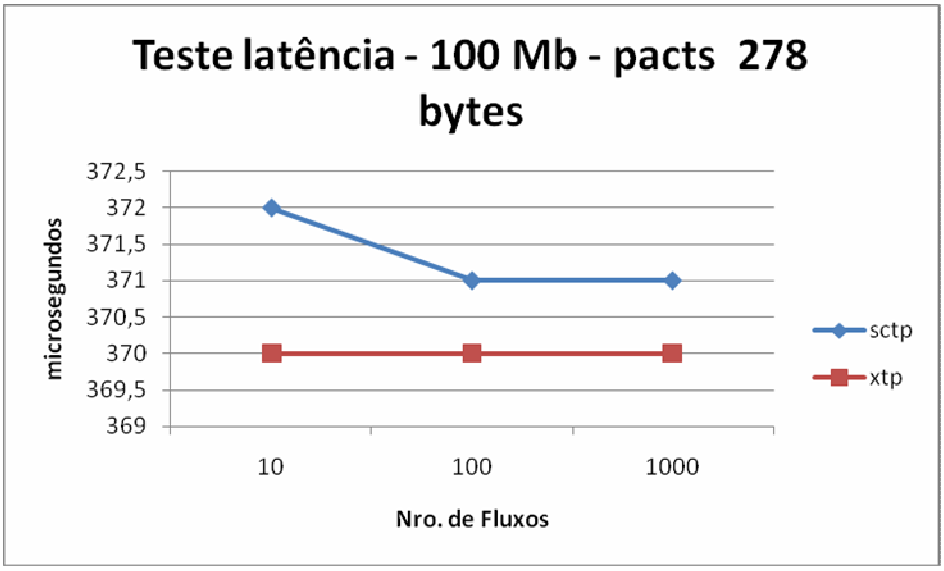

Figura 21: Teste de Rede 100Mbps - Vários fluxos

Tabela 9: Tabela de Rede 100Mbps - Vazão – Vários fluxos

Protocolo / Nro. Fluxos	10	100	1000
sctp	372	371	371
xtp	370	370	370

Para os testes de latência acima, aonde mensagens de tamanho 257 bytes foram enviados do cliente ao servidor, SCTP teve seu desempenho prejudicado até 100 fluxos. Acredita-se que por causa da abertura de várias conexões e o seu controle de segurança o desempenho pode ter sido influenciado. Até o momento XTP não apresentou ganhos e nem perdas com relação aos testes de latência realizados. Devido ao seu controle de erros e controle de segurança esperava-se mesmo que XTP tivesse o desempenho similar ao SCTP.

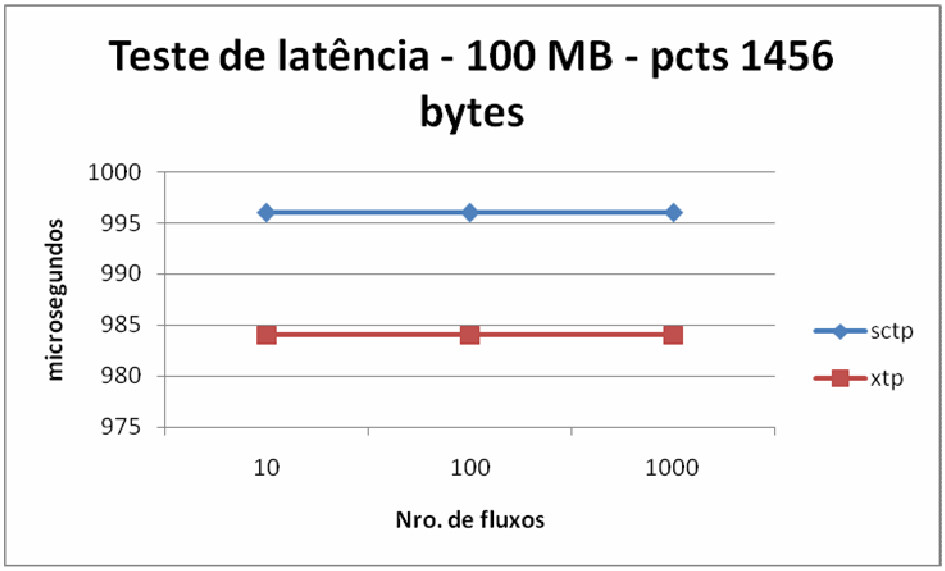

Figura 22: Teste de Rede 100Mbps - Latência - Vários fluxos

Tabela 10: Tabela de Rede 100Mbps – Latência – Vários fluxos

Protocolo / Nro. de fluxos	10	100	1000
sctp	996	996	996
xtp	984	984	984

Nos testes com mensagens um pouco maiores, 1426 bytes SCTP continua com o aumento de latência a medida que os fluxos aumentam, mas nesse teste o que mais chama a atenção é o fato de TCPM estar com o seu desempenho prejudicado mesmo com as modificações realizadas nele para tentar igualar-se ao SCTP. TCP continuou com a mesma latência, acredita-se que devido ao seu RTO não existir limites, adaptando-se a latência da rede.

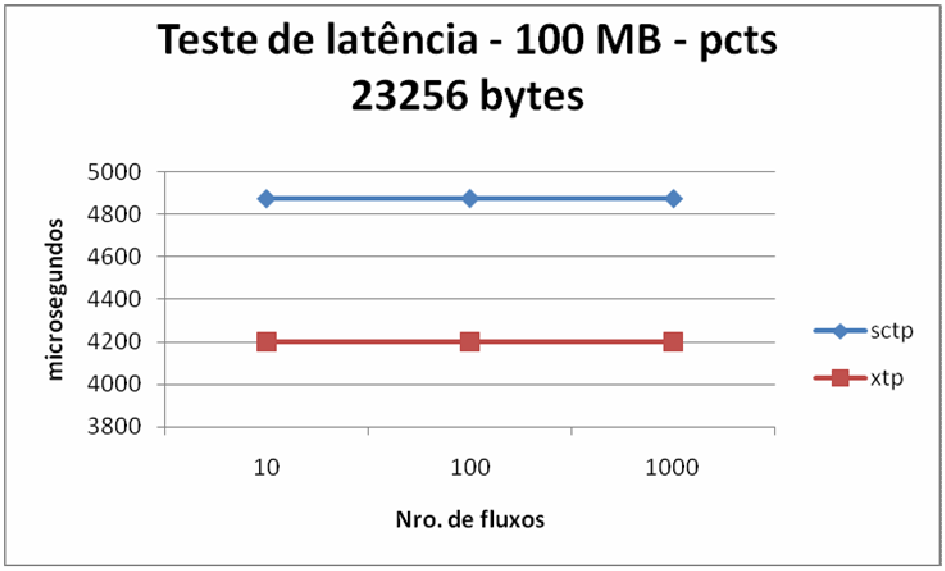

Figura 23: Teste de Rede 100Mbps - Latência - Vários fluxos

Tabela 11: Tabela de Rede 100Mbps – Latência – Vários fluxos

Protocolo / Nro. de fluxos	10	100	1000
sctp	4874	4875	4874
xtp	4200	4200	4200

Como teste final para latência em redes de 100 Mbps com vários fluxos, percebe-se que com mensagens de 23256 bytes o desempenho de SCTP é muito abaixo do que XTP e TCPM. Neste caso, pode-se levar em conta o trafego na rede nos momentos dos testes ter influenciado contra SCTP. Redes com uma latência mais elevada, tende a ter um RTO mínimo mais inadequado, fazendo com que o desempenho de SCTP fica mais próximo de TCP.

5.4 – Conclusões sobre os resultados

Os primeiros testes, de vazão comprovam o que era é de se esperar, o TCP teve um desempenho superior ao SCTP. Acredita-se que isso foi causado pelo cálculo do somatório CRC-32c que ocorre para cada pacote no protocolo SCTP, e também a compulsória separação de mensagens que faz com que ocorra uma troca de contexto no receptor para cada mensagem.

Como sugestão em outras literaturas realizou-se o teste de desligar o cálculo do CRC-32c no kernel do Linux, mostrando que o ganho de desempenho é relativamente pequeno. Como sugestões futuras acreditam-se num estudo mais cuidadoso, que a principio indica como o principal motivo pelo menor desempenho do protocolo SCTP a separação de mensagens que nele ocorre.

Continuando os testes, chega-se aos testes de latência, assim descreve-se que para as trocas de contexto têm de seguir uma ordem estrita: processo cliente, kernel, processo-servidor, kernel, processo-cliente. Acredita-se que todos os overheads (bytes de controle, custo de CPU, ineficiências da implementação) ficaram evidenciados. Pelos testes percebe-se o baixo desempenho de SCTP não importando o tamanho da mensagem.

O desempenho inferior do SCTP para esses tipos de testes, 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. Sabe-se através do estudo em cima do protocolo SCTP que devido as suas características isso não é possível.

Em (KANG & FIELDS, 2003) fala-se que a vazão do SCTP aumenta com a utilização de diversos fluxos, o que não comprova-se nesse trabalho. Contudo o teste realizado por eles tem como premissa que o protocolo de aplicação utiliza ao mesmo tempo todos os fluxos, e pode receber as mensagens de forma desordenada. Caso só possível se fosse marcada todas as mensagens como urgentes. Este trabalho juntamente com os testes tem como premissa que o protocolo exige um transporte confirmado e assim o uso da entrega desordenada deve ser descartada e não possível de ser utilizada para aumento de desempenho

Com mensagens maiores, a vazão tende ao máximo permitido pela rede nos protocolos em questão. A vazão do SCTP é menor que XTP, pois o protocolo tem maior overhead inerente. A

partir do momento em que as mensagens diminuem de tamanho, o overhead aumenta em importância, e o desempenho do SCTP fica mais prejudicado. XTP utiliza-se do overhead para continuar com o seu desempenho constante.

Nos testes de vazão com múltiplos fluxos e tamanhos de pacotes variáveis, o SCTP beneficiou-se de uma característica do aplicativo construído para os testes de vazão, que os pacotes tem a característica de serem continuamente mandados em ambas as direções, o que permite ao SACK atuar com presteza.

O principal fator para o desempenho não "agradável" nos testes de latência deve-se a configuração default do controle de congestionamento do SCTP. O RTO mínimo utilizado e padrão do LK–SCTP é de 1 segundo; lembrando que o RTT da Ethernet é 5000 vezes maior. Assim, a medida que perdas acontecessem, causam um atraso de no mínimo 1 segundo na transação, elevando consideravelmente a média geral. O TCP não tem um limite mínimo dentro do RTO; ele será proporcional à latência da rede real, e, portanto adaptar-se a ela corretamente.

No momento em que o protocolo de aplicação troca pacotes continuamente nas duas direções, percebe-se que o SACK do SCTP tende a informar a perda no mesmo instante e o terminal origem toma providências antes do RTO. Assim para alguns testes, a vazão do SCTP tende a se aproximar do TCP e XTP o que não ocorre nos testes de latência.

6. CONCLUSÕES E TRABALHOS FUTUROS

O fundamento deste trabalho foi de realizar um estudo comparativo entre os protocolos TCP, SCTP e XTP usando como cenário real de uma rede interna de uma operadora de celular. A forma de realizar esse trabalho começou com uma pesquisa detalhada de cada protocolo, analisando e ajustando os mesmos para os testes dentro do ambiente. Até esse ponto, estudos sobre esse assunto já haviam sido relacionados e descritos nas referencias bibliográficas.

A conclusão que se chega após o estudo é que SCTP gera vários recursos interessantes para os protocolos de aplicações, sendo que colocados em cenários específicos podem dar ganhos bem maiores do que hoje TCP gera, principalmente em uma rede de operadora trafegando dados GPRS.

Percebe-se também que, para um estudo comparativo, os outros protocolos devem ter algumas alterações para que os resultados sejam mais justos, dentre eles ajustas para que os protocolos com mensagens grandes que não realizam a sua separação visto que assim, não conseguem aproveitar a atomicidade de mensagens do protocolo SCTP.

SCTP fornece como principais características o uso de diversas conexões de transporte para realizar uma única conexão lógica, agupando assim e uma única associação. Foram medidas vazão e latência em cenários de rede reais, com os resultados interessantes para SCTP que em alguns momentos ficou abaixo do TCP, devido ao maior overhead, controle de segurança, entre outros. Devido a múltiplos fluxos, características da rede, SCTP teve ganhos em alguns testes, o que comprova a vantagem da utilização deste recurso.

Os programas utilizados nos testes de latência e vazão foram construídos especialmente para isso, a medida de justiça foi de impor uma separação de mensagens na camada de aplicação do TCP, já em loopback testes foram realizados com TCP sem separação de mensagens.

Como sugestões para trabalhos futuros que abordem o SCTP, sugere-se:

- Alterações no algoritmo de controle de congestionamento, visto que em função dele alguns testes foram prejudicados.

- Alteração da rede para 10, 11, 1 Mbps visto que o cenário real utilizado estava apenas com rede de 100 Mbps. Em redes de operadoras, esses cenários são o ideal, mas para compara aplicações de terceiros, vários cenários pode trazer resultados mais vantajosos.
- Desligamento do checksum CRC-32c.
- Alteração dos protocolos comparados para aproveitar os benefícios que SCTP proporciona.

REFERÊNCIAS BIBLIOGRÁFICAS

ARIAS-RODRIGUEZ, I. **Stream Control Transmission Protocol – The design of a new reliable transport protocol for IP networks.** Helsinki University of Technology, Electrical and Communications Engineering Department. Networking Laboratory 2002/02/12.

BELLOVIN, S. M.; IOANNIDIS, J.; KEROMYTIS, A. D. et al. **On the use of SCTP with IPSec** (draft-ietf-ipsec-sctp-06.txt). IETF Network Working Group, 2003.

COENE, L. **RFC 3257** – Stream Control Transmission Protocol Applicability Statement. IETF Network Working Group, 2002.

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

COSTA, G. D. **SCTP – Uma alternativa aos tradicionais protocolos de transporte da internet**. Ciência Moderna, 2005.

DANTAS, M. **Tecnologia de rede de comunicação e computadores**. Axcel Books, 2002.

GEORGE, T.; BIDULOCK, B.; DANTU, R. et al. **SS7 MTP2-User Peer-to- Peer Adaptation Layer.** IETF Network Working Group, 2003.

JARDINI, G. Estudo de um Protocolo de transporte Lightweight e seus mecanismos de multicast.2000.180f. Dissertação (Mestrado em Ciências da Computação– UNB, 2000

JUNGMAIER, A.; RATHGEB, E. P.; SCHOPP, Michael et al. SCTP – A Multi-link End-to-end Protocol for IP-based Networks. **AEÜ – International Journal of Electronics and Communications.** 55 (2001) No. 1, 46-54, 2001.

JUNGMAIER, A.; RESCORLA, E.; TUEXEN, M. **RFC 3436** – Transport Layer Security over Stream Control Transmission Protocol. IETF Network Working Group, 2002.

JUNIOR, M. R. Uma ferramenta para simulação de desempenho do protocolo TCP em enlaces de rádio. 2004. Dissertação (Mestrado em Ciências da Computação) PUC.

KANG, S.; FIELDS, M. **Experimental Study of the SCTP compared to TCP**. Texas A&M University. Computer Communications and Network Fall, 2003.

KARN, P.; SIMPSON, W. **RFC 2522** – Photuris: Session-Key Management Protocol. IETF Network Working Group, 1999.

LKSCTP: *http://lksctp.sourceforce.net*. Sítio do LK-SCTP *(Kernel-Level SCTP)*, a implementação do SCTP no Linux. Acesso em 20 de mar. 2002.

MORNEAULT, K.; DANTU, R.; SIDEBOTTOM, G. et al. **RFC 3331** – Signaling System 7 (SS7) Message Transfer Part 2 (MTP2) – User Adaptation Layer. IETF Network Working Group, 2002.

NAGLE, J. **RFC 896** – Congestion Control in IP/TCP Internetworks. IETF Network Working Group, 1984.

ONG, L.; RYTINA, I.; GARCIA, M. et al. **RFC 2719** – Framework Architecture for Signaling Transport. IETF Network Working Group, 1999.

ONG, L.; YOAKUM, J. **RFC3286** – An Introduction to the Stream Control Transmission Protocol (SCTP). IETF Network Working Group, 2002.

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

PFÜTZENREUTER, E. Aplicabilidade e desempenho do protocolo de transporte SCTP. 2004. 119f. Dissertação (Mestrado em Ciência da Computação) – UFSC, 2004.

RUSSELL, T. **Signaling System #7.** 4. ed. McGraw-Hill, 2002.

SCTP.ORG:. Sítio dos principais autores do SCTP. Disponivel em < *http://www.sctp.org*>. Acesso em 09 set. 2007.

STEVENS, R. W. **RFC 2001** – TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithms. IETF Network Working Group, 1997.

STEVENS, R. W. **TCP/IP Illustrated Volume 1.** The Protocols. Addison- Wesley, 1994.

STEWART, R.; XIE, Q.; MORNEAULT, K. et al. **RFC 2960** – Stream Control Transmission Protocol. IETF Network Working Group, 2000.

STEWART, R. R.; XIE, Q. **Stream Control Transmission Protocol (SCTP).** Addison-Wesley, 2002.

STEWART, R.; RAMALHO, M.; XIE, Q. et al. **SCTP Partial Reliability Extension** (draft-stewart-tsvwg-prsctp-03.txt). IETF Network Working Group, 2003.

STEWART, R.; RAMALHO, M.; XIE, Q. et al. **Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration**. IETF Network Working Group, 2003.

STEWART, R., ONG, L., ARIAS-RODRIGUEZ, I. et al. **Stream Control Transmission Protocol (SCTP) Implementer's Guide**. IETF Network Working Group, 2003.

STEWART, R.; XIE, Q.; YARROLL, L. et al. **Sockets API Extensions for Stream Control Transmission Protocol – SCTP**. IETF Network Working Group, 2004.

STEWART, R; RAMALHO, M. et al. **RFC 3758 – Stream Control Transmission Protocol (SCTP) Partial Reliability Extension.** IETF Network Working Group, 2004.

STONE, J.; STEWART, R.; OTIS, D. **RFC 3309 – Stream Control Transmission Protocol (SCTP) Checksum Change.** IETF Network Working Group, 2002.
TANENBAUM, Andrew S. **Redes de computadores**. 4.ed. Rio de Janeiro: Campus, 2002.

TANENBAUM, Andrew S. **Redes de computadores**. 4.ed. Rio de Janeiro: Campus, 2002.

TELECO.COM:. Sítio de vários artigos sobre telecomunicações. Disponivel em: <*http://www.teleco.com.br*>. Acesso em: 10 jan. 2008.

UNICERT.COM: Sitio contend artigos sobre TCP. Disponivel em: <http://www.unicert.com.br/arquivos/>. Acesso em 20 out. 2007.

WEAVER, C. A.; SANDERS, R. The Xpress Transfer Protocol (XTP) – Tutorial. Computer Networks Laboratory, University of Virgina, 2007.

XIE,Q.; STEWART, R.; SHARP, C. et al. **SCTP Unreliable Data Mode Extension** (draft-ietf-tsvwg-usctp-00.txt). IETF Network Working Group, 2001.

ANEXO 1: Características de segurança do SCTP

Os protocolos de transporte da pilha TCP/IP não foram inicialmente projetados com o objetivo de proporcionar segurança. O protocolo TCP tem alguma segurança, mais incidental que

propositada, contra ataques "cegos" *(spoof)*, mas é sensível a ataques de negação de serviço *(DoS)*. As fraquezas do TCP são graves, pois é o protocolo de transporte mais utilizado, e a maioria das aplicações não apresenta qualquer mecanismo adicional de segurança.

As duas principais fraquezas do TCP são:

- não autenticar as partes durante a criação da conexão;
- alocar recursos do servidor antes de a conexão estar completamente aberta.

A única proteção do TCP contra ataques *blind spoof* é confiar que cada terminal gere números iniciais de seqüência (ISN) imprevisíveis. Essa imprevisibilidade foi prevista pelos criadores do TCP, porém como forma de distinguir mais facilmente encarnações diferentes da "mesma" conexão (i.e. conexão envolvendo mesmos endereços e portas). A implicação de segurança do ISN imprevisível foi percebida mais tarde.

Assim como uma conexão aberta, a conexão meio-aberta TCP ocupa uma estrutura TCB, pois lado passivo da conexão (tipicamente o servidor) precisa armazenar o ISN remetido pelo cliente no pacote SYN.

Como em geral o TCP é implementado no *kernel* do sistema operacional, e a memória do *kernel* é limitada, o número máximo de TCBs, e, portanto o número de conexões TCP simultâneas, tem um limite bem definido.

Assim, um invasor pode simplesmente disparar um grande número de pacotes TCP SYN com origem falseada – é o ataque *SYN flood*. As conexões meio-abertas nunca são efetivadas e acabam descartadas por *timeout*, porém nesse meio tempo elas ocupam todos os TCBs disponíveis e impedem a realização de conexões legítimas (BERNSTEIN). Felizmente, há uma técnica que praticamente elimina essa fraqueza do TCP: o *SYN cookie*. Consiste de um ISN gerado pelo servidor por um algoritmo criptográfico descrito por BERNSTEIN.

Depois de calcular o ISN e remeter o segundo pacote de *handshake*, o servidor não aloca TCB e não armazena nada sobre a conexão meio-aberta. Pode-se afirmar que o *SYN Cookie* transfere a responsabilidade do armazenamento da conexão meio-aberta para o cliente e para a rede. Se o cliente for legítimo e remeter o terceiro pacote de *handshake*, o servidor reconhece o seu ISN matematicamente, e só então atribui um TCB à conexão totalmente aberta.

Os *SYN cookies* funcionam, mas o número de seqüência (32 *bits*) é algo pequeno para fins de autenticação criptográfica. O uso de *SYN cookies* diminui a aleatoriedade do ISN. Por esse motivo, costuma ser ativado apenas quando há indícios de ataque e os TCBs estão escasseando.

Um "novo" ataque de negação de serviço contra o TCP, na verdade já conhecido há muito tempo, mas considerado apenas um risco teórico, tem se tornado provável com o aumento da velocidade da Internet, tanto nos *backbones* como nos terminais. É o simples disparo de pacotes RST (abortamento de conexão), com origem falseada, contra participantes de uma conexão legítima.

O agente hostil precisa conhecer os endereços IP e números de porta envolvidos, o que só é provável quando as vítimas usam padrões repetitivos de conexão. As principais vítimas em potencial são os roteadores globais da Internet, que utilizam o protocolo BGP, pois este último cria conexões TCP de longa duração e em portas fixas.

A proteção do TCP contra esse tipo de ataque é rejeitar pacotes cujo TSN esteja fora da janela atual. Com um TSN de 32 *bits* e uma janela típica de 4Kbytes, a chance de aceitar um RST falseado é de aproximadamente uma em um milhão (1 em 2^{32-12}) .

Porém:

- a disponibilidade de enlaces de banda larga (v.g. ADSL) tornou possível a um usuário doméstico emitir grande quantidade de pacotes por unidade de tempo;
- as conexões TCP tendem a usar janelas cada vez maiores (64Kbytes ou maiores) para máxima performance, aumentando a probabilidade de aceitar um pacote falseado;

- os ataques podem ser (e são) desfechados de forma distribuída, inclusive por vírus.

Tudo isso pode trazer problemas muito sérios ao TCP em breve. Os protocolos MDTP e SCTP foram desde o início projetados com salvaguardas de segurança contra os ataques descritos (STEWART & XIE, 2002).

4-way handshake

Em primeiro lugar, a criação da associação usa 4 pacotes, 2 em cada direção, o que permite um *handshake* mais elaborado. Os últimos dois pacotes de *handshake* já podem transmitir mensagens, de modo a iniciar a transmissão de dados o mais rápido possível. Em tese, o terceiro pacote de *handshake* TCP também pode transmitir dados. Mas a API *BSD Sockets* retorna de *connect()* apenas depois que o terceiro pacote já foi transmitido.

Felizmente, a extensão do *Sockets* para SCTP não apresenta o mesmo problema pois abre as associações de forma diferente. Os pacotes envolvidos na criação da associação, listados na seqüência normal de transmissão, têm os seguintes nomes (STEWART et al, 2000):

- **INIT**, do cliente para o servidor (”cliente” é quem toma a iniciativa da abertura);
- **INIT-ACK**, do servidor para o cliente em resposta a INIT;
- **COOKIE-ECHO**, do cliente para o servidor. Ao recebê-lo, o servidor considera estabelecida a associação;
- **COOKIE-ACK**, do servidor para o cliente, confirmando o recebimento de COOKIE-ECHO. Ao recebê-lo, o cliente considera estabelecida a associação.

Etiqueta de verificação (*verification tag*)

Na parte fixa da estrutura do pacote SCTP, existe um parâmetro de 32 *bits* denominado etiqueta de verificação (*verification tag*). Cada associação possui duas etiquetas, uma para cada direção. Não existe nada semelhante em TCP. Nos pacotes INIT e INIT-ACK, cada lado calcula e

transmite uma etiqueta de inicialização (*initiation tag*), e dali por diante deve repetir esse valor na etiqueta de verificação de todos os pacotes daquela associação. Todo pacote SCTP deve apresentar a etiqueta correspondente à sua associação e direção, sob pena de ser descartado (STEWART et al, 2000).

Essa etiqueta serve primariamente para identificar encarnações diferentes de uma mesma associação (i.e. entre os mesmos terminais e os mesmos números de porta). Mas também permite discriminar facilmente pacotes forjados, eliminando assim a possibilidade de *blind spoof*, tanto na tentativa de abertura de conexões anônimas quanto na tentativa de seqüestro ou abortamento de uma conexão existente.

O valor da etiqueta deve ser imprevisível para agentes externos, para que a proteção contra *blind spoof* seja efetiva. É o mesmo cuidado que deve ser tomado na geração do ISN do TCP, ou do ISN do SCTP, visto a seguir.

Número de seqüência de transmissão (TSN)

O TCP apresenta o parâmetro TSN (*transmission sequence number* – número de seqüência de transmissão) para indicar, em cada pacote, qual o segmento da seqüência de octetos que está sendo transmitido. Esse valor permite remontar a seqüência e detectar as lacunas. No TCP, o TSN é incrementado pelo número de octetos transmitidos.

O SCTP também apresenta o TSN nos trechos de dados, e sua utilidade é idêntica ao TSN no TCP, exceto pelo fato de ser incrementado por trecho, e não por octeto. Da mesma forma que no TCP, o TSN inicial (ISN) é informado de parte a parte durante a criação da associação, e tal qual TCP ele deve ser imprevisível por parte de um agente externo. Mas o TCP conta *apenas* com esse recurso para detectar ataques *blind spoof* que visem abortar ou seqüestrar uma conexão.

Já o SCTP conta com o TSN e também com a etiqueta de verificação, tornando-se bem mais resistente. A RFC 2960 (STEWART et al., 2000) sugere inicialmente que o ISN poderia ser uma duplicata da etiqueta de verificação, pois ambos têm o mesmo tamanho. Porém, STEWART &

XIE (2002) recomendam em seu livro que a geração dos dois valores seja independente para aumentar a resistência a ataques.

Cookies

Eliminado o *blind spoof*, resta evitar que o lado passivo da associação SCTP aloque TCBs com associações meio-abertas e fique vulnerável a ataque análogo ao *SYN flood*. Assim como no *SYN Cookie* do TCP, a solução do SCTP é transferir para o cliente a responsabilidade total pelo armazenamento dos dados da associação meio-aberta, através dos *cookies*.

O *cookie* SCTP é uma estrutura de tamanho variável, opaca (i.e. apenas o criador conhece seu formato interno), que o servidor transmite ao cliente no pacote INIT-ACK.

O cliente deve retransmitir o *cookie* ao servidor no pacote COOKIE-ECHO, como sugere o próprio nome do pacote. Se o cliente for na verdade um agente hostil "cego" mandando pacotes forjados, o *cookie* nunca chegará de volta ao cliente, nem será devolvido ao servidor. Como o servidor não ocupa memória com associações meio-abertas, o ataque não satura a tabela de associações e não impede as associações legítimas. Ataques no estilo *SYN flood* não são viáveis contra o SCTP.

Após transmitir o pacote INIT-ACK, o servidor não conserva *nenhuma* informação sobre a associação, nem sobre o potencial cliente, e a associação só é efetivada com o pacote COOKIE-ECHO. Portanto, o servidor depende exclusivamente das informações contidas no *cookie* para criar a associação. Logo, embora o formato do *cookie* seja de livre escolha, ele tem de conter obrigatoriamente os seguintes dados:

- os dados relevantes do pacote INIT. Normalmente será uma simples transcrição dos dados do pacote;
- os dados relevantes (gerados pelo servidor) do pacote INIT-ACK. Normalmente será também uma simples transcrição;

- duas etiquetas de 32 *bits* denominadas *tie tags*. Normalmente, essas etiquetas contém o valor zero, mas podem ser eventualmente preenchidas com as etiquetas de verificação do cliente e/ou do servidor. Elas servem para identificar retransmissões de pacotes de criação de associação.

Os dados acima são o mínimo necessário para a implementação funcionar, porém mais alguns são necessários para garantir a segurança do SCTP:

- *timestamp* para que *cookies* velhos possam ser descartados, bem como para proteção contra ataques de repetição (*replay attacks*);
- assinatura digital que garanta a integridade dos dados, permita ao servidor reconhecer o *cookie* como seu, e atrele o *cookie* ao cliente.

KARN e KRAWCZYK et al. descrevem os mecanismos para a geração de um *cookie* seguro. A assinatura digital é tipicamente obtida concatenando-se todos os dados do *cookie*, os endereços IP, as portas, uma chave secreta e calculando-se um *hash* de qualidade criptográfica. (KRAWCZYK et al. sugere MD5 ou SHA-1). A RFC 2960 recomenda que a chave secreta seja trocada de forma razoavelmente freqüente. Deve haver uma janela de aceitação da chave velha para evitar que associações em fase de criação sejam incorretamente ilegitimadas.

Obviamente, a chave secreta é concatenada apenas na memória do servidor para fins de cálculo do *checksum*, e não vai fazer parte do *cookie* transmitido.

A criptografia dos dados do *cookie* não é especialmente encorajada, pois não traz qualquer vantagem de segurança; os dados ali contidos poderiam ser obtidos facilmente de outras formas. O *hash* é suficiente para fins de autenticação.

Embora a implementação seja livre para escolher o tamanho do *cookie*, deve procurar escolher o menor tamanho possível para evitar problemas de interoperabilidade. Se considerarmos 16 octetos para os dados relevantes de INIT, mais 16 para os dados de INIT-ACK, 8 para os *tie-tags*, 8 para o *timestamp* e 16 para a assinatura digital, chegamos a um *cookie* de 64 octetos.

Somatório de verificação (*checksum*)

Segundo SHANNON, um *checksum* de n *bits* pode detectar, no máximo:

- 100% dos erros de até n *bits*;
- 2^n - 1 em cada 2^n erros que afetem aleatoriamente mais de n *bits*.

Este é o melhor desempenho teoricamente possível (o trabalho de Shannon não descreveu *como* criar somatórios eficazes); algoritmos reais terão menor capacidade de detecção. O algoritmo CRC aproxima a expectativa da previsão teórica para erros em rajadas, desde que seu "polinômio gerador" seja bem escolhido.

Isso vale não só para números binários. O dígito verificador presente em números de conta bancária tem as mesmas propriedades: detecta 100% dos erros simples de digitação, e deixa passar 10% dos demais erros – se o algoritmo gerador for de boa qualidade, como o "Módulo 11".

O *TCP checksum*, somatório utilizado em TCP e UDP, é menos eficiente que o CRC na detecção de erros, mas ainda detecta 100% dos erros de 1 *bit* – que constituem a grande maioria dos erros – e é muito mais eficiente no cálculo por *software* que o CRC. (Deve-se levar em conta que essa escolha foi feita numa época em que as CPUs tinham velocidade muito menor que hoje.)

Em média, 1 em cada 5000 pacotes da Internet (0,2%) é entregue com algum erro (ARIAS-RODRIGUEZ *apud* PAXSON). Os meios de transmissão da Internet são muito confiáveis (v.g. Ethernet e FDDI usam CRC-32) e não justificam tamanha quantidade de erros. Os erros são na verdade provocados majoritariamente por problemas nos roteadores – desde memória de má qualidade, interferência eletromagnética até *bugs* na implementação dos respectivos *softwares*.

Um *checksum* de 16 *bits*, por melhor que seja seu algoritmo, deixa passar 1 em 2^{16} dos pacotes errados, para erros completamente aleatórios. Isso significa que 1 em 327.680.000 (5000 _

65536) dos pacotes da Internet entregues à camada de aplicação contém erros. É uma chance de erro pequena, porém observável, o que obriga a camada de aplicação a implementar algum mecanismo adicional de proteção. HTTP e FTP não possuem tal mecanismo, e portanto não são confiáveis para transmissão de dados sensíveis, se a integridade dos arquivos não é verificada de outra forma.

Como o SCTP tem a pretensão de transmitir mensagens de telefonia, o grau de confiabilidade tem de ser mais alto que o oferecido pelo *checksum* de 16 *bits*. E estabelecer essa confiabilidade no protocolo de transporte desonera as aplicações dessa tarefa. O *checksum* inicialmente escolhido para o SCTP foi o Adler-32. Pesquisas posteriores determinaram que o poder de detecção de erros desse algoritmo é fraco para mensagens menores que 1 Kbyte (que é o caso do pacote SCTP típico).

O código CRC é antigo, mas ainda é dos mais poderosos na deteção de erros. Sua principal desvantagem é a lentidão no cálculo por *software* (embora seja rápido em *hardware*).

Após muita deliberação, optou-se finalmente pelo CRC-32c. Levou-se em conta o grande poder de processamento dos dispositivos modernos. A mudança de *checksum* do SCTP está oficializada na RFC 3309 (STONE et al, 2002).

Quanto à escolha do polinômio gerador, a primeira opção foi o CRC-32 tradicional utilizado em Ethernet e FDDI, porém o polinômio CRC-32c protege melhor mensagens pequenas, de modo que acabou este último sendo o eleito para o SCTP.

Um *checksum* ótimo de 32 *bits* deixa passar apenas 1 em 232 pacotes errados, para erros completamente aleatórios. Levando em conta a taxa de erros da Internet (1 em 5000), o SCTP entregaria à aplicação apenas 1 pacote errado a cada aproximadamente 21.474.836.480.000 trafegados, ou seja, a entrega de uma mensagem corrompida à aplicação é virtualmente impossível.

O somatório CRC não é uma assinatura digital e portanto não protege os dados contra fraude.

Sem garantias criptográficas

Os mecanismos de segurança do SCTP evitam apenas os ataques “cegos”. O SCTP não oferece por conta própria garantias criptográficas (confidencialidade, autenticidade, integridade), nem é resistente a ataques do “homem do meio”. Tais garantias de segurança podem ser providas por outros protocolos, e.g. IPSEC e SSL.

A utilização de SSL com SCTP está descrita na RFC 3436 (JUNGMAIER et al, 2002). As questões de utilização de IPSEC com SCTP em multicaminhos são descritas em BELLOVIN et al.

ANEXO 2: Detalhes de funcionamento do protocolo SCTP

Formato dos pacotes

O pacote SCTP possui um cabeçalho fixo de 4 parâmetros (12 octetos), mais um número variável de trechos (*chunks*) alinhados em 32 *bits*, conforme o esquema abaixo. Os parâmetros fixos estão em negrito.

Cabeçalho IP	
Porta de origem *(16 bits)*	**Porta de destino** *(16 bits)*
Etiqueta de verificação *(32 bits)*	
Somatório de verificação *(32 bits)*	
Trecho 1 *(trechos alinhados em 32 bits)*	
Trecho 2	
Trecho n	

Toda a troca de informações de controle do SCTP (abertura de associação, fechamento de associação etc.) é feita através de trechos de tamanho variável, e não *bitmaps* como em TCP.

Os números de porta de origem e destino têm o mesmo fim que em TCP e UDP. A função dos demais campos é detalhada no ANEXO 1.

Notar que as portas estão nas mesmas posições relativas utilizadas por TCP e UDP, o que facilita a interpretação desses números por aplicativos de diagnóstico de rede. Os primeiros 8 octetos do cabeçalho SCTP contém todas as informações necessárias para identificar univocamente uma associação, consoante com as mensagens ICMP que incluem no mínimo 8 octetos do pacote que provocou o erro. Como em todo protocolo da pilha TCP/IP, os dados que representam números inteiros devem ser representados em *network byte order* – o primeiro octeto é o mais significativo.
Os trechos são estruturas de dados TLV *(type, length, value* – tipo, comprimento e valor). São alinhados em 32 *bits* e podem apresentar até 3 octetos de enchimento. O valor do comprimento inclui os 4 octetos dos três primeiros campos, porém não inclui o enchimento.

Tipo *(8 bits)*	*Flags (8 bits)*	Comprimento *(16 bits)*
Dados úteis do trecho, mais enchimento ($32n$ *bits*)		

O parâmetro *"Flags"* é um mapa de *bits* interpretado de acordo com o tipo de trecho (não há nenhum *flag* com significado genérico). Como o SCTP foi projetado para ser extensível, pode suceder de um tipo de trecho presente em uma implementação não ser suportado por outras

implementações. A reação do receptor a um trecho não suportado depende dos dois *bits* MSB do tipo de trecho:

Tipo	*Ação tomada pelo receptor, se não suporta o trecho*
0x00 a 0x3F	descartar o pacote que contém o trecho
0x40 a 0x7F	descartar o pacote e remeter uma notifi cação de erro
0x80 a 0xBF	desprezar o trecho e continuar a processar o pacote
0xC0 a 0xFF	desprezar o trecho, continuar a processar o pacote, e remeter uma notifi - cação de erro

O código do tipo deve ser escolhido em função da reação desejada.

No espaço de dados úteis, os trechos predefinidos pelo protocolo têm parâmetros *permanentes* e *opcionais*.

Os parâmetros permanentes têm tamanho e ordem predefinidos para cada tipo, tal qual uma estrutura da linguagem C, e sempre vêm primeiro em relação aos parâmetros opcionais. O bloco de parâmetros permanentes deve ser alinhado em 32 *bits*. Em seguida vêm os parâmetros opcionais, em número variável, que também são
estruturas TLV:

Tipo de parâmetro *(16 bits)*	Comprimento *(16 bits)*
Dados úteis do parâmetro, mais enchimento ($32n$ *bits*)	

Assim como nas estruturas TLV dos trechos, o comprimento do parâmetro inclui os 4 octetos do tipo e do próprio comprimento, mas não inclui os octetos de enchimento. Analogamente ao tipo de trecho, os dois *bits* MSB do tipo de parâmetro também especificam a reação a um parâmetro desconhecido pelo receptor:

Tipo de parâmetro	*Ação tomada pelo receptor, se não suporta o parâmetro*
0x0000 a 0x3FFF	descartar o trecho que contém o parâmetro
0x4000 a 0x7FFF	descartar o trecho e remeter uma notifi cação de erro
0x8000 a 0xBFFF	desprezar o parâmetro e processar o trecho
0xC000 a 0xFFFF	desprezar o parâmetro, processar o trecho, e remeter uma notifi cação de erro

Motivação de uso da estrutura TLV

A estrutura TLV é simples de entender, implementar e processar, e sem dúvida muito mais versátil que um *bitmap* fixo. Alguns testes realizados por Randall Stewart (um dos criadores do SCTP), citados por ARIAS-RODRIGUEZ, determinaram que o processamento de uma estrutura TLV é mais rápido que o de um *bitmap*.

O bitmap não é de todo desvantajoso. Ele pode ser muito pequeno, reduzindo a sobrecarga de rede. Protocolos como IP e TCP utilizam bitmaps, pois, à época de sua criação, as baixas velocidades dos enlaces justificavam qualquer esforço de diminuição
da sobrecarga.

A situação atual, segundo TANENBAUM, é oposta - a vazão é limitada predominantemente pela latência de processamento dos terminais, e a redução dessa latência deve ser a principal preocupação dos projetistas de protocolos. Os protocolos devem ser simples, ágeis no processamento e extensíveis.

Desta forma, o SCTP utiliza majoritariamente estruturas TLV para transporte de dados e informações de controle. Até mesmo tarefas como abertura e fechamento de associação utilizam tais estruturas, e não *flags* (*e.g.* ACK, SYN, FIN do TCP).

Suporte ao SCTP em aplicativos-ferramentas

A alocação de todas as informações em estruturas TLV torna mais difícil a interpretação dos pacotes SCTP por parte de ferramentas de diagnóstico de rede. Não basta enquadrar o pacote numa estrutura C; a ferramenta precisa conhecer suficientemente o protocolo para separar e interpretar os TLVs.

Como o próprio *payload* está encapsulado em uma estrutura TLV, a figura didática da separação rígida entre cabeçalho de transporte e *payload*, não existe em SCTP. Um pacote SCTP pode ter dois ou mais trechos de dados, então haverá dois *payloads* incrustados dentro das estruturas de controle SCTP.

Em TCP e UDP, a tupla formada pelos pelos endereços e portas dos terminais identifica univocamente uma conexão, mesmo na Internet. Pode-se começar a monitorar uma conexão a qualquer momento, e individualizá-la imediata e facilmente. Em SCTP, o suporte a multicaminhos torna mais difícil a individualização da associação. Para uma identificação perfeita, o aplicativo tem de monitorar a associação desde sua criação, para conhecer os endereços alternativos informados de parte a parte.

Se o monitoramento começar com a associação multicaminhos já estabelecida, a individualização será imperfeita. As tuplas formadas pelas etiquetas de verificação pode ser usadas como dado auxiliar de individualização, mas não são perfeitas pois existe sempre a (na verdade muito pequena) probabilidade de associações diferentes apresentarem as mesmas etiquetas.

Embora não sejam problemas insolúveis, certamente os fatores acima têm atrasado o suporte a SCTP em aplicativos-ferramentas, bem como em equipamentos ativos de rede.

Decisões de implementação

Conforme a RFC 2960 (STEWART et al, 2000), há algumas decisões deixadas à implementação do protocolo SCTP.

Maior mensagem suportada

Muito embora o tamanho máximo teórico da mensagem seja de $2^{32} – 1$ octetos, a implementação pode estabelecer um limite prático menor. Um dos limites é o tamanho máximo do *buffer* de recepção, que está na memória do *kernel* e sofre restrição relativamente severa de tamanho. Outro limite prático é imposto pelo PTMU da rede e pelos 16 *bits* do SSN – uma mensagem pode ser fragmentada em no máximo 2^{16} trechos, o que limita seu tamanho a 2^{16} x PMTU.

É também facultada à implementação limitar a mensagem ao PMTU, de modo que caiba integralmente em um datagrama e dispense o algoritmo de remontagem de mensagens. Dispositivos de memória restrita como celulares podem determinar tais restrições. Mensagens SS7 podem ter, no nível de aplicação, no máximo 272 ou 4091 octetos, dependendo do meio de transmissão (ONG et al, 1999); dispositivos que utilizem SCTP exclusivamente para sinalização de telefonia podem ater-se a esses limites.

Formato do *cookie*

O tamanho e forma de geração do *cookie* de autenticação é de responsabilidade da implementação. As RFCs 2522 e 2104 sugerem algoritmos e fontes de dados para ele.

A implementação também pode restringir o tamanho máximo do *cookie* recebido, por limitações de memória etc. Isso limita sua interoperabilidade com outras implementações que usem *cookies* maiores.

Por esse motivo, e prevendo que o SCTP possa ser usado em dispositivos naturalmente limitados em capacidade de memória e processamento, a RFC 2960 recomenda que a implementação crie o menor *cookie* possível.

Suporte a multicaminhos

A implementação de multicaminhos é opcional em cada uma de suas modalidades (endereços IPv4, endereços IPv6 e nomes DNS).

Por exemplo, uma implementação pode suportar apenas multicaminhos com endereços IPv4. Isso é interessante se o sistema operacional subjacente não suporta IPv6 – se todas as modalidades fossem obrigatórias, sistemas sem IPv6 estariam impedidos de implementar SCTP. Também é possível que algumas implementações optem por não suportar multicaminhos baseado em endereços DNS, por questões de segurança.

Ainda, em alguns dispositivos, não faz nenhum sentido usar multicaminhos.

Suporte a confiabilidade parcial (PR–SCTP)

A extensão PR–SCTP modifica a semântica do prazo de validade das mensagens. Uma mensagem cujo prazo de validade estoure, é descartada mesmo que sua transmissão já tenha sido tentada. Ela é de implementação opcional.

Convivência do SCTP com outras tecnologias

A grande maioria dos problemas de convivência do SCTP com outras tecnologias de rede TCP/IP são derivadas do recurso de multicaminhos.

IPv6

As versões iniciais do SCTP previam apenas multicaminhos para IPv4, o que era uma grande limitação. Felizmente, o suporte a multicaminhos IPv6 foi introduzido numa das revisões do protocolo. O SCTP permite até mesmo mistura de endereços IPv4 e IPv6 para multicaminhos de uma mesma associação.

Uma observação importante, encontrável na RFC 2960, é que não se pode usar um parâmetro de endereço IPv6 contendo um endereço IPv4 mapeado em IPv6 (::FFFF:0/96). Para endereços IPv4, deve-se utilizar o parâmetro de endereço IPv4, contendo o endereço IP de 32 *bits*.

Roteadores NAT

NAT para SCTP sem multicaminhos tem o mesmo grau de dificuldade que TCP. A priori não faz sentido fazer NAT para SCTP com multicaminhos, pois há muitas variáveis envolvidas:

- Haverá apenas um roteador NAT envolvido, ou um NAT para cada caminho?
- Se houver mais de um roteador NAT envolvido, como os NAT secundários ficarão sabendo da associação, se os pacotes INIT, INIT-ACK etc. trafegaram apenas pelo NAT primário?
- Quem é multicaminhos: o terminal ou o roteador NAT?
- Existe mistura de endereços válidos e inválidos?
- Quem define que endereços devem ou não ser traduzidos?

No caso específico dos terminais “ocultos” monocaminhos, e um único roteador NAT *multi-homed*, existe um potencial interessante de aumentar a tolerância a falhas. Conforme a alternativa proposta por COENE, o protocolo SCTP foi estendido para aceitar endereços *multi-homed* em forma de nome DNS.

Os computadores "ocultos" podem tirar proveito do roteador NAT *multi-homed*, desde que informem como caminhos alternativos os nomes DNS do roteador NAT. Mais fácil ainda é associar um único nome DNS a todos os endereços IP dos caminhos. É responsabilidade do terminal remoto fazer a resolução DNS do nome. Como o SCTP deve em geral ser implementado no *kernel* dos sistemas operacionais, e o protocolo DNS em geral está fora do *kernel*, é provável que as implementações escolham entre:

- não suportar multicaminhos com nomes DNS;

- criar um processo-agente que resolva nomes DNS para o *kernel*, com alguma provisão de segurança contra ataques de negação de serviço.