Danillo Franscys Borges de Oliveira

TCP-HPL
TCP para Redes sem Fio com Alta Taxa de Perdas

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2010

Danillo Franscys Borges de Oliveira

TCP-HPL
TCP para Redes sem Fio com Alta Taxa de Perdas

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Redes de Computadores

Orientador: Mauro Fonseca

Curitiba
2010

Oliveira, Danillo Franscys Borges de
TCP-HPL
TCP para Redes sem Fio com Alta Taxa de Perdas. Curitiba, 2010.

Dissertação - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Protocolos 2. TCP 3. Redes sem fio I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e Tecnologia. Programa de Pós-Graduação em Informática II - t

Dedico este trabalho primeiramente a minha família, formada por minha esposa Gesiane e meu filho Matheus Henrique, sem os quais não haveria tão expressivo valor no resultado alcançado pois sem o apoio e compreensão da minha família não seria possível o término deste.
Também não poderia deixar de incluir meus pais, os quais sempre me incentivaram e me apoiaram em todos os momentos da minha vida.
A graça de Deus e sua presença em minha vida me possibilitaram a paz e as condições para atingir este tão almejado objetivo.

Agradecimentos

Agradeço ao professor Mauro Fonseca por ter acreditado no meu trabalho, pelo incentivo que me deu principalmente nas horas de maior desânimo e por sua orientação clara e precisa sem a qual não teria atingido os objetivos propostos.

Agradeço a PUC por ter disponibilizado excelentes professores nas disciplinas de preparação do curso de redes de computadores e pela excelente infraestrutura oferecida.

Agradeço a Siemens Ltda por ter patrocinado parte dos custos com o Mestrado e principalmente ao Sr. Márcio Machado por ter incentivado o desenvolvimento intelectual dos colaboradores da área de P&D.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

Lista de Abreviações

TCP-HPL	*TCP for High Packet Losses*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
BER	*Bit Error Rate*
AIMD	*Additive Increase Multiplicative Decrease*
MARFIM	*Medicina Assistida por Redes sem FIo Multimídia*
IAM	*infarto agudo do miocárdio*
AToMS	*AMI Teleconsultation & Monitoring System*
PDA	*Personal Digital Assistants*
RTO	*Retransmission timeout*
OSI	*Open System Interconnection*
RTT	*Round-trip Time*
MSS	*Maximum Segment Size*
SS	*slow-start*
CA	*congestion avoidance*
SACK	*Selective Acknowledgement*
ABE	*Available Bandwidth Estimation*
CW	*Congestion Warning*
ECN	*Explicit Congestion Notification*
HSDPA	*High Speed Data Packet Access*

AQM	*Active Queue Management*
SNR	*Signal-to-noise ratio*
ARQ	*Automatic Repeat reQuest*
FEC	*Forward Error Correction*
NIC	*Network Interface Card*
MSR	*Mobile Support Router*
MH	*Mobile Host*
FH	*Fixed Host*
TCP-F	*TCP feedback*
RFN	*Route Failure Notification*
RRN	*Route Re-establishment Notification*
ELFN	*Explicit Link Failure Notification*
ATCP	*Ad Hoc TCP*
TCP-BuS	*TCP Buffering capability and Sequence information*
ABR	*Associativity-Based Routing*
ERDN	*Explicit Route Disconnection Notification*
ERSN	*Explicit Route Successful Notification*
PN	*Pivoting Node*
LQ	*Localized Query*
RRC	*Router ReContruction*
TCP DOOR	*TCP Detection of Out-of-Order and Response*
OOO	*Out-of-order*
ADSN	*ACK Duplication Sequence Number*
TPSN	*TCP Packet Sequence Number*
LACK	*Local Acknowledgement*
RTO	*Retransmission Timeout*
SRTT	*Smoothed Round Trip Time*
TCB	*Transmission Control Block*

MSL *Maximum Segment Lifetime*

Resumo

Este trabalho descreve uma arquitetura denominada TCP-HPL (*TCP for High Packet Losses*) para amenizar a perda de pacotes em redes sem fio com alta taxa de perdas de pacotes. Ele é transparente para ambos o TCP originador e receptor com relação a sintaxe do TCP. O TCP-HPL aplica métodos como bufferização, controle de temporizações e controle de ACKs afim de evitar a ativação do mecanismo de controle de congestionamento após perda de pacotes, degradação na taxa de envio de dados e retransmissão desnecessária de segmentos pelo TCP originador.

O TCP-HPL foi desenvolvido em linguagem C através do uso de *raw sockets* para possibilitar a manipulação dos segmentos TCP e a aplicação dos mecanismos propostos.

Um cenário de teste consistindo de 3 computadores em série foi usado para simular um *download* de dados de um dispositivo móvel cujo servidor está localizado na porção de rede cabeada. Todos os computadores foram configurados com linux distribuição Ubuntu, versão 8.04LTS e placa de rede de 100Mbps. O TCP original se refere a implementação do protocolo TCP presente no Ubuntu e definida na RFC 793 (INSTITUTE, 1981), RFC 1122 (BRADEN, 1989) e RFC 2001 (STEVENS, 1997) com as extensões NewReno e SACK.

Primeiramente o TCP-HPL foi configurado apenas para encaminhar pacotes, mas com a possibilidade de introduzir erros aleatórios, de modo a simular os erros na parte de rede sem fio. Os valores da taxa de perda de pacotes entre 0 e 5% foram utilizados para obter a curva de degração do *throughput*no TCP original.

Após a obtenção dos resultados com o TCP original o TCP-HPL foi inserido no computador intermediário e o mesmo procedimento foi realizado para se obter a curva de degração do *throughput* em função da taxa de perda de pacote de 0 a 5%.

O mecanismo proposto para melhorar o desempenho do TCP em redes sem fio com

alta perda de pacotes possibilita que a taxa de transferência de dados seja mantida praticamente constante mesmo com o aumento da taxa de perda de pacotes, diferentemente do TCP original. Uma desvantagem do TCP-HPL em relação ao TCP original é o pior desempenho em condições normais, com uma taxa baixa de perda de pacotes.

A principal vantagem é que o TCP-HPL é transparente tanto para o emissor quanto para o receptor, sem alteração da sintaxe do TCP. Ele não requer qualquer mudança no cliente ou servidor TCP ao contrário de outras propostas.

Palavras-chave: TCP, proxy, redes sem fio.

Abstract

This paper describes an architecture to mitigate the packet loss on wireless networks called TCP-HPL. The TCP-HPL improves the data download rate in environments with high packet loss. It is transparent for both sender and receiver TCP regarding the TCP syntax. The TCP-HPL applies methods like buffering, timing control and ACKs control in order to avoid the congestion phase start after packet losses, throughput degradation and unnecessary segments retransmission by the sender TCP.

The TCP-HPL was developed in C by using *raw sockets* in order to allow the TCP segments handling and the proposed methods applying.

A test scenario consisting of 3 computers in series was used to simulate the data download from a mobile device, whose server is in the wired portion of network. All computers were configured with Linux distro Ubuntu version hardy 8.04LTS and network cards of 100Mbps. The original TCP is the implementation of the TCP protocol present on Ubuntu defined in RFC 793 (INSTITUTE, 1981), RFC 1122 (BRADEN, 1989) and RFC 2001 (STEVENS, 1997) with the NewReno and SACK extensions.

First of all the proxy has been adjusted only to forward packets but with the possibility of introducing random errors in order to simulate the errors in the wireless portion of network. Values of packet loss rate between 0 and 5% were used to obtain the original TCP throughput degration curve.

After obtaining the results with the original TCP the TCP-HPL was inserted in the intermediate computer and the same procedure was used to obtain the throughput curve in function of packet loss rate from 0 to 5%.

The proposed mechanism to improve the TCP performance in wireless networks with high packet loss ensures that the data throughput is kept even with increasing the packet loss rate differently from original TCP. The major problem with the TCP-HPL results is

the poor performance in normal conditions with low packet loss rate.

The major benefit is that the TCP-HPL is transparent for both sender and receiver, that means the TCP syntax is not changed. It does not require any change on client or server TCP as opposed to other proposals.

Keywords: TCP, Proxy, Wireless Networks.

Capítulo 1

Introdução

TCP (*Transmission Control Protocol*) e IP (*Internet Protocol*) são amplamente utilizados na comunicação entre entidades fim a fim para a transmissão de dados. A finalidade do TCP (INSTITUTE, 1981) é prover uma entrega confiável dos dados em redes não confiáveis. A combinação do TCP/IP domina na comunicação de dados em vários tipos de redes, desde o *backbone* cabeado, até as redes heterogêneas, devido a sua simplicidade e confiabilidade e dão suporte a uma série de aplicações, tais como acesso web, transferência de arquivos, telnet e e-mail entre outras.

Nos últimos anos estamos vivenciando uma explosão dos dispositivos portáteis e móveis. Ao mesmo tempo, a Internet tem sofrido um aumento expressivo, se tornando uma rede mista, compreendendo redes sem fio e redes cabeadas. Redes Ad Hoc e redes de telefonia 3G também estão evoluindo e sendo ampliadas progressivamente. As redes Ad Hoc são complexos sistemas distribuídos que consistem de redes móveis sem fio ou nós estáticos que podem livremente e dinamicamente se auto-organizar. Redes sem fios, tais como redes de celular 3G e redes Ad Hoc, consistem de múltiplas camadas de protocolos, entre eles, o TCP. Na prática, muito do desenvolvimento do TCP tem sido aplicado no contexto das redes cabeadas. Nas redes com fio o BER (*Bit Error Rate*) é negligenciável, sendo o congestionamento a principal causa da perda de pacotes.

Cada pacote TCP é associado com um número de seqüência, e somente os pacotes recebidos na seqüência correta são reconhecidos pelo receptor através do envio de pacotes contendo o número de seqüência do próximo pacote esperado. A perda de pacotes ou a recepção de pacotes fora de ordem indicam falhas. Para eliminar tais falhas o TCP implementa algoritmos de controle de fluxo e controle de congestionamento baseado no esquema de janela deslizante e algoritmo AIMD (*Additive Increase Multiplicative Decrease*) (CHIU; JAIN, 1989) de aumento aditivo e decremento multiplicativo da taxa de

envio dos pacotes.

No caso das redes sem fio alguns problemas nos mecanismos do TCP são detectados, devido às particularidades destas redes, entre eles a degradação da taxa de envio de dados (*throughput*), ineficiência na utilização dos recursos da rede e excessivas interrupções na transmissão dos dados, causando limitações no desempenho do TCP nestes ambientes.

1.1 Desafio

Diferentemente das fibras óticas e cabos, os enlaces das redes sem fio usam o ar como meio de transmissão e estão sujeitos a muitos fatores incontroláveis que afetam a qualidade na transmissão, tais como condições do tempo, obstáculos urbanos, interferência multi-percurso, movimentação de objetos grandes e a mobilidade dos dispositivos sem fio. Como resultado, enlace de redes sem fio apresentam BER muito maiores que enlaces de redes cabeadas. Também, limitações de cobertura de rádio e mobilidade do usuário requerem freqüentes *handoffs*, resultando deste modo, em desconexões temporais e reconexões durante a sessão de comunicação. Uma curta desconexão pode reduzir drasticamente a taxa na transmissão TCP por um longo período.

Nas redes celulares 3G geralmente o tráfego, no sentido *downlink*, da estação base para o dispositivo móvel, tende a consumir mais largura de banda que no sentido *uplink*, tornando o tráfego assimétrico. Esta assimetria no enlace resulta em taxas e atrasos variáveis. Esta característica associada ao modo como o TCP gerencia o controle de fluxo pode diminuir o desempenho na transmissão no sentido *downlink* devido às falhas no sentido *uplink*. Em uma rede Ad Hoc as altas taxas de erros, mudanças freqüentes na rota e particionamentos na rede são comuns. Estas características resultam em perdas de pacotes além do congestionamento na rede, devendo ser, portanto, tratadas de maneira diferente pelo TCP.

Devido aos problemas do TCP em ambientes mistos e redes sem fio, muitas propostas de mudanças no protocolo têm surgido com o objetivo de melhorar o seu desempenho. Muitas das mudanças resolvem problemas inerentes a características específicas de um determinado tipo de rede ou tentam propiciar mecanismos para evitar ou tratar as situações de congestionamento, mas o problema principal se resume ao fato que o TCP é incapaz de distinguir perdas devido a falhas na rota e congestionamento na rede. Neste sentido, atualmente não existem soluções que atendam a todos os casos de maneira satisfatória, sendo objeto, portanto, de intensas pesquisas.

1.2 Motivação

Em vista dos vários problemas apresentados quando temos o TCP em ambientes de rede sem fio faz-se necessário o entendimento destes problemas para a proposição de novas alternativas, neste sentido um dos objetivos foi o de estudar os problemas relacionados a este ambiente, de modo a compreender as causas e a maneira como elas afetam a conexão TCP.

Baseados nas verificações e conclusões obtidas procurou-se propor um mecanismo para melhora do desempenho do TCP de maneira a manter inalterada a semântica do TCP nas aplicações visto que várias propostas existem mas requerem alterações no TCP dos dispositivos.

Como exemplo de aplicação do uso do TCP, em ambientes de rede sem fio que exigem alto desempenho na comunicação de dados, podemos citar o caso dos serviços médicos executados dentro das ambulâncias. Esta foi uma das aplicações que motivaram o desenvolvimento de uma solução que atendesse a demanda por uma conexão TCP mais eficiente em redes sem fio.

Atualmente existem serviços médicos que são executados na ambulância durante o atendimento a vítima. Ambulâncias equipadas com acesso a redes sem fio podem se comunicar com as centrais médicas para coletar e fornecer dados da vítima, agilizando desta forma o atendimento médico.

Nas situações em que minutos podem fazer a diferença entre a vida e a morte, é necessário o máximo de eficiência em todos os aspectos. Não poderia ser diferente com relação à comunicação entre a ambulância e a central médica, quanto mais rápida for a transmissão de dados, maiores serão a chance de oferecer uma prestação de socorro eficiente.

Uma evidência desta necessidade é apresentada no projeto MARFIM (*Medicina Assistida por Redes sem FIo Multimídia*) (FLUMINENSE, 2008), o qual é um projeto-piloto para casos de IAM (*infarto agudo do miocárdio*).

O principal objetivo do projeto MARFIM é desenvolver um sistema de telemedicina que dê suporte à decisão de aplicação de trombolíticos em casos de IAM. Para viabilizar esse suporte, o sistema proposto, chamado nesta proposta de AToMS (*AMI Teleconsultation & Monitoring System*), pressupõe que o paramédico atendendo o paciente com IAM seja dotado, no mínimo, de um dispositivo sem fio para monitoramento eletrocardiográfico (WiECGs) e de um computador de bolso PDA (*Personal Digital Assistants*) com capa-

cidade de comunicação sem fio integrada (p. ex. incluíndo rede celular e/ou sem fio com comunicação de dados, voz e imagem). Essa configuração oferece uma série de facilidades no atendimento de pacientes com IAM. A Figura 1.1 (FLUMINENSE, 2008) apresenta uma visão geral do sistema.

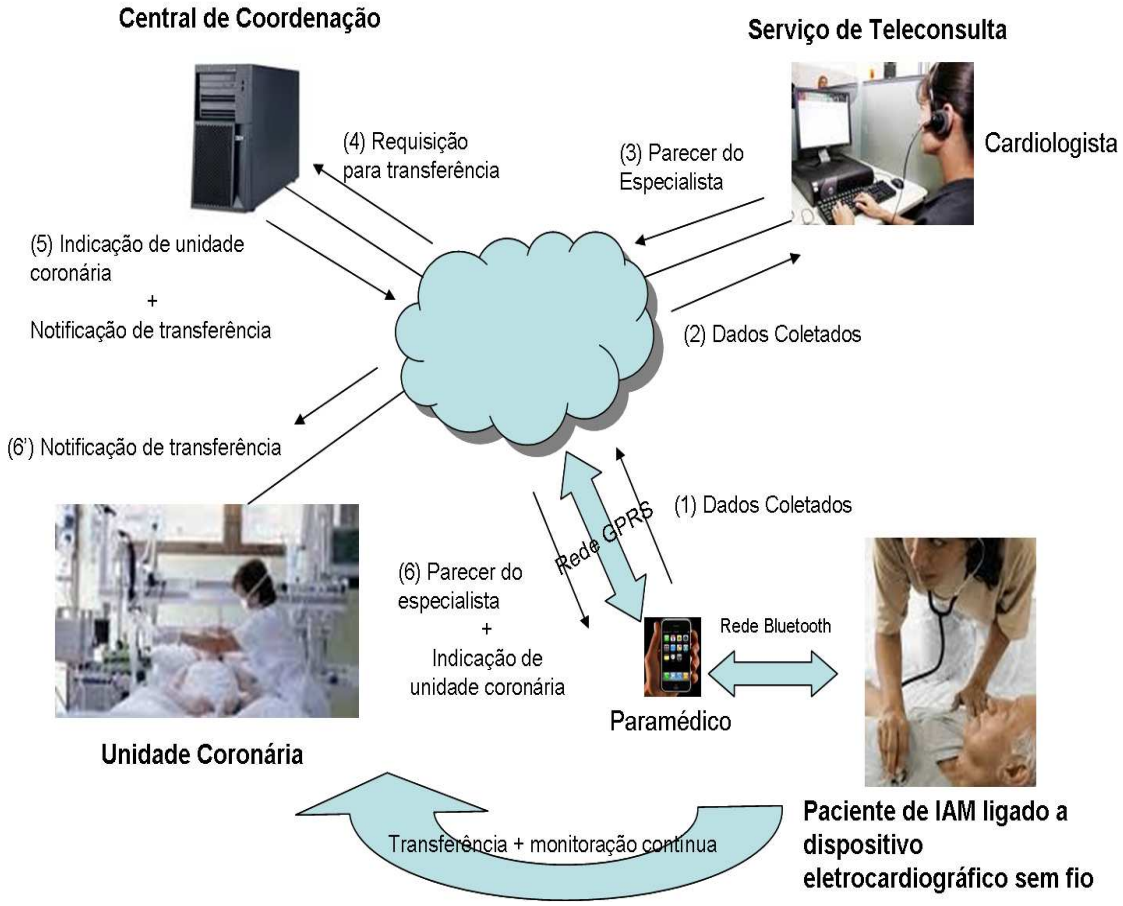

Figura 1.1: Visão geral do sistema AToMS

O projeto adota a tecnologia WiFi (IEEE 802.11) e topologias do tipo malha em suas redes.

Redes em malha possuem a vantagem de serem redes de baixo custo, fácil implantação e bastante tolerantes a falhas. Nessas redes, roteadores sem fio são tipicamente instalados no topo de edifícios e comunicam-se entre si diretamente ou através de roteadores sem fio intermediários, modo este de funcionamento conhecido como ad hoc. Usuários nos edifícios podem se conectar à rede em malha de forma cabeada (tipicamente via Ethernet), ou de forma sem fio (normalmente via WiFi).

Novas soluções podem ser propostas visando otimizar o desempenho e a robustez da infra-estrutura de comunicação sem fio. Neste contexto o TCP-HPL foi desenvolvido.

1.3 Proposta

A proposta do trabalho se refere à criação de um mecanismo para a melhoria do TCP em ambientes de rede sem fio. A proposta inicial pode ser estendida posteriormente para ambientes de rede sem fio com características específicas, tais como redes Ad Hoc e redes celulares 3G, os quais possuem também problemas específicos.

O mecanismo é baseado na implementação de uma camada intermediária entre as camadas do TCP e o IP promovendo a interface entre elas, de modo a não causar alterações no protocolo TCP, mas promovendo uma melhoria na taxa de transmissão de dados e caracterizando uma solução fim a fim.

A proposta visa a criação de um *proxy* que serve como conexão entre a parte de rede cabeada e a rede sem fio de modo que a conexão TCP permaneça inalterada entre os extremos, e, através da introdução de mecanismos como bufferização, ordenação dos pacotes, emulação ou retenção de ACKs, melhorar o desempenho TCP em um trecho específico da rede sem fio.

Após um estudo sobre os principais problemas apresentados pelo TCP em ambientes de rede sem fio e as soluções atuais, foram verificadas as vantagens e desvantagens de cada solução, bem como as limitações impostas por elas. Nota-se que muito se têm feito para as soluções dos problemas de utilização do TCP em redes sem fio, mas, apesar de excelentes soluções terem sido apresentadas há um problema ainda maior: devido à imensa difusão do TCP em vários equipamentos seria muito complicada a troca do protocolo em todos estes equipamentos, inviabilizando a adoção de qualquer uma das soluções anteriores.

A proposta tem como premissa tornar transparente a utilização do TCP entre os dispositivos de forma que possa ser aplicada em soluções pontuais, melhorando o desempenho do TCP em um trecho específico da rede.

Medidas como bufferização, ordenação dos pacotes, omissão e retenção de ACKs, de modo à "enganar"o TCP do transmissor e receptor são empregadas, baseadas em algumas das soluções já vistas, como o ATCP (LIU; SINGH, 2001) . Também mecanismos heurísticos de estimativa da largura de banda e temporizações, como os empregados no RTO (*Retransmission timeout*) fixo (DYER; BOPPANA, 2001) foram incorporados a solução.

1.4 Contribuição

A principal contribuição se refere a possibilidade de utilização do TCP-HPL de forma transparente para os dispositivos que participam da conexão TCP sem haver necessidade de substituição da variante TCP que roda nestes dispositivos.

O TCP-HPL se mostrou bastante eficaz em situações com alta taxa de perdas de pacotes podendo ser usado em enlaces que sofrem dos principais problemas do TCP em redes sem fio tais como: baixa qualidade de sinal, interferências e constantes handoffs.

1.5 Organização

O documento apresenta no capítulo 2 um estudo sobre as principais variantes TCP e as soluções que elas apresentam. Também são apresentados os principais problemas nas redes sem fio, redes celulares e redes Ad Hoc e as principais propostas para resolvê-los. As propostas são comparadas e analisadas.

No capítulo 3 é descrita a arquitetura proposta para melhora do desempenho do TCP em ambientes de rede sem fio. Os principais módulos são descritos em termos funcionais e o ambiente de desenvolvimento é detalhado.

No capítulo 4 são apresentados os resultados obtidos com a utilização do TCP-HPL.

Finalmente no capítulo 5 são apresentadas as principais conclusões sobre os resultados obtidos e sobre o trabalho em geral e também são expostas possibilidades de trabalhos futuros.

Capítulo 2

Estado da Arte

Neste capítulo são abordados temas referentes ao estado da arte em termos de funcionalidade, utilização e problematização do uso do TCP em redes de computadores cabeadas e redes sem fio, bem como as soluções empregadas atualmente. Na seção 2.1 são apresentadas a origem e as funcionalidades do TCP. Na seção 2.2 são apresentadas as variantes do TCP e as modificações feitas para o aprimoramento de suas funcionalidades. Na seção 2.3 são apresentados os algorítmos e técnicas utilizadas nas variantes do TCP com maior profundidade. Na seção 2.4 são apresentados os problemas e desafios da utilização do TCP em redes sem fio e na seção 2.5 nas redes Ad Hoc. Na seção 2.6 são apresentadas as propostas atuais de soluções para os problemas do TCP em redes sem fio e na seção 2.7 as propostas atuais para as redes Ad Hoc. Finalmente na seção 2.8 é apresentada a conclusão geral deste capítulo.

2.1 Funcionalidades do TCP

Criada pelo departamento de defesa dos E.U.A, a ARPANET, a qual fez uso da tecnologia de comutação de pacotes, foi a precursora da Internet moderna. Ambos o TCP e o IP foram inspirados e originalmente escritos baseados respectivamente nas RFC 793 (INSTITUTE, 1981) e RFC 791, em setembro de 1981. Os protocolos residem em diferentes camadas da arquitetura OSI (*Open System Interconnection*) e, portanto, tem funções distintas, mas relacionadas na comunicação de dados. IP é um protocolo de rede não orientado a conexão, baseado no esquema *best-effort*, o qual realiza entrega de pacotes de comprimento variável, mas sem garantia de entrega, tempo ou seqüência. Ele foca no mecanismo de roteamento, o qual guia os pacotes de um *host* a outro, ou múltiplos

destinos, baseados em um esquema de endereçamento. TCP é um protocolo de transporte orientado a conexão que visa garantir uma entrega confiável e seqüenciada dos pacotes de dados em redes cabeadas. Para este propósito, funcionalidades básicas, tais como: controle de fluxo, controle de erro e controle de congestionamento são indispensáveis. Um mecanismo de janela deslizante é utilizado para o controle de fluxo, no qual três janelas são usadas: janela de congestionamento, janela de anúncio e janela de transmissão. A janela de congestionamento indica o número máximo de pacotes que o transmissor pode transmitir sem congestionar a rede. Este número é determinado pelo transmissor baseado no retorno indicado pela janela de anúncio. A janela de anúncio, entretanto, é determinada pelo receptor, nas mensagens de reconhecimento (ACK). Esta janela indica para o transmissor a quantidade de dados que o receptor está apto a receber. Normalmente ela é igual ao tamanho do *buffer* disponível no receptor, de modo a prevenir o estouro do *buffer*. A janela de transmissão indica a quantidade de segmentos que o transmissor pode transmitir de uma vez, sem o recebimento de algum ACK do receptor. Seu limite inferior indica o maior número de segmentos reconhecidos pelo receptor. Para evitar o congestionamento na rede ou estouro do *buffer*, o tamanho da janela de transmissão é determinado pelo mínimo da janela de congestionamento e a janela de anúncio do receptor. Para notificar o transmissor que os dados foram corretamente recebidos, TCP emprega um mecanismo de reconhecimento acumulativo de ACK. Quando um ACK é recebido, o transmissor sabe que os segmentos de dados anteriormente transmitidos com um número de seqüência menor que aquele indicado no ACK foram corretamente recebidos pelo receptor. No caso de um segmento fora de ordem recebido pelo receptor, um ACK duplicado é gerado e enviado de volta para o transmissor. No caso de redes cabeadas, a entrega de um segmento fora de ordem geralmente implica em perda de pacote. Se três ACKs cumulativos são recebidos, o transmissor assume que o pacote foi perdido. A perda de pacote também é assumida quando o transmissor não recebe um ACK do segmento dentro de um intervalo chamado tempo de retransmissão RTO, o qual é dinamicamente calculado baseado no RTT (*Round-trip Time*) mais quatro vezes o desvio médio. Pela retransmissão dos pacotes perdidos, o TCP garante confiabilidade na entrega. No caso das redes cabeadas, a maior parte da perda de pacotes é devida ao congestionamento da rede e não por erros na transmissão. Por isso, em adição a retransmissão, o TCP responde a perda de pacotes ativando seus mecanismos de controle de congestionamento. O controle de congestionamento do TCP também é baseado no mecanismo de janela deslizante, o qual foi descrito anteriormente e consiste de duas fases principais: slow start e congestion avoidance. Na fase slow start, o tamanho inicial da janela de congestionamento (cwnd) é configurada com um tamanho máximo de segmento MSS (*Maximum Segment Size*) e é

incrementado de um MSS a cada novo ACK. Depois que o tamanho da janela de congestionamento alcança um limite pré-determinado (ssthresh), o controle de congestionamento inicia e é incrementado linearmente, por um segmento para cada RTT. Depois de um timeout, o sstresh é configurado com a metade da atual janela de transmissão e a janela de congestionamento é reduzida de um MSS. Então o mecanismo de slow start inicia novamente. Este procedimento é chamado de algoritmo de incremento aditivo e decremento multiplicativo (AIMD (CHIU; JAIN, 1989)). Em (CHEN et al., 2005) é apresentado um gráfico que descreve o comportamento da janela de congestionamento do TCP.

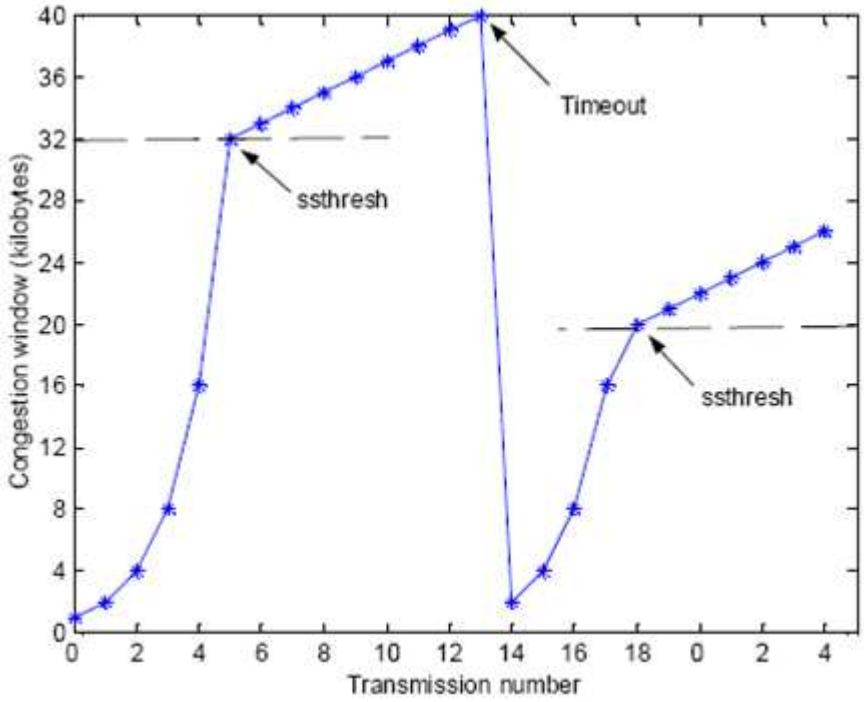

Figura 2.1: Janela de Congestionamento Dinâmica do TCP

2.2 Variantes do TCP

Muito do progresso feito no TCP está relacionado com recuperação de erros e controle de congestionamento. De acordo com (LI; ANSARI, 2005) segue o resumo de alguns progressos efetuados no protocolo TCP e suas variantes.

2.2.1 TCP Tahoe

Na implementação original do TCP, um modelo go-back-n usando ACK cumulativos é empregado. Esta implementação também necessita do estouro de um temporizador para o reenvio dos dados perdidos durante o transporte. Mais tarde achou-se que este tipo de TCP fez pouco para minimizar o congestionamento na rede. TCP Tahoe (V.JACOBSON, 1988) divide a transmissão em três fases, denominadas SS (*slow-start*), CA (*congestion avoidance*), e *fast retransmit*. Estas caracteristícas novas melhoraram a utilização do canal e taxa de transmissão de dados da conexão.

2.2.2 TCP Reno

TCP Reno (STEVENS, 1997) introduziu um mecanismo, o qual é chamado *fast recovery*, e ele é baseado na idéia que o único caminho para uma perda ser detectada via um *timeout* e não via a recepção de pacotes duplicados é quando o fluxo de pacotes parou completamente, indicando com isso um congestionamento pesado. Através do uso do *Fast Recovery* poucos pacotes são enviados até que o TCP reconheça que não há perigo iminente de congestionamento.

2.2.3 TCP New Reno

Na variante TCP Reno vários problemas foram encontrados com a perda de múltiplos pacotes em uma janela de dados (geralmente na ordem da metade da janela). Alguns estudos (K.FALL, 1996) mostraram que quando mais de uma perda ocorre dentro de uma janela de dados, os mecanismos de *fast retransmit* e *fast recovery* serão disparados várias vezes em um RTT. Para resolver estes problemas foi proposto o TCP New Reno. Ele atua do mesmo modo que o Reno no caso da perda de um segmento, entretanto, quando existe a perda de múltiplos pacotes, os ACKs para os pacotes retransmitidos irão reconhecer alguns, mas não todos os segmentos enviados antes do procedimento de *fast retransmit*. Diferente do Reno, o TCP New Reno retransmite um pacote por RTT até que todos os pacotes perdidos sejam retransmitidos. Este procedimento evita o disparo de múltiplos fast retransmit dentro de uma janela de dados.

2.2.4 TCP SACK

TCP SACK (*Selective Acknowledgement*) (MATHIS J. MAHDAVI; ROMANOW, 1996) é outra tentativa para resolver o problema do TCP Reno, mas com uma abordagem que requer mudanças em ambos o transmissor e receptor. O SACK não requer que os segmentos sejam reconhecidos cumulativamente mas seletivamente, desde que o transmissor possa descobrir um simples pacote perdido por RTT através das informações disponíveis nos ACKs cumulativos. Para isso um campo opcional SACK é adicionado ao pacote de ACK, o qual reporta blocos não contíguos de dados que precisam ser recebidos corretamente. Deste modo, o transmissor pode retransmitir somente a parte que falta.

2.2.5 TCP-Jersey

As variantes anteriores funcionam bem no caso de redes cabeadas, onde a perda de pacotes é devida principalmente aos congestionamentos e os erros devido a problemas triviais nos enlaces. Entretanto, em ambientes de rede wireless, o TCP tradicional apresenta degradações no throughput (LEFEVRE, 2000), devido à perda de pacotes, a qual ocorre devido às características do meio e dispositivos, não podendo ser, portanto, negligenciáveis. TCP-Jersey (XU YE TIAN, 2004) foi proposto com uma abordagem fim a fim. Duas melhoras foram introduzidas: o algoritmo de estimativa de largura de banda disponível ABE (*Available Bandwidth Estimation*) e a configuração de rotas com aviso de congestionamento CW (*Congestion Warning*). ABE é indicado para estimar com maior precisão o tempo de variação da largura de banda da rede. O transmissor TCP é modificado para monitorar a taxa dos ACKs recebidos para calcular o tamanho ótimo da janela de congestionamento de modo a ajustar a sua taxa de transmissão quando a rede se encontra congestionada. TCP-Jersey calcula a janela ótima de congestionamento a cada RTT usando um algoritmo de janela deslizante modificado. O cálculo do ABE é efetuado conforme descrito abaixo:

$$R_n = \frac{RTT * R_{n-1} + L_n}{(t_n - t_{n-1}) + RTT} \tag{2.1}$$

Onde R_n é a largura de banda estimada quando o n-ésimo ACK é recebido e L_n a quantidade de dados. RTT é o atraso estimado fim a fim no tempo t_n. Através da largura de banda estimada, a janela ótima de congestionamento cwnd em unidades de segmentos pode ser calculada:

$$cwnd_n = \frac{RTT * R_n}{segment\_size} \tag{2.2}$$

CW é um ECN (*Explicit Congestion Notification*) (FLOYD, 1993) como configuração de roteadores de rede permitindo que estes possam alertar as estações finais através da marcação de todos os pacotes quando há indícios de congestionamento na rede. As diferenças entre ECN e CW recaem principalmente no esquema de marcação. O ECN marca os pacotes de maneira probabilística quando o comprimento médio da fila está entre min e max, enquanto CW marca todos os pacotes quando o tamanho médio da fila ultrapassa um limite. A vantagem de uma marcação não probabilística é que o transmissor, quem recebe estas marcas, pode decidir sua estratégia de ajuste da janela sem ser influenciado pela marcação probabilística dos pacotes no caso do ECN. O propósito do CW é carregar uma imagem simplificada dos gargalos na fila para o transmissor. A marcação dos pacotes pelos roteadores configurados com CW também ajuda o transmissor na diferenciação da perda de pacotes causada por congestionamentos na rede daquela causada por erros no link wireless. TCP-Jersey trabalha com a combinação do ABE e CW. Em (LI; ANSARI, 2005) é apresentado o resultado de uma simulação em uma rede HSDPA (*High Speed Data Packet Access*) com uma área de cobertura de 19 hexágonos com foco na medição da quantidade de bits efetivamente entregues através da rede.

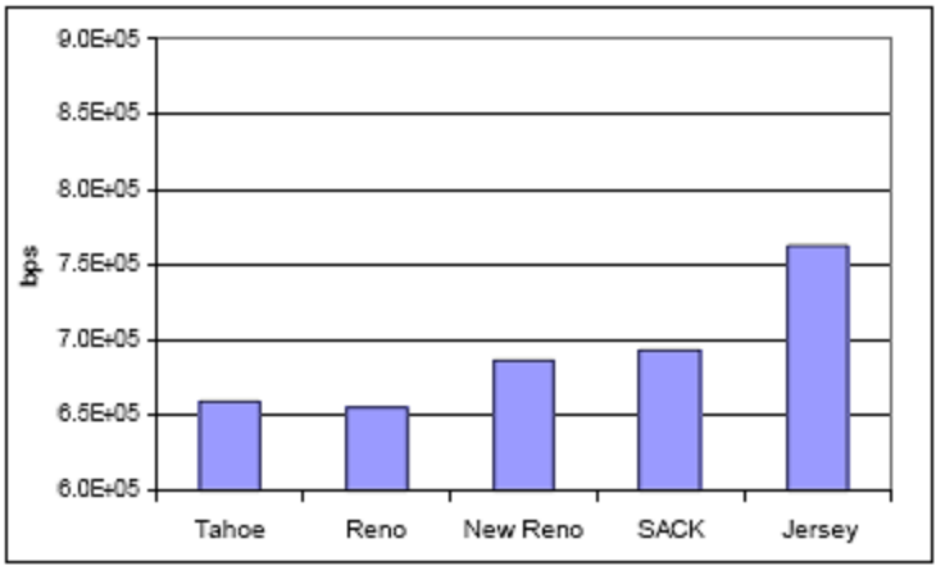

Figura 2.2: Média da taxa de saída de bits (19 usuários/célula)

Os resultados mostram que o TCP-Jersey tem um desempenho superior se comparado com as outras variantes TCP. Duas características contribuem para este desempenho:

a) O CW permite que o TCP-Jersey possa diferenciar erros nos pacotes nos enlaces wireless dos congestionamentos na rede. Quando o transmissor recebe ACK duplicado, sem a marcação do CW, ele assume que ocorreu um erro no link.

b) No caso de um sistema de rede celular sabe-se que a taxa de saída é maximizada quando os recursos de rádio são alocados de acordo com a condição de mudanças nos canais devido a mobilidade do usuário. Neste caso o TCP-Jersey trabalha bem em conjunto com o protocolo MAC. De fato, o ABE ajuda o transmissor a estimar mais precisamente as mudanças de largura de banda, possibilitando uma melhor coordenação com o escalonador de recursos da camada MAC.

2.3 Melhorias no TCP

Em (CHEN et al., 2005) são apresentados algumas das melhorias introduzidas no TCP.

2.3.1 Fast Retransmission e Fast recovery

Um pacote é considerado perdido quando três ACKs duplicados são recebidos. Neste caso, o TCP realiza uma retransmissão rápida do pacote (STEVENS, 1997). Este mecanismo permite ao TCP evitar um timeout demorado durante o qual o pacote é transferido. Ao mesmo tempo, ssthresh é configurado para metade da janela de congestionamento, cwnd, sendo que cwnd é configurado com o valor de ssthresh mais três segmentos. Se o ACK é recebido em aproximadamente um RTT depois que o segmento perdido é transmitido, é acionado o mecanismo de retransmissão rápida. Isto significa que ao invés de configurar cwnd com um segmento e iniciar com o slow start, o TCP configura o cwnd para o valor de ssthresh, e então salta para a fase de congestion avoidance. Entretanto, somente um pacote perdido pode ser recuperado durante a fase de fast retransmission e fast recovery. Pacotes adicionais perdidos na mesma janela podem requerer que o RTO expire antes da retransmissão.

2.3.2 Selective Acknowledgment

Partindo do fato que os mecanismos de fast retransmission e fast recovery podem somente manipular um pacote perdido dentro de uma janela de dados, pode ocorrer

um desempenho ruim do TCP quando múltiplos pacotes são perdidos em uma janela. Para superar esta limitação, foi desenvolvida a opção de reconhecimento seletivo (SACK) (MATHIS J. MAHDAVI; ROMANOW, 1996) como um adicional à implementação padrão TCP. A extensão SACK adota duas opções TCP. Uma opção pode ser habilitada, a qual pode ser enviada para indicar que a opção SACK será usada a partir do estabelecimento da conexão. A outra é a própria opção SACK, a qual pode ser enviada pelo receptor sobre uma conexão estabelecida se a opção SACK é habilitada através do envio da primeira opção. A opção SACK contém até quatro blocos SACK, os quais especificam blocos contíguos de dados recebidos. Cada bloco SACK consiste de dois números de seqüência, os quais delimitam a faixa de dados que o receptor recebeu e enfileirou. Um receptor pode adicionar a opção SACK aos ACKs que ele envia de volta para um transmissor com SACK habilitado. No caso de múltiplos pacotes perdidos dentro de uma janela, o transmissor pode verificar quais pacotes foram perdidos e devem ser retransmitidos usando a informação disponibilizada nos blocos SACK. Um transmissor com SACK habilitado pode retransmitir múltiplos pacotes perdidos em um RTT ao invés de detectar somente um pacote perdido em cada RTT.

2.3.3 Random Early Detection (RED)

O RED (FLOYD, 1993) é um mecanismo de controle de congestionamento baseado no roteador, o qual procura detectar congestionamentos iminentes e notificar alguns transmissores TCP do congestionamento através do controle do tamanho médio da fila no roteador. Para notificar os transmissores TCP do congestionamento, o roteador pode marcar ou descartar alguns pacotes, dependendo se os transmissores são cooperativos. Como resposta,os transmissores devem reduzir suas taxas de transmissão. Isto é feito através de dois algoritmos. O primeiro algoritmo calcula o tamanho médio da fila através do uso de uma média variável com peso exponencial. Se representarmos por avg e q, respectivamente, o tamanho médio da fila e o tamanho atual da fila, então temos que:

$$avg = (1 - wq) * avg + wq * q \tag{2.3}$$

onde wq é o peso da fila. O outro algoritmo calcula a probabilidade de marcação ou descarte de pacotes pa. Se avg cai na faixa entre th_{min} e th_{max}, a probabilidade de marcação do pacote é dada por:

$$pb = p_{max}\frac{avg - th_{min}}{th_{max} - th_{min}} \tag{2.4}$$

A probabilidade final de marcação ou descarte é dada por:

$$pa = \frac{pb}{1 - count * pb} \tag{2.5}$$

Onde p_{max} e *count* são parâmetros de desenvolvimento, denotando, respectivamente, o valor máximo para *pb* e o número de pacotes que chegaram desde o último pacote marcado ou descartado. Se *avg* ultrapassa th_{max}, $pa = 1$, o que significa que o roteador marca ou descarta cada pacote que chega. Através do controle sobre o tamanho médio da fila em prioridade ao estouro da fila, RED possibilita a prevenção de congestionamentos pesados na rede e sincronização global.

2.3.4 Explicit Congestion Notification (ECN)

Muitos dos roteadores atuais para Internet empregam o mecanismo tradicional de gerenciamento da fila através do descarte do final da fila *drop-tail*. Neste mecanismo os roteadores descartam pacotes somente quando ocorre o estouro da fila, o qual pode levar a problemas de sincronização global bem como, a um congestionamento pesado na rede. Mecanismos de gerenciamento ativos da fila AQM (*Active Queue Management*) têm sido proposto pois eles podem detectar congestionamentos nos roteadores e informar os transmissores antes do estouro das filas, evitando deste modo, alguns dos problemas causados pela política *drop-tail*. Na falta do ECN, entretanto, a única opção disponível para o AQM, para indicação de congestionamento para os sistemas finais, é o descarte de pacotes nos roteadores. Com o ECN, mecanismos de AQM têm uma alternativa para permitir roteadores na notificação de sistemas finais sobre o congestionamento na rede. O ECN (RAMAKRISHNAN; BLACK, 2001) requer algumas mudanças no cabeçalho, tanto do TCP quanto do IP. No cabeçalho IP é utilizado um campo com dois bits. Através da configuração destes campos com bits específicos, o roteador pode enviar uma indicação de congestionamento para o sistema final. No TCP, dois novos flags no campo reservado são especificados. Através da manipulação destes dois novos flags, o transmissor e o receptor podem habilitar o ECN via negociação durante a configuração da conexão; o receptor pode informar o transmissor se ele recebe indicações de congestionamento dos roteadores intermediários, e o transmissor pode informar o receptor que ele está ativando mecanismos

de controle de congestionamento.

2.4 Desafios do uso do TCP em redes wireless

Comparada com redes cabeadas, redes wireless com somente um salto têm características inerentemente adversas que podem deteriorar significativamente o desempenho do TCP se nenhuma ação for tomada. Em essência, essas características incluem surtos de erros nos canais, mobilidade e assimetria na comunicação. De acordo com (CHEN et al., 2005) os problemas mencionados nos próximos sub-capítulos estão presentes em ambientes wireless.

2.4.1 Erros no Canal

Nos canais de uma rede sem fio a taxa relativamente alta de erro nos bits (BER) devido a problemas tais como multipath fading e sombras podem corromper pacotes na transmissão, levando a perdas de segmentos de dados TCP ou ACKs. Se o ACK não é recebido durante o tempo de retransmissão, o transmissor reduz imediatamente sua janela de congestionamento para um segmento e, exponencialmente reduz o tempo de RTO e retransmite o pacote perdido. Erros intermitentes no canal podem fazer com que o tamanho da janela de congestionamento no transmissor permaneça pequeno, resultando em uma baixa taxa de saída do TCP.

2.4.2 Mobilidade

Redes celulares são caracterizadas por constantes handoffs devido à mobilidade do usuário. Normalmente handoffs podem causar desconexões temporárias, resultando na perda de pacotes e atrasos. Neste caso o TCP sofrerá uma considerável degradação se ele tratar tais perdas como congestionamento e disparar mecanismos desnecessários de controle de congestionamento. Espera-se que os handoffs sejam mais freqüentes nas próximas gerações de redes celulares porque a estrutura de micro células irá acomodar um maior número de usuários. Problemas similares podem ocorrer em uma LAN wireless se os usuários móveis encontrarem interrupções na comunicação caso eles se movam para os limites da faixa de transmissão do ponto de acesso.

2.4.3 Assimetria

Em redes wireless com somente um salto (*one-hop*) o enlace entre a estação base e o terminal móvel é por natureza assimétrico. Comparado com a estação base o terminal móvel tem limitações de potência, capacidade de processamento e espaço para bufferização.

Em resumo, enlaces wireless exibem uma taxa de BER muito maior que em redes cabeadas. Também, limitações de cobertura de rádio e mobilidade do usuário requerem *handoffs* frequentes, resultando em desconexões e reconexões temporais. Uma simples desconexão curta pode eventualmente paralizar a transmissão TCP por um longo período, conforme pode ser observado na Figura 2.3 (TIAN; ANSARI, 2005).

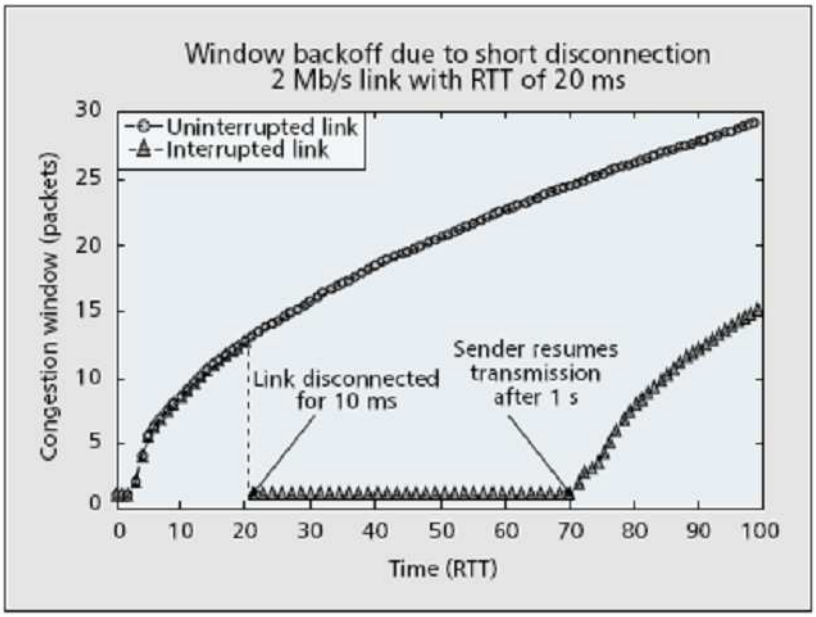

Figura 2.3: Efeito de uma curta desconexão na transmissão TCP

O efeito da evolução da janela de congestionamento se reflete no throughput da transmissão. Este resultado foi obtido através de uma conexão TCP entre dois *hosts*, sendo que a capacidade do link é de 2Mb/s e o RTT é de 20ms. Durante a desconexão, ambos os pacotes de dados e os ACKs são descartados, e cada tentativa de retransmissão leva a uma retransmissão sem sucesso. Estas falhas nas sucessivas tentativas de retransmissão implicam que o transmissor TCP retroceda exponencialmente o seu temporizador por uma granularidade típica de 500ms. O tempo total de retransmissão, no qual a transmissão ficou paralizada foi de aproximadamente 50 RTTs, ou seja, 1s. Um TCP padrão, como o Reno, não controla efetivamente as altas taxas de BER e frequentes desconexões.

Como todos os pacotes são considerados como o resultado de um congestionamento na rede, a perda de pacotes randômicos causada pela alta BER nos links wireless irá fazer com que o transmissor reduza a taxa de transmissão desnecessariamente. Os algoritmos de fast retransmit e fast recovery introduzidos pelo TCP Reno podem recuperar perdas rápidas e esporádicas de pacotes somente se estas ocorrerem uma vez dentro de um RTT. Entretanto, ruídos e outros fatores relacionados ao ambiente wireless geralmente causam erros randômicos nos bits, os quais ocorrem em rajadas curtas, levando a uma alta probabilidade de múltiplas perdas randômicas de pacotes dentro de um RTT. Novamente, múltiplas retransmissões sem sucesso dentro de um RTT causam um retrocesso exponencial do tempo de retransmissão. Dois pacotes perdidos, por exemplo, dentro de um RTT levaria o TCP a paralizar a transmissão por um período de aproximadamente 1s. Este fenômeno é mostrado na Figura 2.4 (TIAN; ANSARI, 2005).

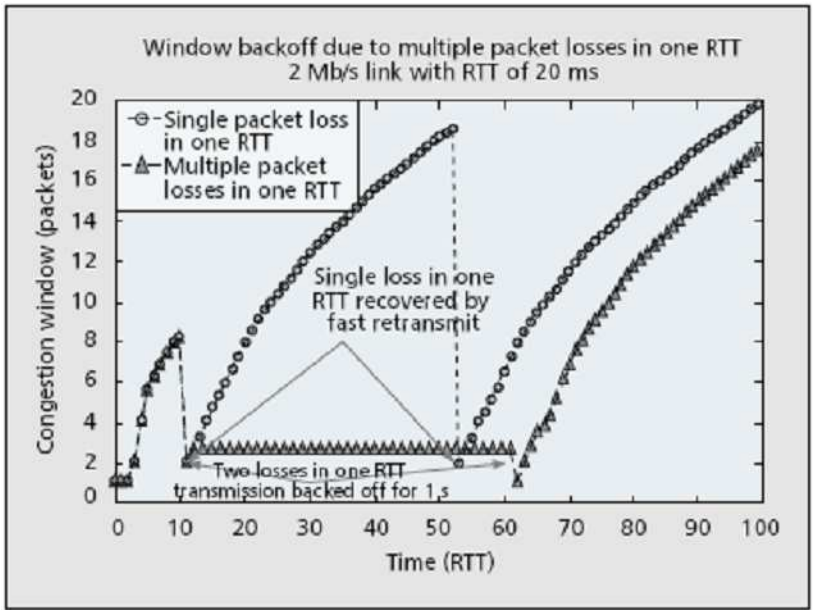

Figura 2.4: Efeito de múltiplas perdas em um RTT na transmissão TCP

Dois pacotes descartados em períodos afastados no tempo não causariam paralização na retransmissão. A inabilidade do protocolo padrão TCP de distinguir entre perdas de pacotes devido a congestionamentos e erros randômicos contribuem para a degradação do throughput.

2.5 Desafios do uso do TCP em redes Ad Hoc

Redes Ad Hoc são complexos sistemas distribuídos que consistem de nós móveis ou estáticos, os quais podem livremente e dinamicamente se auto-organizar. Deste modo eles formam topologias arbitrárias e temporárias Ad Hoc, permitindo aos dispositivos se interconectarem em áreas sem uma infraestrutura pré-definida. A introdução de protocolos tais como Bluetooth (BLUETOOTH...,), IEEE 802.11 (IEEE...,) e HyperLan (BROADBAND..., 2000-02) possibilitou o uso de redes Ad Hoc com propósito comercial. Freqüentemente o uso da expressão "MANET"é empregado no caso de redes móveis Ad Hoc, e o termo "SANET"no caso de redes estáticas Ad Hoc. É esperado que, comparado com as redes wireless com somente um salto (*one-hop*), o TCP encontre dificuldades mais sérias em prover comunicação fim a fim em redes móveis Ad Hoc, as quais não possuem estrutura, são auto-organizáveis em redes multi-hop e carecem de um gerenciamento centralizado da rede.

Segundo (HANBALI; ALTMAN, 2005) os 4 maiores problemas no TCP com relação as redes Ad Hoc são:

a) TCP é incapaz de distinguir entre perdas devido às falhas na rota e congestionamento na rede.

b) TCP sofre com freqüentes falhas na rota.

c) A concorrência no canal wireless.

d) TCP é injusto

Os dois primeiros problemas causam a degradação do desempenho do TCP em Ad Hoc móveis ("MANET") e, os outros dois nas redes Ad Hoc estáticas (SANET).

2.5.1 Perdas no canal

As principais causas de erros em canais wireless são as seguintes: a) Atenuação do sinal: deve-se ao decremento na intensidade da energia eletromagnética no receptor, devido principalmente a longas distâncias, o que leva a uma baixa relação sinal-ruído SNR (*Signal-to-noise ratio*). b) Doppler shift: deve-se as velocidades relativas entre o transmissor e o receptor. O efeito doppler shift causa deslocamento nas frequências na chegada do sinal, complicando deste modo o sucesso na recepção do sinal. c) Multipath fading: ondas eletromagnéticas são refletidas nos objetos ou sofrem difração, o que pode resultar em

flutuações na amplitude, fase, e ângulo geográfico do sinal recebido pelo receptor. A fim de melhorar o sucesso das transmissões, protocolos na camada de enlace implementam as seguintes técnicas: ARQ (*Automatic Repeat reQuest*), FEC (*Forward Error Correction*) ou ambas. Os pacotes transmitidos em um canal com fading podem ocasionar problemas para os protocolos de roteamento, os quais podem concluir que existe um novo vizinho com somente um hop. Este vizinho poderia prover uma rota mais curta mesmo para nós mais distantes.

2.5.2 Estações escondidas e expostas

Nas redes Ad Hoc, as estações podem utilizar-se de mecanismos na camada física para detecção de portadora ("carrier-sensing") . Estes mecanismos não resolvem completamente o problema das estações escondidas e das estações expostas (TOBAGI; KLEINROCK, 1975). Uma situação típica de terminais escondidos é mostrada na Figura 2.5. As estações A e C tem um quadro para transmitir para estação B. A estação A não pode detectar a transmissão de C porque ela está fora da faixa da transmissão de C. A estação C está portanto escondida para a estação A. Desde que as áreas de transmissão de A e C não estão disjuntas, haverá colisão em B. Estas colisões tornam as transmissões de A e C para B problemáticas. Para aliviar estes problemas foram introduzidos mecanismos de virtual carrier sensing (IEEE. . . ,), (BHARGHAVAN A. DEMERS; ZHANG, 1994), os quais são baseados em um two-way handshaking que precedem a transmissão dos dados. Entretanto, alguns estudos (FU et al., 2005), (XU; BAE, 2003) apontam que este problema pode persistir nas redes Ad Hoc IEEE 802.11 mesmo com a utilização dos mecanismos de RTS/CTS handshake. Isto se deve ao fato de que a potência necessária para a interrupção de um pacote é muito menor que a potência necessária para a entrega do pacote com sucesso. Ou seja, a faixa de transmissão dos nós é menor que a faixa de sensibilidade dos nós.

O problema das estações expostas resulta do fato que a transmissão tem de ser atrasada devido à transmissão entre outras duas estações dentro da mesma faixa de transmissão da primeira estação. Na Figura 2.6 (HANBALI; ALTMAN, 2005) é mostrado um cenário típico onde ocorre o problema de estações expostas. Se assumirmos que A e C estão dentro da faixa de transmissão de B, e A está fora da faixa de transmissão de C. E também que B está transmitindo para A, e C tem um quadro para transmitir para D. De acordo com o mecanismo de carrier sense, C "sente"que o canal está ocupado devido à transmissão de B. Entretanto, a estação C irá diminuir a transmissão para D, embora esta

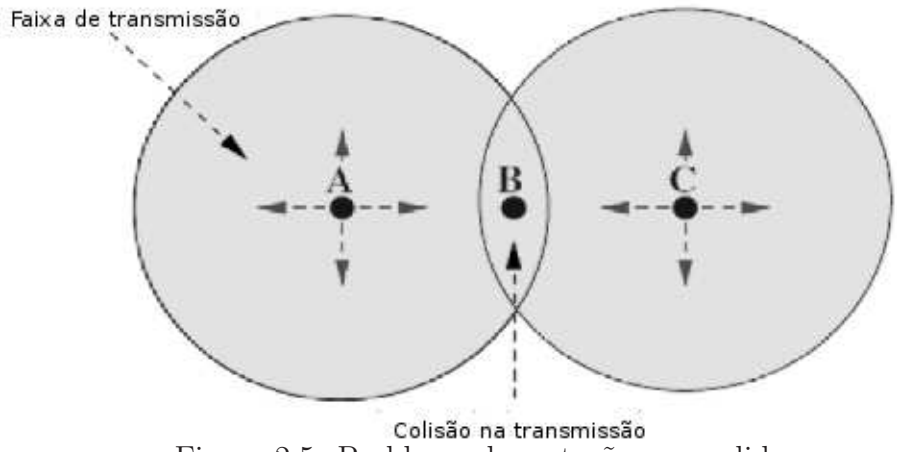

Figura 2.5: Problema das estações escondidas

transmissão não cause interferência em A. O problema da estação exposta pode resultar na redução da utilização do canal.

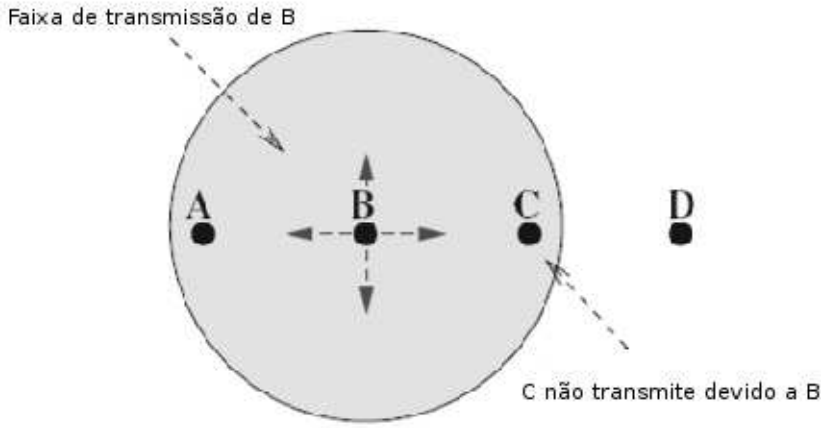

Figura 2.6: Problema das estações expostas

Ambos o problema das estações escondidas quanto das estações expostas estão relacionados com a faixa de transmissão. Através do incremento da faixa de transmissão o problema das estações escondidas ocorre com menor freqüência, em contrapartida, o problema das estações expostas se intensifica.

2.5.3 Assimetria no percurso

O problema de assimetria no trajeto (path asymmetry) pode aparecer em várias formas como assimetria na largura de banda, assimetria na taxa de perda e assimetria na rota. Assimetria na largura de banda: redes de satélites sofrem devido à alta assimetria

na largura de banda, resultando em vários problemas de engenharia (tais como potência, massa e volume), bem como o fato de que para as missões científicas no espaço a maior parte dos dados originados sai do satélite para a Terra. O enlace de retorno geralmente não é usado para transferência de dados. No broadcast nas redes de satélite a taxa da largura de banda no sentido satélite-Terra para o sentido Terra-satélite é de aproximadamente 1000 (DURST; MILLER; TRAVIS, 1997). Nas redes Ad Hoc, por outro lado, o grau de assimetria na largura de banda não é muito alto. No caso do IEEE 802.11g (IEEE...,) a relação da largura de banda fica entre 2 e 54. A assimetria resulta do uso de diferentes taxas de transmissão. Por causa das diferentes taxas de transmissão, mesmo um caminho simétrico entre a fonte e o destino pode sofrer de assimetria na largura de banda. Assimetria na taxa de perdas: este tipo de assimetria ocorre quando ocorrem mais perdas em um sentido do que no outro. Em redes Ad Hoc, esta assimetria ocorre devido ao fato que a perda de pacotes depende das limitações locais que podem variar de local para local. A assimetria na taxa de perdas pode ocasionar também assimetria na largura de banda. Assimetria na rota: diferentemente das formas de assimetria anteriores, onde o caminho de ida e volta pode ser o mesmo, a assimetria na rota implica em caminhos distintos para serem usados nos dados e nos ACKs no protocolo TCP. Esta assimetria pode ser em decorrência do protocolo de roteamento. A assimetria na rota aumenta os overheads no roteamento e a perda de pacotes no caso de um alto grau de mobilidade. Devido aos movimentos nos nós, o uso de rotas distinto em sentidos oposto aumenta a probabilidade de falhas nas conexões TCP.

2.5.4 Particionamento da rede

Uma rede Ad Hoc pode ser representada por um simples grafo G. As estações móveis são os vértices. Uma transmissão bem sucedida entre duas estações é como uma aresta. O particionamento da rede ocorre quando G é desconectado. A maior causa das desconexões em redes MANET é a mobilidade dos nós. Outros fatos que pode levar a um particionamento da rede são as limitações de operação dos nós com relação à energia. Na Figura 2.7 (HANBALI; ALTMAN, 2005) é mostrado um exemplo de particionamento na rede. As linhas tracejadas indicam os enlaces entre os nós. Quando o nó D se move para longe do nó C isto resulta no particionamento da rede em duas componentes separadas. Portanto, o agente TCP do nó A não pode receber o ACK transmitido pelo nó F. Se a desconexão persistir por um período maior que o tempo de retransmissão (RTO) do nó A, o agente TCP irá disparar o algoritmo backoff exponencial (PAXSON; ALLMAN, 2000),

o qual consiste em dobrar o valor de RTO sempre que ocorra o estouro do temporizador. Originalmente o TCP não tem a indicação do momento exato da reconexão da rede. Esta falta de indicação pode levar a longos períodos de idle durante o qual a rede é conectada novamente, mas o TCP ainda está no estado de backoff.

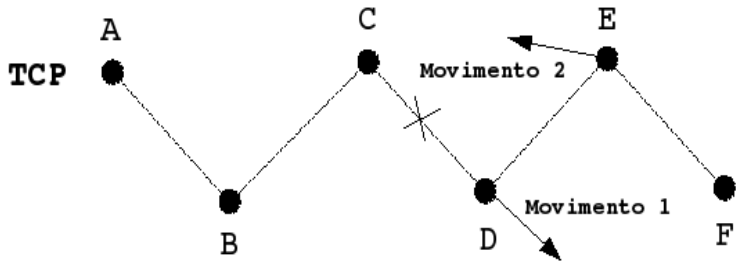

Figura 2.7: Cenário com Partionamento da Rede

2.5.5 Falhas na Rota

Em redes cabeadas falhas na rota ocorrem raramente. Em redes MANET elas são eventos freqüentes. A principal causa de falhas na rota é a mobilidade dos nós. Outros fatos que pode levar a falhas na rota são os erros nos links devido à concorrência no canal wireless, sendo a principal causa de degradação do desempenho do TCP em redes SANET. A duração do reestabelecimento das rotas depois de uma falha, nas redes Ad Hoc, depende do protocolo de roteamento, padrão de mobilidade dos nós e das características do tráfego. Desde que o TCP não tem indicação sobre o reestabelecimento da rota, a taxa de envio e o atraso na sessão irão sofrer degradações devido ao longo período de idle. Também, se as novas rotas estabelecidas forem mais longas ou mais curtas, em termos de hops, do que as rotas antigas ocorrerá uma variação brutal no RTT. Adicionalmente, em redes Ad Hoc, protocolos de roteamento que se utilizam mensagens Hello em broadcast para detectar vizinhos, podem sofrer problemas relacionadas a zonas cinzas de comunicação. Nestas zonas as mensagens de dados podem ser que não sejam trocadas mesmo que as mensagens Hello em broadcast e as mensagens de controle indiquem que os vizinhos estejam ao alcance. A origem destas zonas está nas taxas de transmissão heterogêneas, na ausência de reconhecimento das mensagens de broadcast, nos pequenos pacotes da mensagem Hello e nas flutuações dos links wireless (LUNDGREN; NORDSTRö; TSCHUDIN, 2002).

2.5.6 Limitações de Potência

Por causa das baterias carregadas por cada nó móvel, as quais são limitadas no fornecimento de energia, a utilização de potência é limitada. Esta é a maior questão em redes Ad Hoc, onde cada nó atua como um sistema final e como um roteador ao mesmo tempo, com a implicação de que energia adicional é requerida para o transporte e redirecionamento dos pacotes. O TCP precisa usar de forma eficiente este recurso escasso de energia. Neste caso o termo eficiente se refere a diminuição do número de retransmissões desnecessárias na camada de transporte bem como da camada de link. Em geral há dois problemas correlatos com relação a potência nas redes Ad Hoc: o primeiro é o esquema de power saving que objetiva redução no consumo de energia e o segundo é o power control que objetiva o ajuste da transmissão de potência para os nós móveis. Estratégias para otimização do consumo de energia têm sido investigada nos vários níveis dos dispositivos móveis, incluindo transmissões na camada física, o sistema operacional e as aplicações (JONES et al., 2001). O controle de energia pode ser usado juntamente com agentes de roteamento ou transporte para prover melhorias no desempenho das redes Ad Hoc (KLEMM; KRISHNAMURTHY; TRIPATHI, 2003), (CHIANG, 2005). Limitações no uso de energia durante a comunicação também podem refletir no esquema de cooperação entre os nós, no modo como os nós podem participar de procedimentos de roteamento e direcionamento de mensagens a fim de economizar energia da bateria.

2.6 Propostas para solução dos Problemas em redes wireless

De acordo com (TIAN; ANSARI, 2005) as soluções para os problemas do TCP podem ser classificadas em termos de ajustes no TCP para aplicações específicas, visando acomodar suas necessidades ou em termos da forma como são feitas as mudanças no algorítmo do TCP para suporte do ambiente de rede heterogêneo. Do ponto de vista das aplicações para redes wireless, existem adaptações para melhorar o desempenho do TCP nas redes wireless mais comuns, tais como: redes de satélite, redes Ad Hoc, LAN wireless e redes celulares. O projeto do wireless TCP considera as características particulares de cada tipo de rede e as suas necessidades. Por exemplo, em redes de satélites há um longo atraso na propagação e em redes Ad Hoc existe a falta de uma infraestrutura. Independente do tipo de rede wireless um grande obstáculo é o problema da alta taxa de

BER. Em redes heterogêneas, o principal objetivo do design do TCP é o de diferenciar a causa de descarte de pacotes, seja ela por congestionamento ou por erros randômicos. Deste modo, o transmissor pode tomar a decisão apropriada em como ajustar a janela de congestionamento.

2.6.1 Redes de Satélite

Os esquemas TCP para redes de satélites são baseados na observação que o transmissor leva um longo tempo em uma conexão TCP para atingir altas taxas de envio durante o uso da tradicional fase slow start. Enquanto muitas aplicações tais como http são baseadas na transmissão de pequenos arquivos, com o longo atraso de propagação do link de satélite pode acontecer que a transferência inteira ocorra dentro da fase slow start e a conexão não chegue a atingir a utilização máxima disponível na largura de banda da rede. O TCP-Peach (AKYILDIZ; PALAZZO, 2001) emprega dois novos mecanismos, sudden start e rapid recovery, em combinação com os algoritmos tradicionais do TCP congestion avoidance e fast retransmit. O mecanismo de sudden start introduz o uso de pacotes dummy, os quais são cópias dos últimos pacotes que o transmissor enviou. O transmissor envia múltiplos pacotes dummy entre dois pacotes consecutivos de dados, sendo que a recepção dos ACKs referente aos pacotes dummy indica a disponibilidade dos recursos da rede, deste modo o transmissor tem a indicativa de que ele pode aumentar sua janela de envio rapidamente. Os pacotes dummy são rotulados com uma prioridade menor, portanto um roteador poderia descartá-los primeiro em caso de congestionamento. Simulações mostram que com o mecanismo de sudden start a largura de banda disponível pode ser alcançada dentro de dois RTT, enquanto que o mecanismo tradicional do TCP Reno levaria cerca de sete RTT para alcançar a mesma taxa. O mecanismo de rapid recovery substitui o mecanismo clássico de fast recovery com o objetivo de melhorar o throughput na presença de uma alta taxa de erros. Com este mecanismo, ao invés de tentar distinguir as perdas por congestionamento ou por erros, o transmissor utiliza-se dos pacotes dummy, com prioridade menor, para preencher os pacotes de dados e inflar o tamanho da janela de transmissão.

2.6.2 Redes Ad Hoc

Em uma rede Ad Hoc as altas taxas de erros, mudanças freqüentes na rota e particionamentos na rede são comuns. Estas características resultam em perdas de pacotes

além do congestionamento na rede, devendo ser, portanto, tratados de maneira diferente. O ATCP (LIU; SINGH, 2001) foi proposto como uma solução fim a fim para melhorar o throughput TCP neste ambiente. Ele implementa uma fina camada a qual é inserida entre o TCP padrão e a camada IP. Ele confia na notificação explícita de congestionamento (ECN) para detectar congestionamentos e distinguir as perdas por congestionamentos das perdas por erros. Ele utiliza-se da mensagem ICMP Destination Unreachable para detectar uma mudança de rota ou um particionamento temporário em uma rede Ad Hoc. De acordo com as respostas da rede, a camada ATCP coloca o transmissor corretamente nos estados de persistência, controle de congestionamento ou retransmissão. Quando a perda de pacote é devido a um alto BER, o ATCP retransmite o pacote, portanto o TCP não executa o controle de congestionamento. Se a mensagem ICMP indica uma rota mudada ou particionamento da rede, o ATCP coloca o transmissor no estado de persistência e esperando a reconexão da rota. O ATCP também reoordena os pacotes para evitar que o TCP gere os ACKs duplicados. Quando o ECN indica um congestionamento real, o TCP entra no estágio normal de controle de congestionamento. Através da inserção da camada de ATCP entre o TCP a camada IP, não há modificação do código do TCP, adicionalmente, a camada ATCP não gera pacotes de ACK por conta própria mantendo a semântica fim a fim do TCP.

2.6.3 Redes Celulares

Em redes celulares, onde a estação base interconecta uma rede rápida fixa e uma rede mais lenta móvel, modificações no algorítmo do TCP focam nas características da rede celular, tais como handoff e nos problemas comuns de todas as redes wireless, como o alto BER. O Freeze-TCP (AL., 2000) é uma solução fim a fim que provê melhorias no TCP em ambientes móveis. Ele não impõe nenhuma restrição nos roteadores e requer somente modificações no código da unidade móvel ou do lado do receptor. Ele foca na degradação no throughput causada por frequentes desconexões/reconexões devido aos handoffs ou bloqueios temporários do sinal de rádio por obstáculos. Ele assume que a unidade móvel tem conhecimento da potência do sinal de rádio, e portanto pode prever a eminência de desconexões. O receptor Freeze-TCP na unidade móvel configura de maneira pró-ativa o tamanho da janela de anúncio para zero nos pacotes de ACK na iminência de uma desconexão. Então o transmissor seleciona o tamanho da janela para o mínimo, forçando-o para o estado persistente, onde ele cessa de enviar mais pacotes. Para prevenir o transmissor do procedimento de backoff exponencial, quando o receptor detecta

que houve reconexão, o receptor envia vários pacotes de ACKs positivos reconhecendo os últimos pacotes recebidos antes da desconexão. Desta maneira o transmissor pode retornar rapidamente a taxa anterior à desconexão. Para a implementação do Freeze-TCP informações do cross-layer precisam ser trocadas, e a camada do protocolo TCP precisa saber de alguns detalhes de roaming e algoritmos de handoff implementados pela placa de interface de rede NIC (*Network Interface Card*).

Do ponto de vista da implementação, os algorítmos do TCP em redes wireless podem ser divididos em split mode ou fim a fim. Devido a significativa diferença nas características entre as redes cabeadas e as redes wireless, no esquema de split mode a conexão TCP é dividida nas porções wireless e com fio, sendo que os ACKs são gerados separadamente para cada porção. Através deste mecanismo o desempenho na porção com fio é menos afetado pela porção wireless. Do outro lado, o modo fim a fim trata a rota do transmissor até o receptor como um caminho fim a fim, sendo que o transmissor recebe ACKs diretamente do receptor.

2.6.4 Split Mode

A porção com fio das redes heterogêneas é mais confiável do que a porção wireless em termos da capacidade do link e taxa de erros, portanto a transmissão pode sofrer um gargalo no link wireless, o qual é mais lento e com maior taxa de perdas. O split mode tenta blindar a rede fixa da porção wireless através da separação do fluxo de controle no roteador intermediário, ou estação base no caso de redes celulares, de modo que o comportamento da parte wireless tenha um impacto menor sobre a parte da rede fixa. O roteador intermediário se comporta como se fosse um terminal em ambas as porções fixa e wireless. Ambos os host finais se comunicam com o roteador intermediário independentemente, sem conhecimento do outro lado final. O roteador intermediário é reforçado com funcionalidades para coordenar as transações entre as duas porções da rede. Isto é mostrado na Figura 2.8 (TIAN; ANSARI, 2005).

No I-TCP (BAKRE; BADRINATH, 1995) ("Indirect TCP") o MSR (*Mobile Support Router*) conecta o host móvel MH (*Mobile Host*) ao host fixo FH (*Fixed Host*), e estabelece duas conexões TCP separadas com o FH e o MH. O MSR se comunica com o FH em nome do MH. A janela de congestionamento é mantida separada para as partes fixa e wireless. Quando o MH chaveia de célula, um novo MSR retoma a comunicação com o FH. Deste modo o FH é protegido da característica de não confiabilidade da conexão wireless.

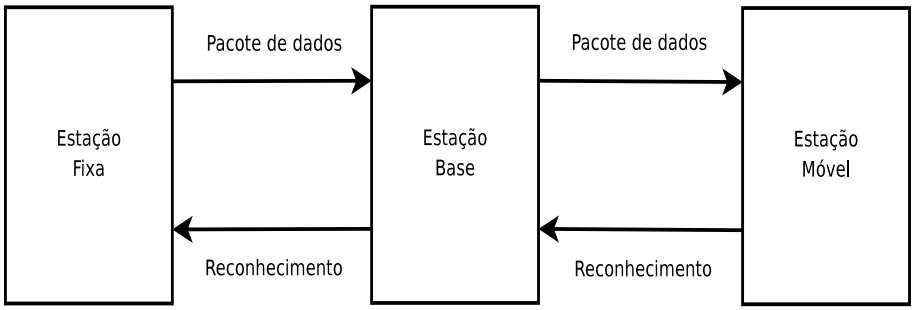

Figura 2.8: Conexão TCP em modo Split

2.6.5 Modo Fim a Fim

No split mode um roteador intermediário tem de analisar as informações do TCP e processar os dados antes que eles cheguem no destino, violando, portanto a semântica fim a fim do TCP original. Na abordagem fim a fim, somente os hosts finais participam do controle do fluxo. O receptor provê indicações refletindo as condições da rede, e o transmissor toma as decisões quanto ao controle de congestionamento. Isto é mostrado na Figura 2.9 (TIAN; ANSARI, 2005).

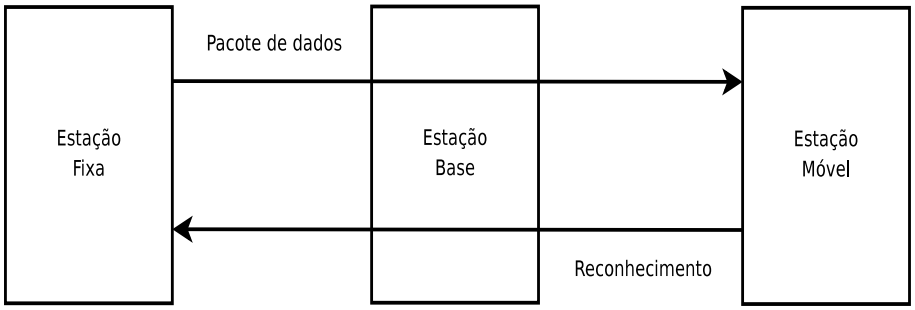

Figura 2.9: Conexão TCP fim a fim

A habilidade para prever com precisão a largura de banda disponível é a chave para melhorar o desempenho, o que ainda é um grande desafio. A disponibilidade de largura de banda de um fluxo significa o mínimo da capacidade do link não utilizada ao longo do caminho e compartilhada de uma forma justa. A abordagem fim a fim pode ter os mecanismos de controle de congestionamento implementados de duas formas: de modo reativo ou pró-ativo. Através do controle de congestionamento reativo o transmissor ajusta a janela de congestionamento quando a rede se torna ruim ou quando um determinado limite é atingido. Através do controle de congestionamento pró-ativo, o transmissor recebe indicações da rede que o guiam para realocar os recursos da rede de forma a prevenir o congestionamento.

Controle de Congestionamento Reativo

O TCP Reno (STEVENS, 1997) padrão emprega controle de fluxo reativo. A janela de congestionamento é ajustada baseada no retorno coletivo dos ACKs positivos e duplicados gerados pelo receptor. O TCP testa a largura de banda disponível incrementando continuamente a janela de congestionamento até que a rede atinja o estado de congestionamento. Neste sentido, o congestionamento é inevitável. Então o TCP irá retornar para uma taxa de transmissão muito mais baixa, a qual pode ser desnecessária em caso de erros randômicos na rede wireless. Muitos esquemas TCP têm sido propostos como um melhoramento do Reno padrão em relação ao seu modo reativo. O algoritmo de fast recovery do Reno se preocupa com o descarte de um único pacote dentro de uma janela. Depois que o pacote é recuperado, o Reno finaliza o procedimento de fast recovery. Devido a natureza das redes wireless, erros correlacionados podem induzir a múltiplos pacotes perdidos, consequentemente o Reno acionaria os mecanismos de fast recovery várias vezes de modo a recuperar os pacotes perdidos. O novo Reno modifica o mecanismo de fast recovery do Reno tradicional para poder tratar os casos com múltiplas perdas dentro de uma janela; isto é uma característica das redes wireless, onde fading no canal pode causar perdas contínuas nos pacotes. No novo Reno (FLOYD; HENDERSON, 1999) o mecanismo de fast recovery não termina até que vários pacotes perdidos, indicados pela recepção de ACKs parciais de uma janela, sejam todos recuperados. A limitação do novo Reno é, entretanto, que ele não pode distinguir a causa da perda dos pacotes, com isso, um modo mais eficiente de fast recovery não pode ser implementado.

TCP SACK (MATHIS J. MAHDAVI; ROMANOW, 1996) é uma opção de ACK seletiva para o TCP, visando o mesmo problema que o Reno tenta resolver. Enquanto o retorno do Reno e do novo Reno é baseado nos ACKs cumulativos, o SACK emprega uma política de retransmissão seletiva. Ele indica um bloco de dados que foram recebidos com sucesso e armazenados na fila do receptor, quando ocorrem perdas de pacotes, ao invés de enviar um ACK parcial, como no novo Reno. Deste modo o transmissor tem um conhecimento melhor sobre o número exato de pacotes que foram perdidos, ao invés de limitar o conhecimento às perdas da borda esquerda da janela somente, como no esquema padrão do TCP. O SACK requer modificações no transmissor e no receptor. Os blocos de SACK são codificados em um campo opcional do TCP, o qual limita o número de blocos SACK que um ACK pode carregar. Além do mais, como no Reno, o SACK responde reativamente as perdas de pacotes, e consequentemente tem uma atuação limitada no controle de congestionamento.

Controle de Congestionamento Pró-Ativo

No controle de congestionamento pró-ativo, o transmissor tenta ajustar a janela de congestionamento pró-ativamente para um valor ótimo de acordo com as informações coletadas via o retorno da rede, a qual reflete a sua condição. Assim, o transmissor reage de maneira mais inteligente as condições da rede ou as perdas de pacotes, devido a congestionamentos ou a erros randômicos, e portanto, prevenindo a rede da entrada em um estado indesejado, tal como congestionamento ou um decremento da janela de congestionamento desnececessário. Diferentes estratégias podem ser empregadas no desenvolvimento para prover ao transmissor condições explícitas sobre a rede.

TCP Vegas (BRAKMO; PETERSON, 1995) estima a reserva de pacotes no buffer em um enlace com gargalos. O Vegas seleciona o mínimo RTT como referência para derivar o throughput ótimo que a rede pode acomodar. Ele também armazena a taxa de envio atual no transmissor durante a transmissão, para derivar o throughput. A diferença entre o throughput ótimo e a taxa atual de envio pode ser usada para derivar a quantidade de reserva de dados na rede. Então dois limites correspondentes aos dois estágios da rede são configurados. Para o primeiro estágio, o Vegas aumenta a janela de congestionamento linearmente e, para o segundo, Vegas decrementa a janela de congestionamento linearmente. Através desre procedimento, TCP Vegas tenta estabilizar o estado de congestionamento na rede ao redor do ponto ótimo através do ajuste pró-ativo da janela de congestionamento sem uma mudança brusca na mesma.

TCP Veno (BRAKMO; PETERSON, 2004) adota a mesma metodologia do TCP Vegas para estimar a quantidade de reserva de pacotes na rede. Adicionalmente ele sugere um modo de diferenciação da causa da perda de pacotes. Se o número de pacotes reservados é menor que um limite, a perda é considerada com randômica, senão, a perda é por congestionamento. Veno adota o mecanismo Reno padrão. Para perdas devidas a erros randômicos, ele incrementa a janela de congestionamento de uma maneira conservadora, ou seja, enviando um pacote para cada ACK recebido.

TCP Westwood (MASCOLO et al., 2001) é baseado na taxa em uma abordagem fim a fim, na qual o transmissor estima dinamicamente a largura de banda disponível da rede através da medição da taxa média de recepção dos ACKs. Ele considera o caminho de retorno ideal, livre de congestionamentos e livre de erros, sendo que a distância entre ACKs reflete os recursos disponíveis na rede. O Westwood emprega um módulo de medição de largura de banda disponível, no lado do transmissor, baseado no intervalo de retorno dos ACKs. Ele calcula a largura de banda explícita e usa ela para dimensionar a taxa de envio. Quando o TCP-Westwood determina que o link está congestionado depois da recepção de três DUPACKs, ele configura o limite de slow start para refletir o produto do atraso

estimado da largura de banda. O TCP-Westwood garante melhorias se comparado ao desempenho do TCP Reno e SACK ao mesmo tempo em que possibilita uma distribuição mais justa. A abordagem fim a fim mantém a estrutura de camadas da rede e requer o mínimo de modificações nos hosts finais, e em alguns casos também nos roteadores.

TCP Jersey (XU; ANSARI, 2004) é um outro esquema pró-ativo que adapta a taxa de envio pró-ativamente de acordo com a condição da rede. Ele consiste de dois componentes chaves, o algoritmo que estima a largura de banda disponível (ABE) e o aviso de congestionamento (CW) configurado no roteador. O ABE é uma adição no lado do TCP transmissor, o qual estima continuamente a largura de banda disponível na conexão e orienta o transmissor no ajuste da taxa de transmissão quando a rede se torna congestionada. CW é uma configuração nos roteadores da rede tal que eles possam alertar as estações finais através da marcação de todos os pacotes quando há uma indicativa de um congestionamento iminente. A marcação dos pacotes pelos roteadores configurados com CW ajuda o transmissor na diferenciação efetiva dos pacotes perdidos por congestionamentos na rede daqueles causados por erros no link wireless. Baseado na indicação de congestionamento dada pelo CW e na estimativa, dada pelo ABE, o TCP Jersey calcula o tamanho ótimo da janela de congestionamento no transmissor. Os roteadores precisam suportar o ECN para implementação do CW.

Os esquemas pró-ativos, em geral, são mais eficientes no tratamento de erros randômicos. A Figura 2.10 (TIAN; ANSARI, 2005) mostra o throughput atingido por um esquema típico pró-ativo (TCP-Jersey) comparado a esquema típico reativo (TCP-Reno). A rede experimental é formada por um link fixo confiável e um link wireless com perdas. A taxa de erros no link wireless é variada de para mostrar como o esquema pró-ativo reage no ambiente wireless em comparação como o esquema reativo. Ambos os esquemas sofrem degradação com o aumento da taxa de erros do link wireless. Entretanto, o TCP Jersey, o qual representa a abordagem pró-ativa, supera o desempenho do TCP Reno padrão em uma taxa de erros mais elevada. Por exemplo, com uma taxa de erros de 2 porcento, o TCP Jersey é 280 porcento mais eficiente que o TCP Reno. Isto se deve principalmente ao fato que os esquemas pró-ativos são mais eficientes na diferenciação de perdas randômicas de congestionamentos do que os esquemas reativos. Deste modo há menos decrementos da janela de congestionamento na transmissão.

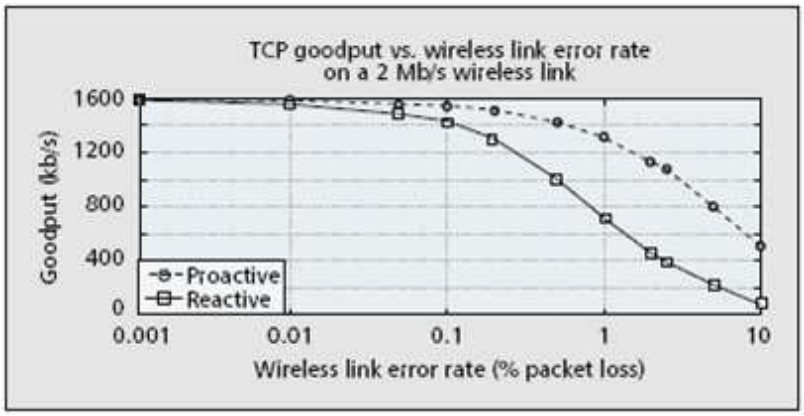

Figura 2.10: Desempenho de um esquema pró-ativo (TCP-Jersey) e um esquema reativo (TCP-Reno) no ambiente de rede sem fio

2.7 Propostas para solução dos Problemas em redes Ad Hoc

Nos sub-capítulos abaixo serão apresentadas propostas para soluções de problemas específicos para o ambiente de redes Ad Hoc.

2.7.1 Propostas para distinguir perdas devido a erros na rota e congestionamento

De acordo com (HANBALI; ALTMAN, 2005) as propostas que tentam resolver o problema do TCP com relação a sua incapacidade de distinguir entre perdas devido a falhas na rota e congestionamentos nas redes móveis Ad Hoc, podem ser agrupadas em duas categorias: propostas para a camada de intersecção entre o TCP e camada de rede (cross layer proposals) e propostas somente na camada TCP. Esta classificação é mostrada na Figura 2.11 (HANBALI; ALTMAN, 2005).

Cross Layer TCP e Rede

TCP-F (*TCP feedback*) (CHANDRAN et al., 1998) é uma abordagem baseada na resposta para manipular erros de rota em MANETs. Esta abordagem permite ao transmissor TCP distinguir entre perdas devido a falhas na rota e aquelas devido ao congestionamento na rede. Quando o agente responsável pelo roteamento de um nó detecta a ruptura de uma rota, ele envia explicitamente um pacote RFN (*Route Failure Notification*) para a origem. Na recepção de um RFN, a origem vai para um estado de inatividade. O TCP transmissor no estado de inatividade irá para o envio de pacotes, e congelará todas as

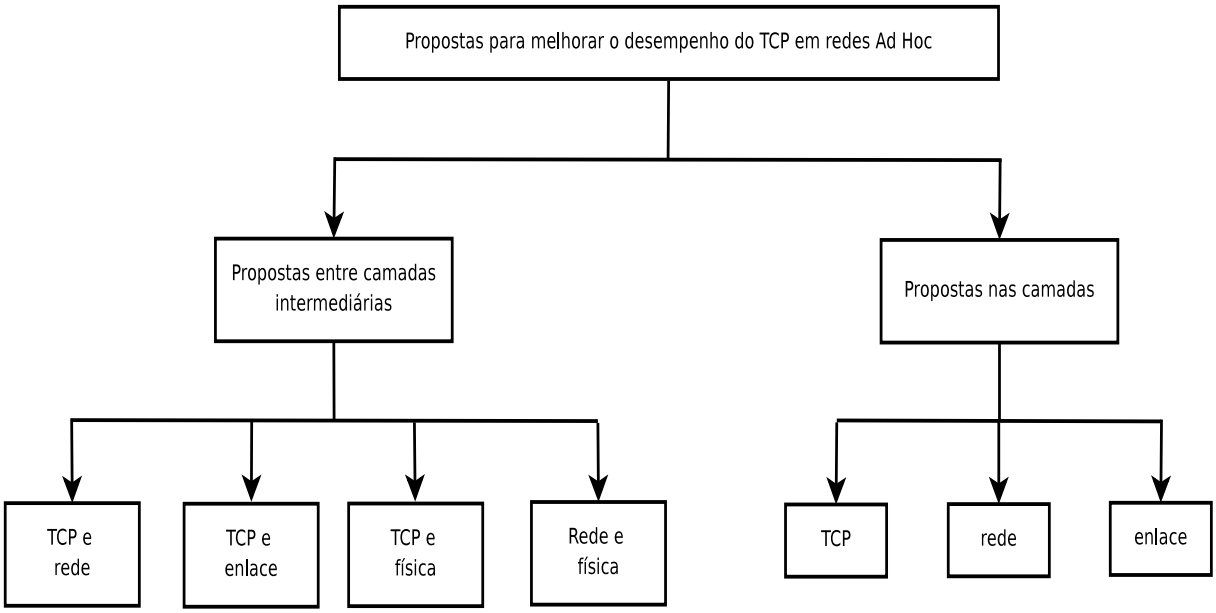

Figura 2.11: Classificação das Propostas para melhorar o desempenho do TCP em redes Ad Hoc

suas variáveis, tais como temporizadores e tamanho da janela de congestionamento. O transmissor permanece neste estado até que ele seja notificado da restauração da rota através de um pacote RRN (*Route Re-establishment Notification*). Após a recepção do RRN, o transmissor deixará o estado de inatividade e continuará a transmissão de pacotes baseada nos valores anteriores da janela e temporizadores. Para evitar que o transmissor fique bloqueado no estado de inatividade, o transmissor dispara um temporizador assim que ele recebe um RFN. Quando este temporizador estoura o algoritmo de controle de congestionamento é chamado normalmente.

ELFN (*Explicit Link Failure Notification*) (HOLLAND; VAIDYA, 1999) é similar ao TCP-F. Entretanto, em contraste ao TCP-F, a implementação da proposta é baseada na interação real entre o TCP e o protocolo de roteamento. Esta interação objetiva informar o agente TCP sobre falhas na rota quando elas ocorrem. Na implementação é usada uma mensagem ELFN, a qual é encapsulada na mensagem de falha da rota, enviada pelo protocolo de roteamento ao transmissor. A mensagem ELFN é como uma mensagem ICMP com destino inalcançável, a qual contém os endereços e portas do transmissor e do receptor, e o número de sequencia dos pacotes TCP. Na recepcão das mensagens ELFN, a origem desliga os temporizadores de retransmissão e entra em um modo de espera. Durante o período de espera, o transmissor verifica a rede para constatar se a rota foi reestabelecida. Se a resposta do pacote de verificação é recebida, o transmissor deixa o estado de espera e retoma os temporizadores de retransmissão e continua a operação

normalmente.

ATCP (*Ad Hoc TCP*) (LIU; SINGH, 2001) utiliza o retorno da rede também. Em adição as falhas na rota, o ATCP tenta tratar o problema de BER. O transmissor pode ir para os estados de persistência, controle de congestionamento ou retransmissão. Uma camada chamada ATCP é inserida entre o TCP e o IP nos nós transmissores. O ATCP escuta verifica o estado da rede baseada na informação provida pela mensagem ECN (RAMAKRISHNAN; BLACK, 2001) e pela mensagem ICMP "Destination Unreachable", então o ATCP coloca o agente TCP no estado apropriado. Na recepção da mensagem ICMP "Destination Unreachable"o agente TCP vai para o estado persistente. Durante este estado o agente TCP é congelado e nenhum pacote é enviado até que uma nova rota seja encontrada através do teste do estado da rede. O ECN é usado como mecanismo para notificar explicitamente o transmissor sobre o congestionamento na rede ao longo da rota em uso. Na recepção do ECN, o controle de congestionamento do TCP é disparado normalmente sem a espera por um evento de estouro de temporizador. Para detectar perdas de pacotes devido a erros no canal, o ATCP monitora os ACKs recebidos. Quando o ATCP verifica que três ACKs duplicados são recebidos, ele não encaminha o terceiro ACK duplicado mas coloca o TCP no estado persistente e rapidamente retransmite o pacote perdido consultando o buffer do TCP. Depois da recepção do próximo ACK, o ATCP irá voltar o TCP para o estado normal. O ATCP permite a interoperabilidade com TCP de origem e destino que não implementem o ATCP. Além de falhas nas rotas, o ATCP tenta tratar os problema de BER, congestionamento da rede e ordenação dos pacotes. Esta vantagem faz do ATCP uma proposta mais robusta para o TCP em redes MANET, mas algumas premissas, tais como nós com suporte ao ECN, bem como disponibilidade permanente do nó transmissor é algo difícil de encontrar no contexto das redes Ad Hoc. Também, o mecanismo de verificação usado para detectar reestabelecimento das rotas gera problemas em caso de alta carga, como na proposta do ELFN.

TCP-BuS (*TCP Buffering capability and Sequence information*) (KIM; CHOI, 2001), como na proposta anterior, usa o retorno da rede a fim de detectar eventos de falha na rota e reagir da forma mais conveniente. O novo esquema é baseado na introdução do buffering capability nos nós móveis. É utilizado o protocolo de roteamento ABR (*Associativity-Based Routing*) (TOH, 1997). As seguintes melhorias são propostas:

- Notificação explícita: duas mensagens de controle são usadas para notificar a origem sobre falhas na rota e sobre o reestabelecimento da rota. Estas mensagens são chamadas ERDN (*Explicit Route Disconnection Notification*) e ERSN (*Explicit Route*

Successful Notification). Na recepção do ERDN do nó que detectou a falha na rota, chamado de nó pivo PN (*Pivoting Node*), o transmissor cessa o envio de mensagens. E, de modo similar, depois do reestabelecimento da rota pelo PN usando um LQ (*Localized Query*), o PN irá transmitir o ERSN para a origem. Na recepção do ERSN, a origem continua a transmissão dos dados.

- Extendendo os valores de timeout: durante a fase de RRC (*Router ReContruction*), pacotes ao longo do caminho, da origem para o PN são bufferizados. Para evitar eventos de temporização durante a fase de RRC, o valor do tempo de retransmissão para os pacotes bufferizados é dobrado.

- Solicitação de Transmissão Seletiva: como o valor do tempo de retransmissão é dobrado, os pacotes perdidos ao longo do caminho da origem ao PN não são retransmitidos até que tempo de retransmissão ajustado expire. Para superar isto, uma indicação é feita na origem, de modo que a origem possa retransmitir os pacotes perdidos seletivamente.

- Evitando solicitações desnecessárias do mecanismo de fast retransmission: quando a rota é restaurada, o destino notifica a origem sobre os pacotes perdidos ao longo do caminho entre o PN e o destino. Na recepção desta notificação, a origem simplesmente retransmite aqueles pacotes perdidos. Mas os pacotes bufferizados ao longo do caminho entre a origem e o PN podem chegar ao destino antes dos pacotes retransmitidos. Então, o destino irá responder com ACK duplicados. Estas solicitações desnecessárias de pacotes para a retransmissão rápida são evitadas.

- Mensagens de Controle com retransmissão confiável: a fim de garantir a exatidão das operações do TCP-BuS, é proposta a transmissão confiável das mensagens de controle ERDN e ERSN. A transmissão confiável é garantida pelo escuta do canal depois da transmissão das mensagens de controle. Se o nó enviou a mensagem de controle mas não ouviu esta mensagem sendo propagada corretamente durante um tempo, ele irá concluir que a mensagem de controle foi perdida e irá transmitir a mensagem.

Esta proposta introduz muitas técnicas novas para a melhoria do TCP. As novas contribuições são as técnicas de bufferização e as mensagens de controle com transmissão confiável.

Propostas na camada TCP

RTO fixo: Esta técnica (DYER; BOPPANA, 2001) é baseada no transmissor que não trata o retorno da rede. Na verdade é empregada heurística para distinguir entre falhas na rota e congestionamento. Quando dois timeouts ocorrem em seqüência, o que corresponde à situação onde o ACK faltante não é recebido antes do segundo estouro do RTO, o transmissor conclui que um evento de falha na rota ocorreu. O pacote não reconhecido é retransmitido mas o RTO não é dobrado uma segunda vez. Isto está em contraste com o TCP padrão, onde o algoritmo de backoff exponencial é usado. O RTO permanece fixo até que a rota seja restaurada e os pacotes retransmitidos sejam reconhecidos.

TCP DOOR (*TCP Detection of Out-of-Order and Response*) é uma abordagem fim a fim (WANG; ZHANG, 2002). Esta abordagem, a qual não requer a cooperação de nós intermediários, é baseada em eventos entregues fora de ordem OOO (*Out-of-order*). Eventos OOO são interpretados como uma indicação de uma falha de rota. A detecção de eventos OOO é acompanhada ou por mecanismo baseado no transmissor, ou baseado no receptor. O mecanismo baseado no transmissor usa a propriedade não decremental do número de seqüência dos ACKs para detectar os eventos OOO. NO caso de pacotes duplicados de ACK, esses ACK terão o mesmo número de seqüência, então o transmissor precisa de uma informação adicional para detectar os eventos OOO. Esta informação adicional é um campo opcional de um byte adicionado ao ACK e chamado ADSN (*ACK Duplication Sequence Number*). O ADSN é incrementado e transmitido em cada ACK duplicado. Entretanto, o mecanismo baseado no receptor precisa de campo adicional de dois bytes no TCP para detectar os eventos OOO chamado TPSN (*TCP Packet Sequence Number*). O TPSN é incrementado e transmitido a cada pacote TCP incluindo os pacotes retransmitidos. Se o receptor detecta um evento OOO, ele deve notificar o transmissor através de um bit específico, chamado bit OOO e contido no cabeçalho do pacote ACK. Uma vez que o transmissor sabe sobre um evento OOO, ele toma as seguintes ações de resposta: desabilita temporariamente o controle de congestionamento e recuperação instantânea durante o congestion avoidance. Na primeira ação, o transmissor desabilita o algoritmo de controle de congestionamento por um período de tempo específico (T1). Na ação posterior, se o algoritmo de controle de congestionamento foi chamado durante o período de tempo decorrido (T2), o transmissor deve voltar imediatamente para o estado anterior a chamada do controle de congestionamento. De fato, os autores fazem os períodos de tempo T1 e T2 funções do RTT.

De acordo com (HANBALI; ALTMAN, 2005) as seis propostas anteriores combatem o problema da incapacidade do TCP de distinguir entre perdas devido a falhas na rota e congestionamento na rede.

2.7.2 Propostas para reduzir erros na rota

As propostas que atacam os problemas de falhas freqüentes na rota em redes MANETs podem ser classificadas em três categorias: cross layer TCP e rede, cross layer rede e física e camada de rede.

subsubsectionCross layer TCP e Rede Split TCP: conexões TCP com um grande número de saltos sofrem com freqüentes falhas na rota devido à mobilidade. Para melhorar o throughput destas conexões e resolver o problema da injustiça, foi introduzido o Split TCP para dividir conexões TCP longas em segmentos curtos localizados (KOPPARTY S. KRISHNAMURTHY; TRIPATHI, 2002), conforme Figura 2.12 (HANBALI; ALTMAN, 2005).

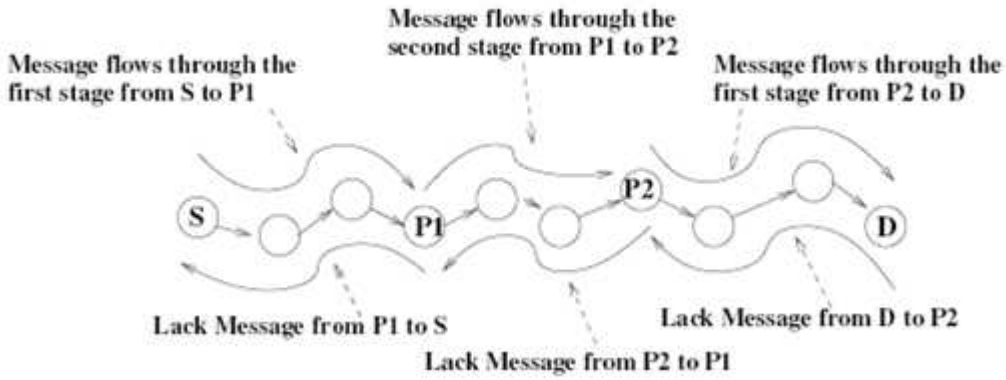

Figura 2.12: Split TCP: Divisão da conexão TCP em segmentos

O nó de interface entre dois segmentos é chamado proxy. O Agente de roteamento decide se seu nó tem o papel de proxy de acordo com o parâmetro de distância inter-proxy. O proxy intercepta os pacotes TCP, bufferiza eles, e reconhece sua recepção para a origem através do envio de mensagens de reconhecimento local LACK (*Local Acknowledgement*). O proxy também é responsável pela entrega dos pacotes com uma taxa apropriada para o próximo segmento local. Na recepção de um LACK (do próximo proxy ou de um destino final) o proxy apagará o pacote de seu buffer. Para garantir a confiabilidade da origem para o destino, um ACK é enviado do destino para a origem, do mesmo modo que o TCP padrão. Este esquema realmente divide as funcionalidades da camada de transporte em confiabilidade fim a fim e controle de congestionamento. Isto é realizado através de duas janelas de transmissão na origem, as quais são designadas janela de congestionamento e janela fim a fim. A janela de congestionamento é uma sub-janela da janela fim a fim. Enquanto a janela de congestionamento muda de acordo com a taxa de chegada dos LACKs do próximo segmento, a janela fim a fim mudará de acordo com a taxa de chegada

dos ACKs vindos do destino. Em cada proxy, haverá uma janela de congestionamento que controlará a taxa de envio entre os proxies.

subsubsectionCross Layer Camada de rede e Física Roteamento preemptivo: em redes MANET o TCP pode sofrer longos períodos de idle induzidos por falhas frequentes na rota. Esta proposta (GOFF et al., 2003) ataca este problema pela redução do número de falhas na rota. Adicionalmente, ele também reduz a latência na reconstrução da rota. Isto é alcançado pela mudança para uma nova rota quando um link da rota atual está na iminência de uma falha. Esta técnica é agrupada com o protocolo de roteamento sobre demanda AODV e DSR. O mecanismo de detecção de falha é baseado na potência. Mais especificamente, quando um nó intermediário ao longo de uma rota detecta que a potência do sinal de um dado pacote recebido de outro nó cai abaixo de um valor, chamado limite de preempção, este nó intermediário irá detectar uma falha na rota. Como exemplo, na Figura 2.13 (HANBALI; ALTMAN, 2005), quando o nó 4 sente que a potência do sinal de um pacote recebido do nó 5 cai abaixo do limite de preempção, ele detectará o evento de falha de roteamento. Na detecção deste evento, o nó 4 irá notificar a origem da rota, o nó 1. Na recepção desta notificação, o agente de roteamento da origem procura uma nova rota pró-ativamente. Quando a nova rota está disponível, o agente de roteamento muda para a nova rota. Quando o valor do limite de preempção parece ser crítico, como no caso de um limite com um valor baixo, não haverá tempo suficiente para a descoberta de um caminho alternativo antes que a rota falhe. Também, no caso de um valor mais alto, uma mensagem de aviso será gerada antecipadamente. Para tratar as flutuações na potência do sinal recebido devido ao fading do canal e efeitos multipercurso, os quais poderiam disparar um aviso preemptivo da rota e causar inundações de solicitações desnecessárias de rota, é utilizada uma mensagem curta de teste, a qual é repetida várias vezes para testar se a mensagem de aviso está correta. Também na Figura 2.13, quando a potência do sinal do pacote enviado do nó 5 e recebido no nó 4 cai abaixo do limite de preempção, o nó 4 envia uma mensagem de ping para o nó 5. Na recepção desta mensagem, o nó 5 replica com uma mensagem de pong. Na recepção da mensagem de pong o nó 4 verifica a potência do sinal da mensagem de pong para testar se o link 4-5 está na iminência de falhar. O teste de ping-pong é repetido várias vezes.

Força do sinal baseada no gerenciamento do link: este algoritmo (KLEMM; KRISHNAMURTHY; TRIPATHI, 2003) é parecido com o anterior. Entretanto, neste algoritmo cada nó mantém um histórico da potência do sinal recebido dos nós vizinhos com até 1 salto. Usando este histórico, o protocolo de roteamento prediz a queda do link no futuro (depois de 0.1 segundos). Esta predição é chamada de gerenciamento pró-ativo do link.

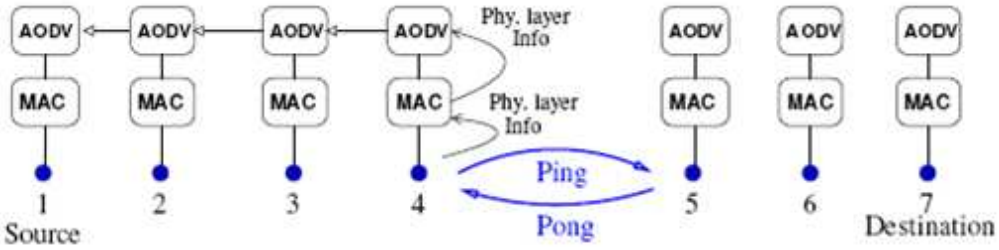

Figura 2.13: Cross Layer: camada de rede e física

Na detecção deste evento, o agente de roteamento da origem é notificado por uma mensagem. Na recepção desta mensagem o agente de roteamento da origem para de enviar pacotes e inicia o procedimento de descoberta de rotas. A novidade desta proposta é o mecanismo de gerenciamento reativo do link. Este mecanismo incrementa a potência da transmissão para restabelecer um link quebrado. Mecanismos de gerenciamento de link reativos e pró-ativos podem ser utilizados da seguinte maneira: na predição que um link está na iminência de cair, o agente de roteamento do nó notifica a origem para cessar o envio de pacotes, e incrementa sua potência de transmissão para tratar pacotes que ainda estejam transitando através do link.

subsubsectionCamada de rede

Backup path routing: Esta proposta (LIM; GERLA, 2003) visa melhorar a disponibilidade do percurso nas conexões TCP utilizando-se de roteamento multipercurso. Os autores acharam que o roteamento multipercurso original deteriora o desempenho do TCP. Isto se deve ao erro durante a medição da média para o cálculo do RTT e a entrega dos pacotes fora de ordem. Com isso, eles introduziram uma nova variante de roteamento multipercurso, chamada backup path routing. Esta nova variante mantém vários percursos da origem ao destino, mas ela usa somente um caminho por vez. Quando o caminho atual para, ele pode ser rapidamente comutado para um caminho alternativo. Usando simuladores, os autores observaram que mantendo um caminho primário e um outro alternativo para cada destino faz com o TCP apresente uma melhoria de desempenho. São considerados dois esquemas como critério de seleção dos caminhos. O primeiro esquema consiste da seleção do caminho mais curto como primário e o caminho com o menor atraso como o alternativo. O segundo esquema consiste da seleção do caminho com menor atraso como primário e o caminho mais separado possível como alternativo. Ou seja, o caminho alternativo deve ter o menor número de nós intermediários em comum com o caminho primário. Comparando os dois esquema, verificou-se que o primeiro é superior ao segundo devido ao fato que o segundo esquema tende a ter uma quantidade maior de saltos, sendo

portanto mais longo.

As quatro propostas apresentadas anteriormente visam atacar o problema das falhas freqüentes nas rotas, as quais levam a um longo tempo de *idle*. A principal causa de falhas na rota nas redes MANETs é a mobilidade.

2.7.3 Propostas para reduzir a concorrência no canal wireless

As propostas que atacam o problema da concorrência nos canais wireless em redes SANET podem ser classificadas em três categorias: propostas na camada TCP, propostas na camada de rede e propostas na camada de enlace.

Camada TCP

Dynamic delayed Ack: Esta abordagem (ALTMAN; JIMéNEZ, 2003) visa reduzir a concorrência no canal wireless através da diminuição do número de ACKs do TCP transmitidos pelo sink. Ela consiste de uma modificação na opção delayed ACK (RFC1122), a qual tem um coeficiente fixo d=2. O coeficiente d representa o número de pacotes TCP que o sink deve receber antes que ele reconheça estes pacotes. Nesta abordagem o valor de d não é fixado e varia dinamicamente com o número de sequencia do pacote TCP. Por esta razão, três limites são definidos l1, l2 e l3, tais que, d=1 para pacotes com número de sequencia N menor que l1, d=2 para pacotes com $l1 < N < l2$, d=3 para $l2 < N < l3$ e d=4 para $l3 < N$. Nas simulações, foram estudadas a taxa de perda de pacotes, throughput, e atraso na sessão do TCP new Reno, no caso de uma sessão TCP curta e persistente sobre um elo estático multihop. Com os valores l1=2, l2=5 e l3=9, esta proposta supera o TCP padrão e também o TCP com a opção de delayed ACK com coeficientes fixos d=2,3,4. Os autores sugerem que um melhor desempenho pode ser obtido fazendo com que d seja uma função da janela de congestionamento do transmissor ao invés do número de sequencia.

Camada de Rede

COPAS: a proposta Contention-based Path Selection (CORDEIRO; AGRAWAL, 2003) ataca o problema de queda do desempenho do TCP devido a concorrência no canal wireless. Ela implementa duas técnicas: a primeira é uma separação dos sentidos das rotas, o qual consiste da seleção separada de rotas para os pacotes de dados TCP e para os pacotes de ACK. A segunda é um balanceamento dinâmico da concorrência, o que consiste em atualizar dinamicamente as rotas separadas. Uma vez que a concorrência na rota excede um certo limite, chamado backoff threshold, uma nova rota e com menor concorrência é

selecionada para substituir a rota com maior concorrência. Nesta proposta a concorrência no canal wireless é medida como função do número de vezes que um nó executa o procedimento de back off durante cada intervalo de tempo. Também se em algum momento a rota é quebrada, adicionalmente ao procedimento de iniciar o seu restabelecimento, COPAS redireciona os pacotes usando a rota alternativa. COPAS supera o DSR em termos de throughput do TCP e overheads no roteamento. COPAS consegue uma melhora no throughput de 90%. Entretanto, o uso do COPAS é limitado a redes estáticas ou redes com baixa mobilidade. Como os nós se movem rápido quando usam rotas separadas para envio e recepção, isto aumenta a probabilidade de falhas nas rotas em conexões TCP.

Camada de enlace

Link RED: Link Random Early Detection (FU P. ZERFOS; GERLA, 2003) visa reduzir a concorrência no canal wireless através no monitoramento do número médio de retransmissões (avg) na camada link. Quando avg se torna que um determinado limite, a probabilidade de descarte/marcação é calculada de acordo com o algoritmo RED (FLOYD, 1993). Desde que ele marca os pacotes, link RED pode ser utilizado com ECN para notificar o transmissor sobre o nível de congestionamento (RAMAKRISHNAN; BLACK, 2001). Entretanto, ao invés de notificar o transmissor sobre o nível de congestionamento, ele aumenta o tempo de backoff na camada MAC.

Adaptive pacing: O objetivo desta proposta (FU P. ZERFOS; GERLA, 2003) é melhorar o reuso espacial do canal. No protocolo atual IEEE 802.11, um nó tenta minimizar a concorrência do canal através de um período randômico de backoff, mais um tempo de transmissão de um pacote, o qual é a anunciado pelo quadro RTS ou CTS. Entretanto, o problema do receptor exposto persiste devido a falta de coordenação entre os nós que estão mais de dois saltos distantes entre si. O Adaptative pacing resolve este problema através do aumento do período de backoff de um tempo adicional para transmissão de pacotes. Esta proposta trabalha junto com o link RED, sendo que o Adaptative pacing é habilitado por ele. Quando um nó identifica que o número médio de retransmissões esta abaixo de um limite, ele calcula o tempo de backoff da forma usual. Caso este número ultrapasse este limite, o adaptative pacing é habilitado e o período de backoff é aumentado pela duração do tempo de transmissão do pacote anterior. Trabalhando juntos, Link RED e Adaptative pacing têm mostrado melhoria do throughput bem com a justiça entre múltiplas sessões do TCP. Entretanto, esta proposta do tempo adicional do backoff é baseada no tamanho do pacote. Então a existência de diferentes tamanhos de pacotes de dados na rede deve ser verificado.

O problema da concorrência no canal wireless é a principal causa da degradação do desempenho do TCP em redes SANET.

2.7.4 Propostas para melhorar a justiça do TCP

As propostas que visam resolver o problema da injustiça no TCP em redes SANET são classificadas como propostas na camada link.

Camada de enlace

Non work-conserving scheduling: O objetivo desta proposta (YANG; SEAH; YIN, 2003) é de melhorar a justiça entre fluxos TCP que cruzam redes wireless Ad Hoc e redes cabeadas. Os autores adotam a política "non work-conserving scheduling"em redes Ad Hoc ao invés da "work-conserving scheduling". A fila da camada link aciona um temporizador sempre um pacote é enviado para a camada MAC. A fila somente envia outro pacote para a camada MAC quando o temporizador estoura. A duração do temporizador é atualizada de acordo com a taxa de saída da fila. Especificamente a duração do temporizador é a soma de três partes D1, D2 e D3. D1 é igual ao comprimento do pacote de dados dividido pela largura de banda do canal. D2 é um atraso, sendo que seu valor é decidido pelo valor atual da taxa de saída da fila. D3 é um valor randômico uniformemente distribuído entre 0 e D2. D3 é usado para evitar problemas de sincronização e para reduzir colisões. A fila calcula a taxa de saída pela contagem do número de bytes, C, que ela envia em cada intervalo fixo T. Para decidir o valor de D2, três limites são configurados X, Y e Z ($X < Y < Z$) para C, e quatro valores de atrasos D21, D22, D23 e D24 ($D21 < D22 < D23 < D24$) para D2, como mostrado na equação 2.6 (HANBALI; ALTMAN, 2005). A heurística por detrás desta abordagem é o de penalizar nós com alta taxa de saída através do incremento de seu atraso na fila D2 e favorecer nós com taxas pequenas de saída. Através de simulações os autores reportam que seus esquemas aumentam bastante a justiça entre as conexões TCP ao custo moderado de uma degradação no *throughput* total.

$$\begin{cases} D_2 = D_{21} & \text{para} \quad C \leq X \\ D_2 = D_{22} & \text{para} \quad X < C \leq Y \\ D_2 = D_{23} & \text{para} \quad Y < C \leq Z \\ D_2 = D_{24} & \text{para} \quad C > Z \\ 0 \leq D_{21} < D_{22} < D_{23} < D_{24} \end{cases} \tag{2.6}$$

Neighborhood RED: esta proposta (XU et al., 2003) visa melhorar a justiça do TCP em redes MANET. Diferentemente das redes cabeadas, os autores mostram que o RED não resolve o problema da injustiça do TCP em redes MANET, devido ao fato que o congestionamento não acontece em um único nó, mas em uma área inteira envolvendo múltiplos nós. A fila local de pacotes em qualquer nó não pode refletir completamente o estado de congestionamento da rede. Por esta razão, os autores definem uma nova fila distribuída, chamada neighborhood queue. Em um nó, a neighborhood queue deve conter todos os pacotes cuja transmissão irá afetar sua própria transmissão em adição aos seus pacotes. Devido a dificuldade de se obter informações sobre todos estes pacotes sem introduzir overheads significativos na comunicação, o qual pode precisar trocar informações com até 2 saltos, é introduzida uma fila neighborhood simplificada. Ela agrega a fila local do nó e as filas de upstream e downstream dos nós vizinhos com 1 salto. O algorítmo RED é então baseado no tamanho médio da fila da neighborhood queue. Os autores usam um algorítmo distribuído para calcular o tamanho médio da fila. Neste algorítmo o tempo é dividido em janelas. Durante cada janela de tempo, o período de idle do canal é medido. Usando esta medida, um nó estima a utilização do canal e o tamanho médio da neighborhood queue. A exatidão da estimativa é controlada pela duração das janelas de tempo.

É difícil comparar propostas diferentes que visam melhorar o desempenho do TCP em redes MANET. Algumas delas foram desenvolvidas para resolver diferentes problemas. Através da análise destas propostas quatro problemas principais são identificados: o TCP é incapaz de distinguir entre perdas devido a falhas na rota e congestionamento na rede, falhas freqüentes nas rotas, concorrência no canal wireless e injustiça do TCP. Os dois problemas iniciais são as maiores causas da degradação do desempenho do TCP em redes MANETs, e os outros dois são as principais causas da degradação do desempenho em redes SANETs. Para resolver estes problemas foram utilizadas propostas com soluções usando cross layer ou baseadas em uma camada específica. Em termos de complexidade as soluções cross layer são mais difíceis de serem implementadas, porque elas necessitam da modificação de pelo menos duas camadas OSI. Elas também quebram o conceito do desenvolvimento dos protocolos independente de outras camadas, seu desenvolvimento requer que o sistema seja considerado inteiramente.

A Tabela 2.1 apresenta um resumo das principais variantes TCP adotadas nos principais sistemas atuais e a Tabela 2.2 apresentada um resumo das principais variantes propostas as quais não foram amplamente difundidas. A Tabela 2.4 apresenta a classificação das soluções para melhorias do TCP baseadas na proposta de implementação.

Tabela 2.1: Comparação entre as principais variantes TCP em uso

Solução	Características	O que faz?	O que resolve?	Vantagens	Desvantagens
TCP (RFC793)	orientado a conexão, possui janela de congestionamento, anúncio e transmissão	Implementa camada de transporte	Entrega confiável e seqüenciada dos pacotes.	-	-
TCP Tahoe	análise de um problema de congestionamento ocorrido em 86	Implementa mecanismos de slow-start, congestion avoidance e fast retransmit	Tenta manter o equilíbrio na rede promovendo uma melhor utilização do canal. Promove uma partida mais suave após uma perda.	Melhora desempenho do TCP	-
TCP Reno	Sempre realiza o mecanismo de slow start, mesmo em uma única rede.	Introduziu um mecanismo de fast recovery. Depois que o fast retransmit envia o que parece ser um segmento perdido, o mecanismo de congestion avoidance é executado. Isto é chamado fast recovery.	A fim de não diminuir bruscamente o fluxo após a detecção de pacotes fora de ordem ele realiza o procedimento de congestion avoidance depois do fast retransmit. Ou seja, após a recepção de 3 ACKS duplicados.	Aumenta o throughput quando ocorrem pacotes fora de ordem ou perda de pacotes.	-
TCP New Reno	Foi descoberto durante testes com TCP SACK pois nem todos os problemas são necessariamente conseqüência da falta de SACK.	ACKs parciais não tiram o TCP fora do mecanismo de fast recovery. Ao contrário, eles são tratados com uma indicativa que pacotes imediatamente em seguida aos pacotes reconhecidos no espaço da janela foram perdidos, e devem ser retransmitidos.Então, quando múltiplos pacotes são perdidos em uma única janela de dados, podem ser recuperados sem um timeout, através da retransmissão de um pacote perdido por round-trip time.	Problemas quando múltiplos pacotes são perdidos em uma janela de dados.	Evita o disparo de múltiplos fast retransmit dentro de uma janela de dados quando ocorrem múltiplas perdas.	-
TCP SACK (RFC2018)	Mudanças na RFC1072	Através do SACK, o receptor pode informar o transmissor sobre todos os segmentos que chegaram, então o transmissor retransmite somente os segmentos que foram realmente perdidos.	Problemas quando múltiplos pacotes são perdidos em uma janela de dados.	Retransmite somente os pacotes perdidos de modo seletivo.	Requer mudanças tanto no receptor quanto transmissor e na semântica do TCP.
TCP-WestWood	Modifica o algoritmo de congestionamento de modo a torná-la menos sensível as perdas randômicas.	Tenta selecionar um threshold para o slow start e uma janela de congestionamento consistente com a largura de banda durante o período de congestionamento (faster recovery).	Ameniza os efeitos de falhas randômicas.	Só modifica o transmissor	Algoritmo para monitoramento dos ACKs é mais complicado

Tabela 2.2: Comparação das principais propostas de melhorias do TCP

Solução	Características	O que faz?	O que resolve?	Vantagens	Desvantagens
TCP-Jersey	utiliza a janela de congestionamento (CW) juntamente com um mecanismo de notificação de congestionamento denominado ABE.	É capaz de distinguir perdas de pacotes em ambiente wireless causadas por erros das perdas causadas por congestionamento através de dois componentes, o ABE e o CW.	Promove uma melhor distinção entre perdas devido a congestionamento e perdas devido a falhas.	Aciona o mecanismo de congestion avoidance com maior precisão.	Requer mudanças no transmissor e nos roteadores.
TCP-Peach	utiliza pacotes dummies, os quais são marcados utilizando um bit do header TCP.	Introduz mais dois estados: sudden start e rapid recovery.	Má utilização da largura de banda em redes de satélite	Melhora a utilização do canal em redes de satélite.	específico para transmissões via satélite. Requer mudanças tanto no receptor quanto transmissor
A-TCP	é inserida como uma camada intermediária entre o TCP e o IP	Utiliza as respostas da camada de rede (através de ECN e ICMP) para colocar o TCP em um estado persistente, controle de congestiona mento ou retransmissão.	Problemas do uso do TCP em redes sem fio multihop	Não afeta o funcionamento do TCP padrão	Necessita do suporte de ECN
Freeze-TCP	utiliza informações do mobile host.	Através das informações de nível de sinal do receptor ele manipula as janelas de congestionamento e envio dos ACKS para melhorar o desempenho do TCP	Problemas de freqüentes desconexões.	Não há necessidade de mudanças no transmissor	requer mudanças no receptor
I-TCP	utiliza os recursos do MSR e divide a rede em duas partes: parte fixa e parte móvel.	Permite o desenvolvimento de protocolos de transporte independentes para solucionar os problemas inerentes de cada parte da rede.	Não permite que os problemas da parte wireless interfiram no lado de rede fixo.	O fluxo TCP fica independente entre as duas partes da rede.	Necessita de alterações no host intermediário
TCP-Vegas	Faz melhorias nos mecanismos de fast retransmit, congestion avoidance e mecanismo de slow-start.	Baseado na estimativa do tempo entre a recepção dos segmentos TCP Vegas retransmite rapidamente um pacote perdido sem esperar os 3 DUPACKs.	Possibilita retransmissões mais rápidas, uma melhor utilização da largura de banda e um pequeno aprimoramento no mecanismo de slow-start, utilizando 4 ACKs antes de estimar a largura de banda.	Melhora o desempenho do TCP em várias fases.	Requer algoritmos mais complicados e armazenamento de informações adicionais
TCP-VENO	só modifica o transmissor	Modifica o algoritmo de AIMD. Utiliza algumas das idéias do TCP-Vegas	Diminui o throughput de forma menos drástica que o TCP Reno.	Só modifica o transmissor	Requer algoritmos mais complicados e armazenamento de informações adicionais
TCP-Feedback	visa monitorar os problemas de mudança de rota	Introduz dois tipos de pacotes para indicar falhas na rota: o Route Failure Notification (RFN) e o Route Reestablishment Notification (RRN). Quando uma falha na rota é detectada através do RFN o transmissor vai para um estado de inatividade e permanece até que a rota seja restabelecida. A conexão TCP continua então normalmente.	Problema de mudanças constantes na rota.	Não afeta a conexão TCP em caso de falhas na rota. Indicado para redes Ad Hoc.	Necessita de pacotes adicionais. Requer modificações em todos os nós da rede.
ELF-N	utiliza mensagens ELFN em conjunto com o TCP.	Através das mensagens de ELFN o TCP é desabilitado e entra em um estado de stand-by. Neste estado pacotes de teste são enviados para monitorar o restabelecimento da rede.	Problema de falhas constantes na rota.	Não afeta a conexão TCP em caso de falhas na rota. Indicado para redes Ad Hoc.	Requer modificações em todos os nós da rede para suportar ELFN e no próprio TCP para interagir com o ELFN.

2.8 Conclusão

Nota-se que muitas propostas surgiram para melhorar o desempenho do TCP em redes sem fio. Embora as diversas variantes TCP garantam a melhora em determinadas

Tabela 2.3: Comparação das principais propostas de melhorias do TCP - cont.

Solução	Características	O que faz?	O que resolve?	Vantagens	Desvantagens
TCP-BuS	utiliza o mesmo princípio do TCP-Feedback	Através de mensagens de controle o TCP-BuS altera o esquema de controle de congestionamento do TCP e propõe mudanças visando integração com o protocolo de roteamento ABR. Atua com mecanismos de:notificação explícita, estendendo valores de timeout, transmissão seletiva, evitando solicitações desnecessárias de fast retransmission e implementando mensagens específicas.	Problema de falhas constantes na rota.	Melhora o esquema de detecção de falhas devido a erros e congestionamento e o desempenho geral do TCP.	Requer modificações em todos os nós da rede para suportar ABR (Associativity-Based Routing) e no próprio TCP para interagir com o ABR.
RTO Fixo	Um esquema heurístico foi empregado para distinguir falhas de rota e congestionamento.	Quando ocorrem timeouts consecutivos, o transmissor assume que uma falha na rota ocorreu e não um congestionamento. O pacote sem ACK é retransmitido novamente mas não dobra o RTO na segunda vez. O RTO permanece fixo até que a rota seja restabelecida e o pacote retransmitido seja reconhecido.	Problema de falhas constantes na rota.	Só modifica o transmissor	Não têm um feedback eficiente sobre falhas na rede.
TCP DOOR	É baseado na detecção da entrega de eventos fora de ordem e inferindo mudanças de rota por estes eventos.	Modifica o cabeçalho TCP com a inserção de um campo adicional da seqüência do pacote no receptor, chamado Packet Sequence Number (TPSN), e um campo para a indicação dos eventos OOO no transmissor, chamado Duplication Sequence Number (ADSN). Quando ocorre um evento OOO o transmissor desabilita temporariamente o controle de congestionamento e aciona a recuperação instantânea durante o congestion avoidance.	Problema de mudanças constantes na rota.	Diminui os efeitos de mudanças de rota no desempenho do TCP	Requer mudanças tanto no receptor quanto transmissor e na semântica do TCP.

condições sempre é necessária a mudança nos dispositivos da rede inviabilizando a sua utilização em larga escala.

Devido as diferenças entre as redes sem fio e cabeadas nem sempre é possível garantir o desempenho em ambas as redes por isto algumas abordagens supoem a utilização de diferentes TCP em cada parte da rede.

Com relação a classificação de acordo com a implementação, apresentada na tabela 2.4, o TCP-HPL funciona com uma solução *end-to-end*, com a vantagem de não alterar a sintaxe do TCP padrão.

Tabela 2.4: Classificação das Soluções TCP de acordo com o tipo de implementação

Tipos de solução	**Vantagens**	**Desvantagens**	**Quem usa**
End to End	Não altera o fluxo nos nós intermediários	Requer modificações nos extremos: transmissor e ou receptor.	TCP-Tahoe, TCP-Reno, TCP-New Reno, TCP-SACK, TCP-Jersey, TCP-Peach, Freeze-TCP, TCP-Vegas, TCP-VENO, TCP-WestWood, RTO fixo e TCP DOOR.
Split Mode	Permite adequar cada segmento de rede de acordo com as suas características específicas.	Requer mudanças nos nós intermediários. Altera a semântica original do TCP.	I-TCP
Cross layers	Permite que informações de outras camadas ou nós sejam utilizadas para melhor ajuste do TCP.	Requer modificações na semântica do TCP.	ATCP, TCP-Feedback, ELF-N e TCP-BuS

Capítulo 3

Arquitetura Proposta

O mecanismo proposto consiste na criação de uma camada entre o TCP e o IP através do uso de *raw socket* para recepção de pacotes tanto do TCP originador quanto do receptor. O TCP originador envia os pacotes ao seu destino normalmente, os quais são interceptados pelo *raw socket*; também a resposta enviada pelo receptor é interceptada pelo *raw socket* e enviada para o TCP originador. Em termos práticos, não há nenhuma alteração na sintaxe TCP entre os dois extremos, porque o *raw socket* atua na camada IP. Assim, do ponto de vista do TCP originador e receptor, se trata apenas de uma conexão fim a fim. Isso se torna uma grande vantagem porque não há necessidade de uma versão especial de TCP rodando na máquina que executa as funções do TCP-HPL. Esta camada é inserida em um *proxy* o qual faz a conexão entre a parte de rede cabeada e a rede sem fio.

O *raw socket* permite que os dados do cabeçalho TCP estejam acessíveis, podendo até mesmo serem alterados. Os dados de controle do TCP controlam a máquina de estados do TCP e por isto servem como base para que o TCP-HPL atue diretamente na conexão TCP fim a fim.

Esta proposta visa estar alinhada com os requisitos do projeto (FLUMINENSE, 2008) visto que uma das grandes necessidades é a comunicação sem fio entre médico e paramédico da forma mais eficiente possível.

O paramédico pode a partir de qualquer lugar em que lhe seja possível ter acesso a alguma modalidade de rede sem fio preencher e enviar ao cardiologista, no ato do atendimento de urgência//emergência, um prontuário eletrônico e o resultado de um ECG digitalizado. Desse modo, o sistema AToMS propicia ao paciente com IAM o atendimento altamente especializado de que necessita, no tempo equivalente ao do atendimento de

urgência//emergência tradicional. As informações enviadas pelo paramédico são todas em formato digital. Isso fornece as condições ideais para que o sistema AToMS seja facilmente auditável.

O uso de tecnologia sem fio na comunicação entre paramédicos e cardiologistas permite também que um paciente com IAM, cuja remoção para uma Unidade Coronariana ou de Tratamento Intensivo faz-se necessária, seja monitorado continuamente durante o traslado. Isso possibilita à equipe especializada da Unidade se preparar para atender prontamente o paciente na sua chegada.

Esta monitoração poderá ser feita dentro de uma ambulância do SAMU, a qual estará em movimento, fazendo com que o TCP sofra com vários dos problemas inerentes a comunicação sem fio além de estar sujeito a constantes *handoffs*.

A arquitetura proposta é mostrada na Figura 3.1.

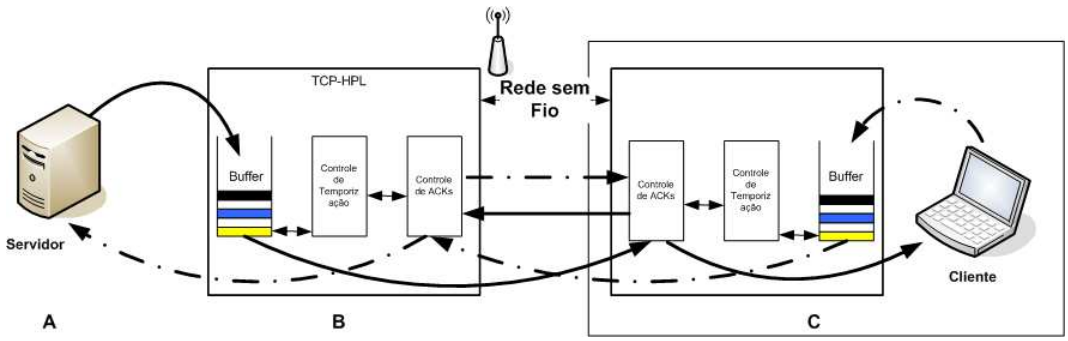

Figura 3.1: Arquitetura do Proxy TCP

O proxy executa as seguintes tarefas:

- *bufferização*
- *Controle de ACKs*
- *Controle de Temporizações*

3.1 Bufferização

A bufferização permite a retransmissão de segmentos TCP perdidos sem a necessidade de consultar o TCP originador.

Todos os segmentos recebidos são copiados para uma fila com um campo contendo o instante em que o segmento deve ser foi armazenado. Este campo serve como base para

o cálculo do instante em que o segmento deve ser retransmitido para o destino em caso de um *timeout*, ou como base para transmitir o ACK recebido do destino para a origem.

A tabela 3.1 apresenta os campos da strutura utilizada para armazenar o segmento TCP.

Tabela 3.1: Estrutura para armazenamento dos segmentos TCP

Tipo	**Item**	**Descrição**
int	dir	Armazena a direção
u32	seq	indica o número de sequência;
u32	ack	indica a sequência de acknowledge
int	sndCounter	indica quantas vezes o pacote foi enviado
int	length	tamanho do pacote em bytes
char	*buffer	ponteiro para o pacote
struct timeval	expTime	tempo em que o pacote expira
struct TAG_IP_PACKET_QUEUE	*next	ponteiro para o próxima estrutura

A idéia principal é manter uma fila ordenada dos segmentos na ordem em que devem ser transmitidos/retransmitidos baseados em um controle de temporização.

Quando o pacote é armazenado na fila o campo `expTime` é calculado baseado no RTT atual em função de alguns coeficientes, os quais serão descritos na seção 3.3 e a fila é ordenada. Assim que o valor de `expTime` é atingido o pacote é transmitido para o seu destino e permanece na fila sendo que o campo `expTime` é recalculado. Quando a mensagem de ACK é recebida o segmento é deletado da fila. Se o `expTime` expirar antes da recepção do ACK o segmento será retransmitido novamente e continuará na fila sendo que o `expTime` será atualizado baseado em um novo fator multiplicativo.

O diagrama apresentado na figura 3.2 demonstra este fluxo.

3.1.1 Mecanismo de Bufferização e Controle de Fluxo do TCP

O TCP padrão possui um mecanismo de janela deslizante para controle do fluxo.

Em uma conexão entre um cliente e um servidor, o cliente informa ao servidor o número de bytes de dados que ele consegue receber de uma só vez do servidor, ou seja,

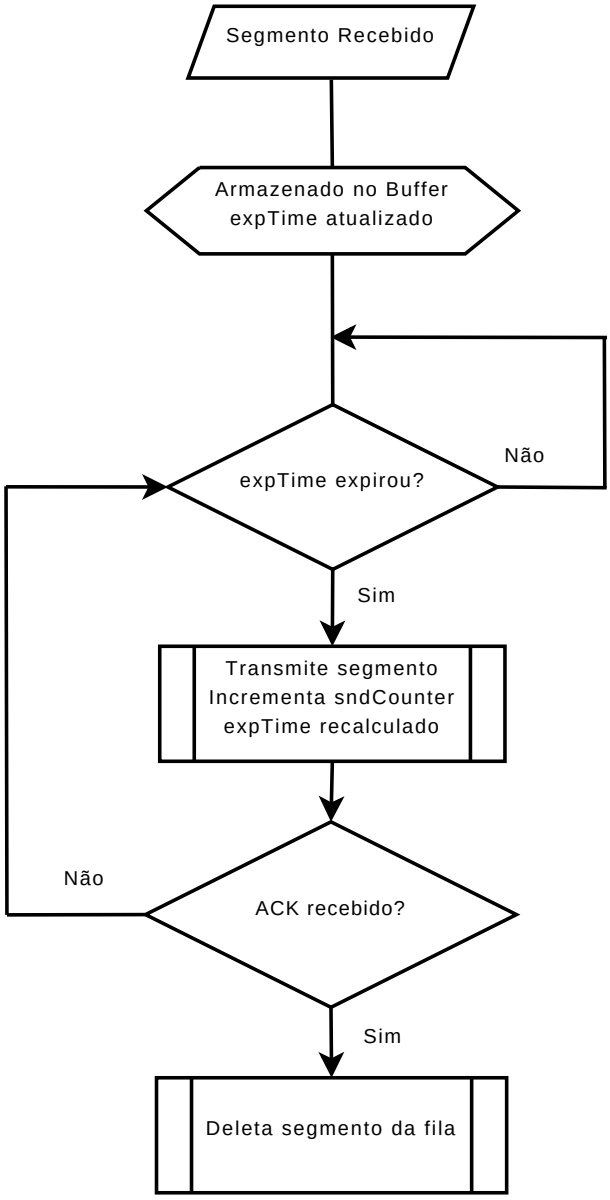

Figura 3.2: Controle de Bufferização

dentro de uma janela. Da mesma forma, o servidor diz ao cliente quantos bytes de dados ele suporta receber do cliente ao mesmo tempo.

Quando o servidor recebe dados do cliente, ele coloca-os em um buffer. O servidor deve então fazer duas coisas distintas, com esses dados: reconhecê-los via um ACK e transferir os dados para o processo de destino. Quando o buffer de recepção não é esvaziado na mesma velocidade com que os dados são recebidos o TCP ajusta o tamanho da janela e sinaliza para o originador. Este é o controle de fluxo do TCP.

O controle de fluxo é de vital importância para o TCP pois este é o método com o qual os dispositivos comunicam o seu estado ao outro. Ao reduzir ou aumentar o tamanho

da janela é assegurado que o outro dispositivo envia dados tão rápido quanto o receptor pode tratá-los.

Este mecanismo não é afetado pela processo de bufferização do TCP-HPL. Na verdade o processo de bufferização gera um atraso intencional para que o TCP original mantenha os valores de RTT mais altos de modo a possibilitar a retransmissão de segmentos perdidos sem que o TCP ative os mecanismos de controle de congestionamento.

3.2 Controle de ACKs

O controle de ACKs permite que o TCP de origem mantenha seu estado mesmo em caso de perdas de pacotes. A idéia é evitar o disparo do mecanismo de controle de congestionamento TCP desnecessariamente. Esta aplicação é baseada no ATCP. Dessa forma os pacotes ACK duplicados serão ignorados. O controle de ACKs é usado como base para o cálculo do RTT.

3.2.1 Congestionamento no TCP e Algoritmo Congestion Avoidance

O controle de fluxo é uma parte muito importante na regulação da transmissão de dados entre dispositivos, mas é limitado no que diz respeito ao seguinte: ele só regula o que está acontecendo em cada um dos dispositivos na conexão, e não o que está acontecendo em dispositivos entre eles, ou seja é uma abordagem fim a fim.

Muitas vezes a transmissão dos segmentos é realizada através de um conjunto de redes e roteadores de redes entre os dispositivos. Estas redes e roteadores também transportam dados de muitas outras conexões. Se o conjunto de redes torna-se muito ocupado, a velocidade na qual os segmentos são transmitidos será reduzida, podendo até mesmo ser descartados. Isso é chamado de congestionamento.

Em nível de TCP não há nenhuma maneira direta de compreender o que está causando o congestionamento ou porquê. Percebe-se simplesmente a dificuldade de transmissão de segmentos através da necessidade retransmissão de alguns deles devido a falta da recepção dos ACKs. Quando este estado é detectado o TCP ativa os mecanismos de controle de congestionamento, evitando a retransmissão desnecessária e diminuindo a taxa de transmissão via controle da janela deslizante.

3.2.2 Sistema de Reconhecimento Acumulativo

O TCP usa um sistema de reconhecimento acumulativo. O campo número de confirmação (ACK) em um segmento recebido por um dispositivo indica que todos os bytes de dados com os números de seqüência inferior a este valor foram recebidos com sucesso pelo outro dispositivo. Um segmento é considerado reconhecido quando todos os seus bytes foram conhecidos, ou seja, quando um número de confirmação que contém um valor maior do que o número de seqüência do seu último byte é recebido.

3.3 Controle de Temporizações

O controle de temporizações tem como objetivo determinar quando um pacote deve ser retransmitido para o destino, ou quando uma resposta deve ser passada para a origem. Este controle é feito por um escalonador que varre a fila de entradas e ajusta o seu calendário de acordo com o próximo evento a ser tratado.

O cálculo do RTT é baseado na recepção de um pacote da origem e na chegada do respectivo ACK do destino. Esse valor é usado como referência para calcular o momento em que o ACK deve ser passado para a origem ou para cálculo do valor do RTO (*Retransmission Timeout*), o qual é utilizada na determinação do instante em que um segmento precisa ser retransmitido.

O valor atual do RTT é multiplicada por um factor C em caso de pacotes recebidos da origem para obter o valor de RTO e por um fator D em caso de pacotes recebidos do destino. O valor resultante será adicionado ao tempo em que o pacote foi recebido para se obter o momento em que ele precisa ser retransmitido ou passado para a origem. Isto faz com que o TCP originador perceba um valor de RTT igual a D * RTT e o TCP de destino perceba um valor de RTT igual a C *D. Ele permite a retransmissão de pacotes perdidos sem que o TCP originador dispare os mecanismos de controle de congestionamento. Dependendo do valor C, mais de uma retransmissão é possível.

As mesmas regras descritas na RFC 2988 (PAXSON; ALLMAN, 2000) são seguidas para o cálculo do RTO.

3.3.1 Gerenciamento de retransmissões através da fila de retransmissão

O método para detecção de segmentos perdidos e retransmissão é conceitualmente simples embora o TCP implemente alguns detalhes que garantem a sua eficácia durante a retransmissão de vários segmentos sem reconhecimento no devido tempo de cada segmento. Cada vez que um segmento é enviado, um timer de retransmissão é iniciado. Se o timer expira antes que o segmento seja reconhecido, o segmento é retransmitido.

A seguinte sequência é seguida:

1. Colocação em fila de retransmissão e início do timer: Logo que um segmento de dados é transmitido uma cópia do segmento é armazenada em uma estrutura de dados denominada fila de retransmissão. Um timer de retransmissão é então iniciado. A fila é mantida ordenada pelo tempo restante no timer de retransmissão, assim o TCP pode controlar os temporizadores que têm menos tempo restante antes que expirem.

2. Processamento do ACK: Se uma mensagem de reconhecimento é recebida antes que o timer do segmento expire ele é removido da fila.

3. Estouro do timer: Se uma mensagem de reconhecimento não é recebida antes que o timer expire o segmento será automaticamente retransmitido.

Como não há garantias de que um segmento retransmitido será recebido, ele é mantido na fila. Cada vez que ele é retransmitido o timer é resetado. Para evitar infinitas retransmissões em determinadas condições há um contador limitando o número máximo de retransmissões.

Conforme mencionado anteriormente o TCP aplica o conceito de reconhecimento acumulativo, ou seja, todos os segmentos com sequência inferior ao ACK serão removidos da fila de retransmissão.

3.3.2 Retransmissão Adaptativa e cálculo do timer de retransmissão

O valor do timer de retransmissão é muito importante para o funcionamento do TCP. Se for muito baixo, poderá ocasionar a retransmissão de um segmento que foi efetivamente recebido, mas não houve tempo suficiente para que o ACK tenha sido recebido. Por outro lado, se o timer for muito longo, muito tempo será perdido caso uma confirmação não seja recebida devido a problemas na rede, reduzindo portanto o desempenho geral do TCP.

Para minimizar estes efeitos o TCP emprega utiliza um método de retransmissão adaptativa baseada em cálculos do tempo de ida e volta das mensagens entre os dispositivos da conexão de modo a ajustá-lo ao longo do tempo de acordo com aumento ou diminuição do tempo médio. Este tempo é o que conhecemos como RTT.

O RTT tenta suavizar os aumentos ou diminuições no tempo médio entre idas e voltas das mensagens na conexão. Isto é obtido através da aplicação de coeficientes que suavizam a influência dos novos valores das medidas de tempo no cálculo do SRTT (*Smoothed Round Trip Time*).

$$SRTT = (\alpha * SRTT) + ((1 - \alpha) * RTT) \tag{3.1}$$

Onde α é um fator de suavização na faixa de valores entre 0 e 1. Valores de α mais próximo de 1 proporcionam uma maior suavização e evitam que mudanças bruscas influenciem diretamente no valor do RTT. Por outro lado isto também faz com que o TCP venha a reagir mais lentamente com relação as mudanças do tempo médio da troca de mensagens. Já valores mais próximos de 0 fazem com que o TCP reflita rapidamente as mudanças na rede. Isto pode, em determinadas condições da rede, provocar instabilidades, fazendo com que o RTT flutue descontroladamente.

Na prática o valor de α fica entre 0,8 e 0,9.

Com o SRTT calcula-se o RTO:

$$RTO = min(MAX\_RTO, max(MIN\_RTO, \beta * SRTT)) \tag{3.2}$$

Onde β é um fator de variância do RTT, geralmente entre 1,3 e 2.

Os valores MAX_RTO e MIN_RTO representam os valores limite para o RTO, geralmente 1 minuto e 1 segundo respectivamente.

Embora a medição do RTT seja conceitualmente simples existem algumas situações que precisam ser identificadas para que o valor do RTT não seja calculado erroneamente. Uma destas situações ocorre quando há a retransmissão de um segmento supostamente perdido. Quando o ACK é recebido não é possível saber se ele se refere ao segmento originalmente enviado ou ao segmento retransmitido.

Esta situação particular não é simples de ser resolvida. Melhorias nesta área foram feitas através da utilização de uma técnica denominada algoritmo de Karn (PAXSON;

ALLMAN, 2000). A principal mudança que este algoritmo traz é não considerar os valores obtidos no tempo de RTT em se tratando de segmentos retransmitidos. Isso elimina completamente o problema da ambiguidade no reconhecimento de segmentos retransmitidos.

Outra medida introduzida por este algoritmo foi a incorporação de um *backoff* para o timer de retransmissão caso o segmento seja retransmitido. Ou seja, quando o segmento é retransmitido, o próximo valor de RTO para este segmento vai ser aumentado por um fator multiplicador, geralmente 2, para dar mais tempo para que o segmento seja reconhecido.

O temporizador continua a ser aumentado até que uma retransmissão seja bem-sucedida, até um valor máximo determinado. Isso impede que retransmissões sejam enviadas muito rapidamente e potencialmente aumentando mais ainda os efeitos de um congestionamento da rede.

Uma vez que uma transmissão bem sucedida ocorre, o valor do RTO retorna para o valor normal.

3.4 Estabelecimento e Término da sessão TCP

Nesta seção serão abordados os aspectos do estabelecimento e término da sessão TCP e as modificações introduzidas pelo TCP-HPL neste processo visando melhorar o TCP original nas condições descritas na seção 1.3.

3.4.1 Máquina de Estados Finita do TCP (TCP FSM)

A máquina de estados do TCP é relativamente complexa. Ela é mostrada na figura 3.3.

A máquina de estados é seguida para cada conexão independentemente.

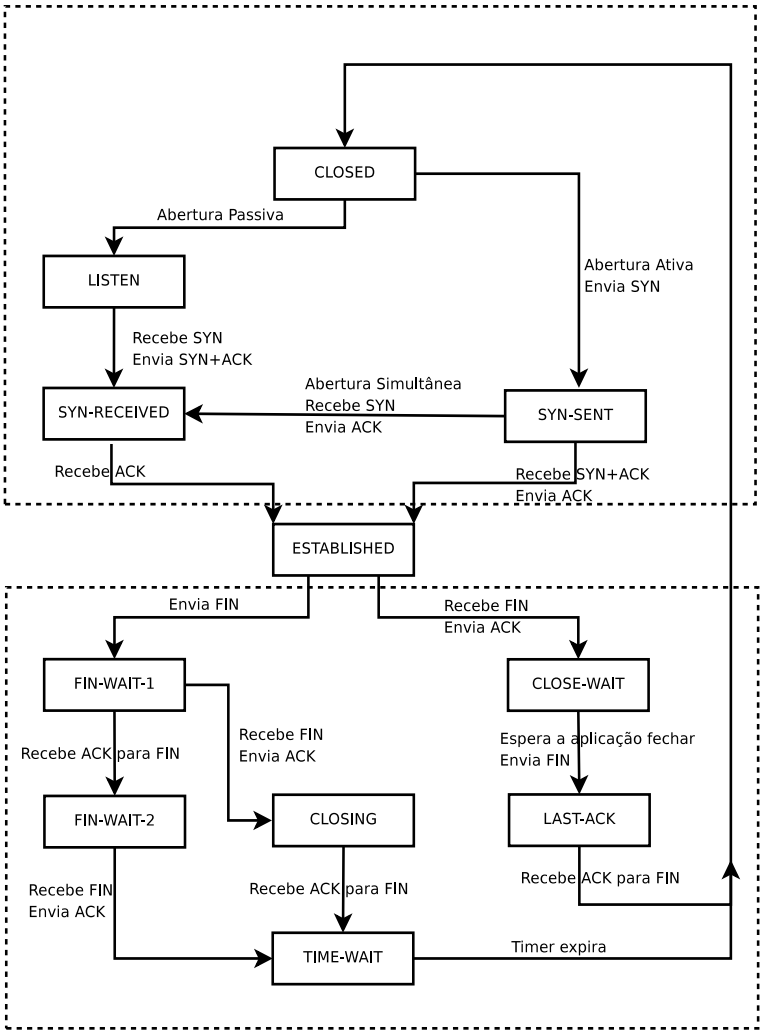

Figura 3.3: Máquina de Estados Finita do TCP

Os estados do estabelecimento da conexão TCP são descritos na tabela 3.2. Estes são

os estados pelos quais o TCP passa no processo de estabelecimento da conexão TCP num processo conhecido como *3-Way Handshake*.

Tabela 3.2: Estados para estabelecimento da conexão TCP

Estado	**Descrição**
CLOSED	É o estado inicial. Esse estado existe antes que uma conexão seja iniciada ou quando ela é finalizada.
LISTEN	É o estado do TCP que indica que um dispositivo está esperando um pedido para iniciar uma conexão.
SYN-SENT	Indica que o dispositivo enviou um SYN para iniciar a conexão e está aguardando a resposta SYN-ACK adequada.
SYN-RCVD	Indica que o dispositivo enviou a resposta SYN-ACK depois de ter recebido o SYN.
ESTABLISHED	Indica que a conexão foi estabelecida. O dispositivo que iniciou a conexão entra nesse estado depois de receber o SYN-ACK e o dispositivo que responde depois que recebe o ACK.

Os estado referentes ao término da conexão são descritos na tabela 3.3.

3.4.2 Three-Way Handshake

Para entender como funciona o Three-Way handshake é necessário conhecer a máquina de estados do TCP, a qual foi foi apresentada na seção 3.4.1.

Antes dos dados poderem ser efetivamente transmitidos é necessário estabelecer a conexão TCP entre os dispositivos que farão parte da conexão fazendo com que a máquina de estados do TCP de ambos saia do estado *CLOSED* e chegue ao estado de *ESTABLISHED*. A partir daí os dados podem ser transmitidos em ambos os sentidos. Este processo é feito através de trocas de mensagens de controle com utilização de flags do campo de opções do TCP.

Estas mensagens de controle usam o mesmo formato padrão de um segmento TCP e servem como indicação para o TCP avançar na máquina de estado tanto no estabelecimento quanto no término da conexão.

Para o estabelecimento da conexão são utilizados dois *flags* de controle.

Tabela 3.3: Estados para estabelecimento da conexão TCP

Estado	**Descrição**
FIN-WAIT-1	É o estado do TCP após o dispositivo ter enviado um pacote FIN inicial pedindo um fechamento correto da conexão TCP.
CLOSE-WAIT	Indica que o dispositivo que recebeu um FIN inicial enviou de volta um ACK para confirmar o FIN.
FIN-WAIT-2	Indica que o dispositivo recebeu a resposta ACK para seu FIN inicial e que agora ele está esperando um FIN final.
LAST-ACK	Indica que o dispositivo acabou de enviar seu segundo FIN, que é necessário para encerramento correto da conexão TCP, e está aguardando uma confirmação.
TIME-WAIT	Nesse estado encontra-se o dispositivo originador que recebeu um FIN final e enviou um ACK para fechar a conexão. Nesse momento ele não irá mais receber nenhuma confirmação do ACK que acabou de enviar, portanto espera um período de tempo para fechar a conexão.
CLOSED	É o estado inicial. Esse estado existe antes que uma conexão seja iniciada ou quando ela é finalizada.

- *SYN*: Este bit setado em um segmento TCP indica solicitação para inicialização de uma conexão.
- *ACK*: Este bit indica o reconhecimento do dispositivo que recebeu o *SYN*.

Para estabelecer uma conexão cada dispositivo precisa enviar um *SYN* e receber um *ACK* do outro dispositivo. Portanto, conceitualmente, seriam necessárias que 4 mensagens de controle fossem trocadas entre os dispositivos. Para aumentar a eficiência na troca de mensagens a mensagem de reconhecimento do *SYN* é enviada com ambos os flags setados *SYN+ACK* diminuindo o total de mensagens para 3. Por esta razão, o processo de estabelecimento da conexão TCP é conhecido como *"Three-Way Handshake"*.

A tabela 3.4 apresenta o fluxo de mensagens relativas no tempo referentes ao processo de *Three-Way Handshake*.

O fluxo de mensagens também é mostrado na figura 3.4.

Devido ao processo inicial do *Three-Way Handshake* impactar diretamente o cálculo do valor do RTT e, sendo que uma das estratégias do TCP-HPL é inflar o valor do RTT

Tabela 3.4: Sequência Three-Way Handshake

Ponteiro	**Descrição**
t	O dipositivo A envia um pacote com o flag de sincronismo (SYN) para o dispositivo B.
t+1	O dispositivo B recebe o pacote (SYN) do dispositivo A.
t+2	O dispositivo B envia um pacote com os flags de sincronismo (SYN) e reconhecimento (ACK) para o dispositivo A.
t+3	O dispositivo A recebe o pacote de B.
t+4	O dispositivo A envia um pacote de reconhecimento positivo (ACK) para o dispositivo B.
t+5	O computador B recebe o ACK e finalmente a conexão TCP é estabelecida. Inicia-se a transmissão de dados até o término da sessão TCP.

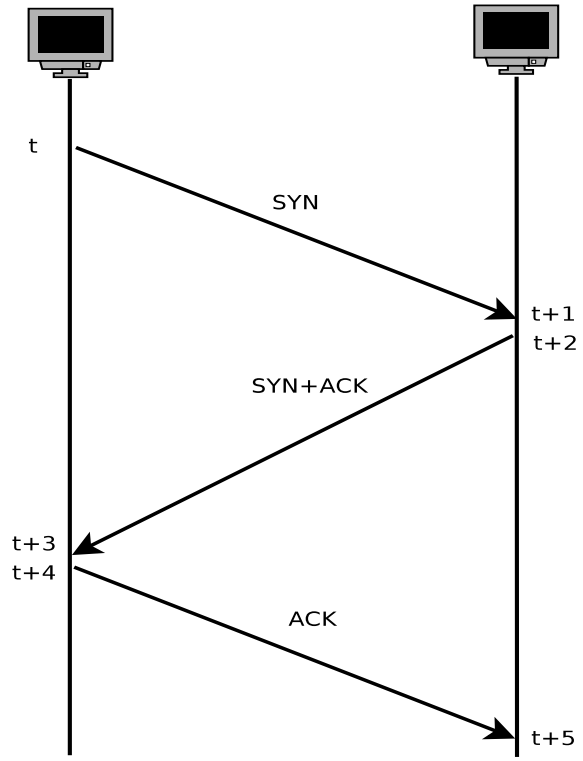

Figura 3.4: TCP Three-Way Handshake

para que perdas de pacotes não impliquem em disparos dos mecanismos de congestionamento do TCP foi necessária a intervenção no processo de modo a aplicar os coeficientes descritos na seção 3.2.2 durante a troca de mensagens com os flags de controle.

Após o TCP-HPL interceptar a mensagem de *SYN* da origem um temporizador configurado com o valor de C*(RTT atual) é iniciado. A mensagem de *SYN* só é enviada para o destino após os estouro do temporizador. Da mesma maneira, quando o TCP-HPL in-

tercepta a mensagem de *SYN+ACK* do destino, é iniciado um temporizador configurado com o valor de D*(RTT atual). A mensagem de *SYN+ACK* só é enviada para o destino após o estouro do temporizador. O mesmo mecanismo é aplicado a mensagem de *ACK* enviada pela origem, finalizando o processo de *Three-Way Handshake*. Após o término deste processo ambos os lados mantém seus valores de RTT inflados de acordo com os coeficientes C e D.

A interação do processo de *Three-Way Handshake* utilizando-se o TCP-HPL é mostrado na figura 3.5.

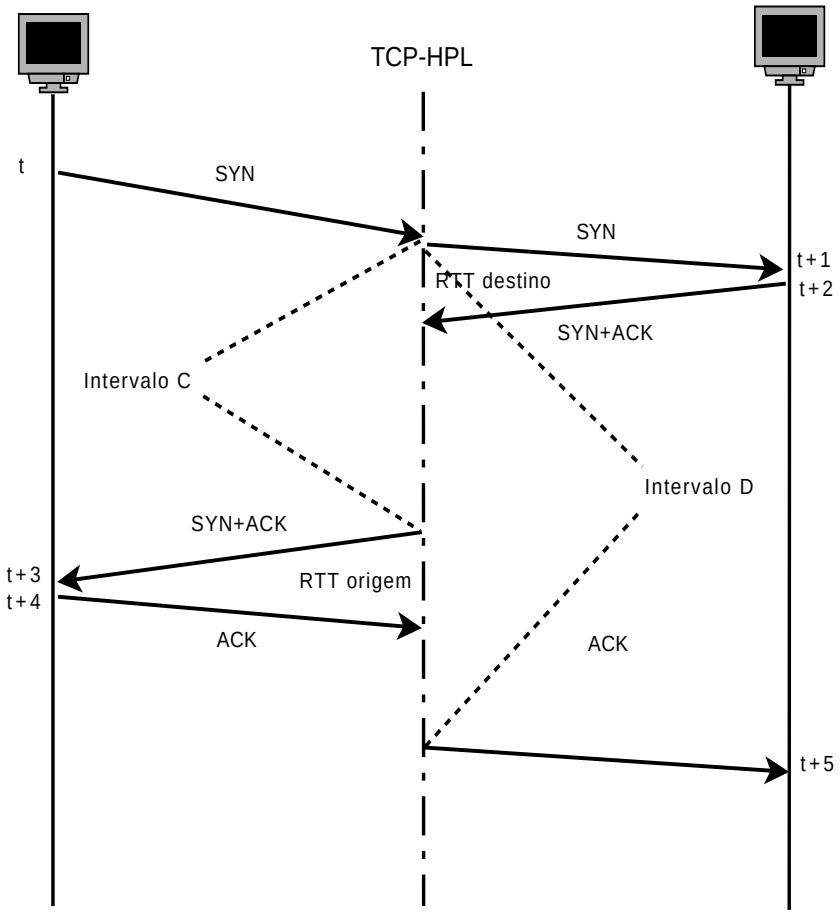

Figura 3.5: TCP-HPL Three-Way Handshake

O TCP também consegue tratar situações onde ambos os dispositivos tentam fazer uma abertura "ativa"ao invés de um deles fazer uma abertura "passiva"primeiro. Isto pode acontecer quando dois clientes tentam estabelecer uma sessão TCP com o outro ao invés do modo tradicional de comunicação onde um exerce o papel de servidor e o outro de cliente. Esta não é uma situação comum, entretanto, pode acontecer em determinadas condições, caso, por exemplo, a mesma porta seja utilizada como origem em ambos os dispositivos.

Neste caso os passos são diferentes para cada dispositivo. Cada cliente executa uma abertura "ativa"e avança nos estados *SYN-SENT* e *SYN-RECEIVED* até que o *SYN* enviado seja reconhecido. Na prática ocorrem dois processos *"Three-Way Handshake"* simultâneos.

3.4.3 Sincronização do número de Sequência

A sincronização do número de sequência e de outros parâmetros que controlam a conexão ocorre durante o processo de *"Three-Way Handshake"*.

O número de sequência é escolhido aleatoriamente de forma a eliminar vários problemas que ocorreriam caso ele fosse 1. Um problema com o número de sequência iniciado com 1 é a possibilidade de ocorrer falsas correlações entre segmentos de diferentes conexões. Quando uma nova conexão é iniciada o valor do número de sequência é gerado randômicamente para minimizar os conflitos que poderiam ocorrer com números de sequência idênticos.

O campo número de sequência é enviado na mensagem inicial de *SYN*. Na recepção do *SYN* o outro dispositivo também informa seu número de sequência e reconhece o número de sequência recebido via indicação no campo *ACK* equivalente ao número de sequência recebido mais um (SEQ+1). Após a recepção da mensagem *SYN+ACK* pelo dispositivo originador, este reconhece o número de sequência recebido do destino via envio da mensagem *ACK* a qual indica o valor de SEQ+1.

A soma do valor 1 na mensagem de reconhecimento indica o próximo número de sequência que o dispositivo espera receber.

Este processo é exemplificado na figura 3.6.

Como o TCP-HPL também gerencia filas de retransmissão ele necessita controlar o número de sequência e de reconhecimento de cada segmento. Portanto ele também precisa saber qual é número de sequência inicial de cada dispositivo. Durante a recepção das mensagens os valores são armazenados nas estruturas de controle de cada segmento.

Adicionalmente ao número de sequência a mensagem *SYN* também carrega informações referentes ao controle da conexão. Estas informações são enviadas através de um campo de opções, o qual contém um tamanho variável.

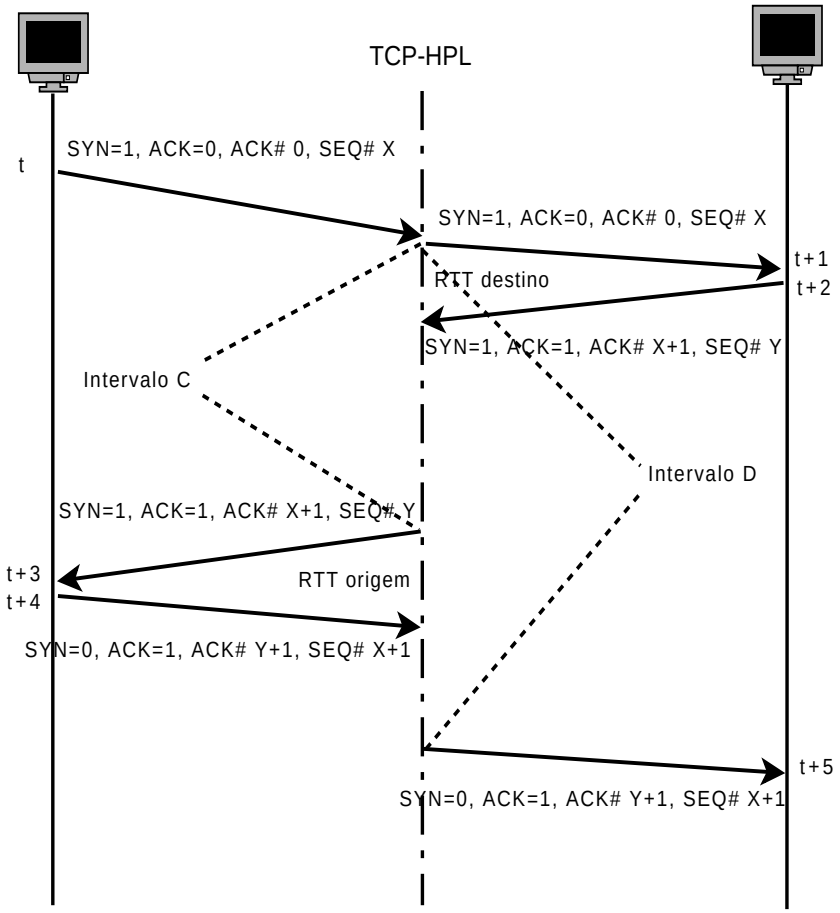

Figura 3.6: TCP-HPL Sincronização do número de Sequência

3.4.4 Troca de configurações entre os dispositivos

No processo de estabelecimento da conexão TCP há troca de configuração entre os equipamentos via parâmetros específicos entre eles e uma vez que toda a configuração é feita, os blocos de controle de transmissão TCB (*Transmission Control Block*), são criados, permitindo o início da transferência de dados.

Cada um dos dois dispositivos em uma conexão deve acompanhar os dados que ele está enviando, assim como os dados que ele está recebendo do outro dispositivo; isto é feito através da divisão conceitual dos bytes tanto recebidos quanto transmitidos.

Enquanto o transmissor deve aguardar a confirmação de cada transmissão, o receptor não necessita.

Tabela 3.5: Categorias na transmissão TCP

Categoria	Descrição
1	Bytes enviados e reconhecidos
2	Bytes enviados mas ainda não reconhecidos
3	Bytes ainda não enviados para o destinatário porém estão prontos para serem recebidos
4	Bytes ainda não enviados para o destinário que ainda não estão prontos para serem recebidos

Tabela 3.6: Categorias na recepção TCP

Categoria	Descrição
1+2	Bytes recebidos e reconhecidos. Este é o complemento da recepção as categorias de transmissão 1 e 2
3	Bytes ainda não recebidos mas prontos para serem. Este é o complemento da recepção à categoria de transmissão 3
4	Bytes ainda não recebidos mas sem preparação para serem. Este é o complemento da recepção à categoria de transmissão 4

3.4.5 Mecânica TCP de janela deslizante de Transferência de Dados e Reconhecimento

Em uma conexão ambos o cliente e o servidor precisam manter informações do fluxo de dados que estão sendo transmitidos e recebidos, isto é feito através da utilização de um conjunto de variáveis especiais denominadas ponteiros, os quais servem para sinalizar as categorias de bytes mostradas anteriormente.

As quatro categorias de transmissão são sinalizadas através de três ponteiros. Dois dos ponteiros são absolutos, ou seja, fazem referência a um número de sequência específico e um indica um *offset* adicionado aos ponteiros absolutos conforme é exemplificado na Tabela 3.7.

As três categorias de recepção utilizam dois ponteiros, os quais são mostrados na Tabela 3.8.

Tabela 3.7: Ponteiros para Controle na Transmissão TCP

Ponteiro	**Descrição**
SND.UNA *Send Unacknowledged*	Indica o número de sequência do primeiro byte de dados que foi recebido mas sem reconhecimento. Ele determina o primeiro byte da categoria 2. Todos os números de sequência anteriores se referem a categoria 1.
SND.NXT *Send Next*	Indica o número de sequência do próximo byte a ser enviado para o outro dispositivo. Ele indica o primeiro byte da categoria 3.
SND.WND *Send Window*	Indica o tamanho da janela. Se adicionarmos o número de sequência do primeiro byte reconhecido ao tamanho da janela teremos a indicação do primeiro byte da categoria 4.

Tabela 3.8: Ponteiros para Controle na Recepção TCP

Ponteiro	**Descrição**
RCV.NXT *Receive Next*	Indica o número de sequência do próximo byte de dados que o receptor espera receber. Ele determina o primeiro byte da categoria 3. Todos os números de sequência anteriores se referem aos bytes já recebidos e reconhecidos, ou seja, categoria 1+2.
RCV.WND *Receive Window*	Indica o tamanho da janela de recepção. Se refere a quantidade de bytes que o receptor suporta receber de uma só vez e esta relacionadao ao tamanho do buffer de recepção. A soma do RCV.WND com o RCV.NXT indica o primeiro byte da categoria 4.

Os ponteiros de transmissão e recepção são complementares pois cada dispositivo gerencia ambos o envio e a recepção de dados do seu parceiro. Eles são mantidos no bloco de controle de transmissão (TCB) de cada dispositivo.

Quando os dados são trocados os ponteiros são atualizados, e a troca de informações sobre o estado do fluxo de dados de recepção e transmissão ocorre através de campos de controle nos segmentos TCP. Os três campos de controle mais importantes são:

- Número de sequência: Identifica o número de seqüência do primeiro byte de dados no segmento sendo transmitido. Este campo é igual ao valor do ponteiro SND.UNA no momento em que os dados são enviados.
- Número de confirmação (ACK): confirma a recepção de dados, especificando o número de sequência que o transmissor do segmento espera transmitir no próximo segmento. Este campo será normalmente igual ao ponteiro RCV.NXT do dispositivo transmissor.

- Tamanho da Janela: O tamanho da janela de recepção do dispositivo que envia do segmento e, portanto, a janela de envio do dispositivo que está recebendo o segmento.

Durante a conexão TCP podem ocorrer perdas de mensagens inerentes ao meio, principalmente em ambiente de rede sem fio, neste aspecto o TCP-HPL consegue reenviar as mensagens perdidas devido ao inflamento das temporizações do TCP nos dispositivos participantes da conexão. A figura 3.7 mostra o fluxo de mensagens nesta situação.

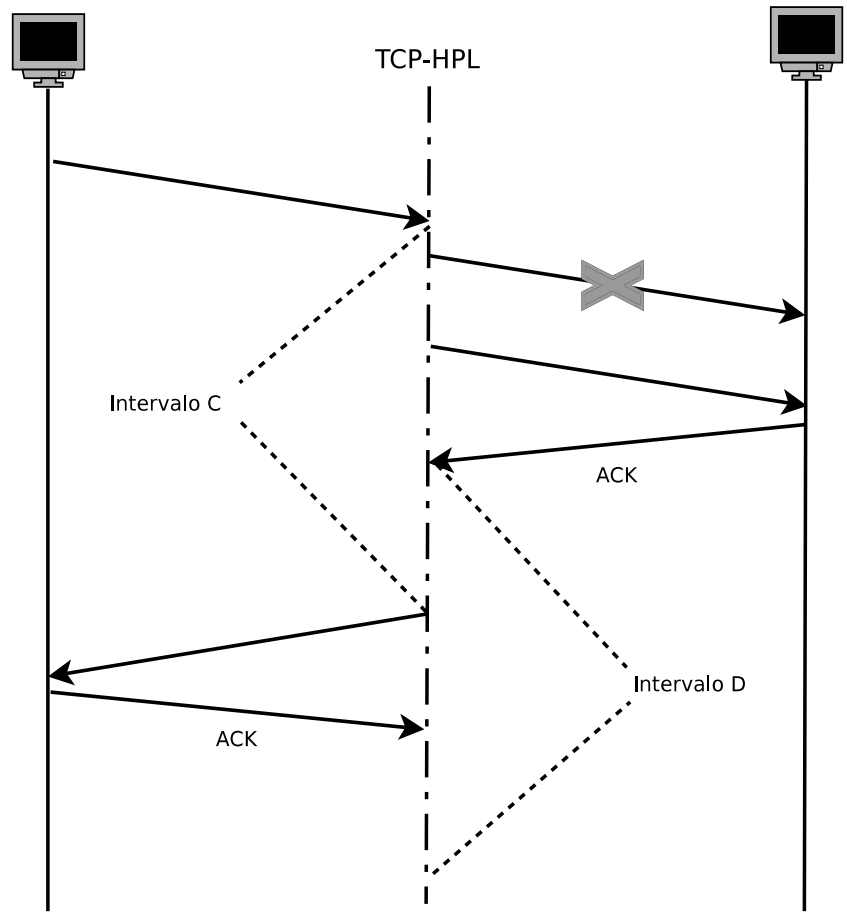

Figura 3.7: TCP-HPL gerenciando perda de mensagens

3.4.6 Término da sessão TCP

Assim como o processo de estabelecimento da conexão exige uma sequência específica de troca de mensagens, também há uma sequência específica de troca de mensagens para o término da conexão TCP.

O processo de término da conexão TCP é um pouco mais complicado, conforme pode

ser observado pelo maior número de estados que envolvem este processo. A razão para esta complexidade é que após o estabelecimento da conexão ambos os dispositivos podem enviar e receber dados simultâneamente. Portanto vários passos precisam ser seguidos para que a conexão seja encerrada de maneira graciosa e sem perda de dados durante este processo.

Geralmente o término da conexão é iniciado independentemente em cada dispositivo, o qual indica ao TCP que ele quer terminar a conexão. A ação de terminar a conexão por um dos dispositivos significa que ele não quer mais enviar dados, mas ainda continua a receber dados até que o outro dispositivo decida parar de enviá-los. Isto permite que todos os dados pendentes sejam enviados para ambos os lados da comunicação antes que a conexão seja encerrada.

Nos casos normais de término de conexão cada lado termina sua conexão através do envio de uma mensagem *FIN*, a qual também permite enviar dados como um segmento normal TCP. O dispositivo que recebe o *FIN* precisa sinalizar o seu reconhecimento através do envio de um *ACK*. A conexão só é totalmente encerrada quando ambos os dispositivos enviaram o *FIN* e receberam seus respectivos *ACK*.

Os estados que cada dispositivo percorre na FSM do TCP é diferente para o dispositivo que fez a requisição e para o dispositivo que a recebeu.

O processo de término da conexão em condições normais é mostrado na figura 3.8.

O dispositivo que recebe o primeiro *FIN* pode ter de esperar um longo tempo no estado *CLOSE-WAIT* até que a aplicação indique que está pronta para encerrar a conexão. Durante este período dados poderão ser enviados para o dispositivo que solicitou o fechamento da conexão, entretanto a recepção de dados de tal dispositivo não é esperada visto que ele solicitou o encerramento da conexão.

Em algum momento o dispositivo que recebeu o *FIN* também enviará uma mensagem *FIN* solicitando o fechamento do seu lado da conexão e esperará a confirmação vinda do outro lado via mensagem *ACK*. Ele ficará no estado *TIME-WAIT* até que a mensagem de *ACK* seja recebida. Este estado é requerido por dois motivos: para certificar que haverá tempo suficiente para que a mensagem de *ACK* seja recebida, mesmo que haja necessidade de retransmissão e, também para prover um período de proteção caso outras conexões estejam sendo iniciadas simultaneamente para este *host*, de forma a evitar confusão com relação ao fluxo de mensagens de diferentes conexões.

O TCP determina que o cliente deve esperar por um tempo denominado MSL (*Ma-*

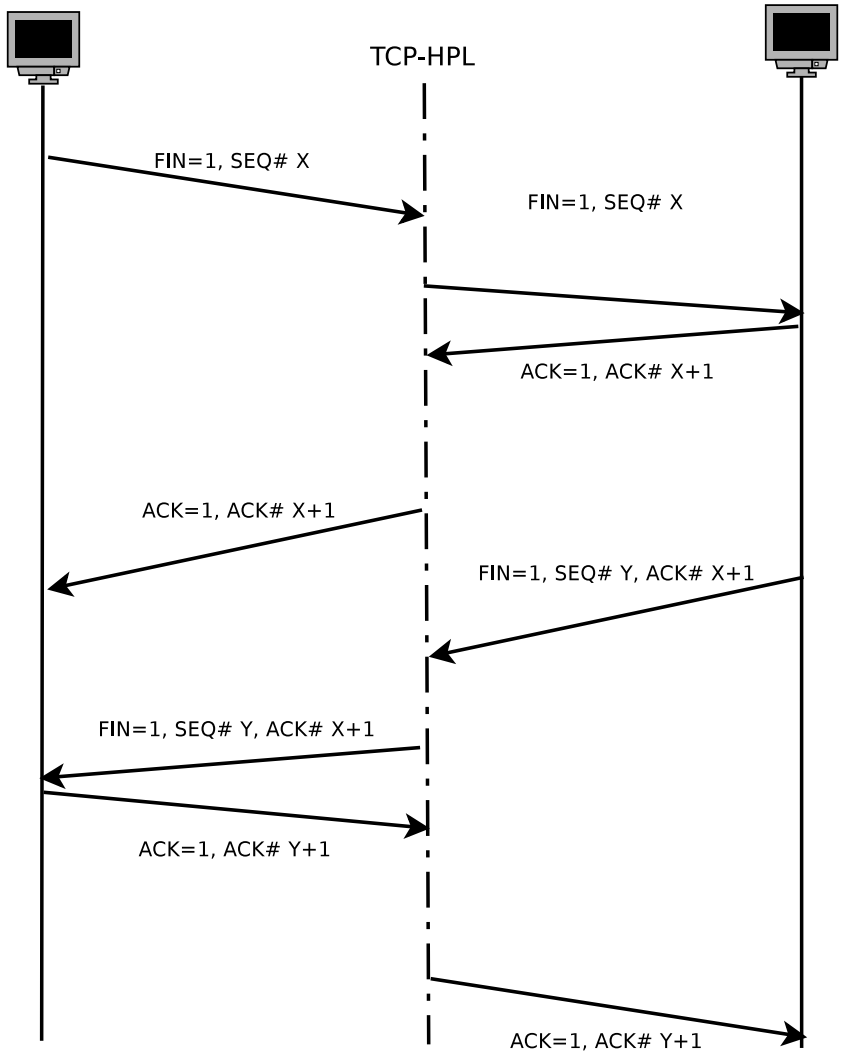

Figura 3.8: TCP-HPL Término da Conexão TCP

ximum Segment Lifetime) antes de finalizar a conexão. O TCP padrão define o valor do *MSL* como 120 segundos, ou seja, 2 minutos. Este valor pode ser alterado dependendo da implementação TCP adotada.

Assim como é possível no estabelecimento da conexão TCP que dois dispositivos façam uma abertura ativa simultaneamente, também pode ocorrer que dois dispositivos iniciem o término da conexão ao mesmo tempo. Neste caso o processo de fechamento é muito mais simétrico com relação a transição de estados. Cada dispositivo responde a mensagem *FIN* recebida com um *ACK*, espera a recepção do seu *ACK* e aguarda um tempo para se certificar que o *ACK* enviado seja recebido pelo outro lado antes de encerrar a conexão.

O TCP-HPL atua no término da conexão da mesma forma que atua no estabelecimento da conexão. As mensagens são armazenadas na fila de mensagens e o tempo em que ela deve ser repassada ao destino é calculado em função dos valores de RTT atu-

ais e dos coeficentes C ou D. Se houver retransmissões o valor do RTO sofre um fator multiplicativo.

3.5 Ambiente de Desenvolvimento

Com relação ao ambiente de implementação, o sistema operacional Linux foi utilizado, visto que possibilita a manipulação e configuração das pilhas de protocolos de uma forma mais abrangente que o ambiente Windows além de se tratar de um software livre.

O TCP-HPL foi desenvolvido utilizando-se a linguagem C. Para a compilação do programa foi utilizado o compilador gcc, o qual é extensamente difundido e utilizado nos ambientes de desenvolvimento em Linux, além de se tratar de um compilador extremamente eficiente e robusto.

As bibliotecas de *threads* e semáforos do *POSIX* foram utilizadas para desenvolvimento das rotinas de tratamento dos segmentos TCP, de foram bidirecional, e para desenvolvimento do escalonador de controle das transmissões e retransmissões.

Embora não tenha sido utilizado o conceito de classes na elaboração do TCP-HPL, devido a questões de desempenho e por isto optou-se pela linguagem C ao invés de C++, atentou-se para a divisão funcional visando uma estrutura mais organizada e eficiente visto que os mecanismos de temporização do TCP são extremamente sensíveis.

3.5.1 Estrutura Funcional do Software

O diagrama do programa em blocos funcionais é mostrado na figura 3.9. Ele visa implementar a arquitetura proposta mostrada na Figura 3.1 através da especialização das atividades em blocos funcionais.

- *Sistema Operacional*: É o sistema nativo linux.
- *Raw Sockets*: Este bloco faz a intermediação das conexões TCP estabelecidas entre os dispositivos. Atua na camada IP dentro da estrutura do sistema operacional para conexões em rede entre dispositivos. É responsável pela abertura e fechamento do *socket*.
- *Máquina de Estados Finita*: Gerencia o estado da conexão TCP através da análise dos flags do campo de opções dos segmentos TCP. Identifica o processo de estabele-

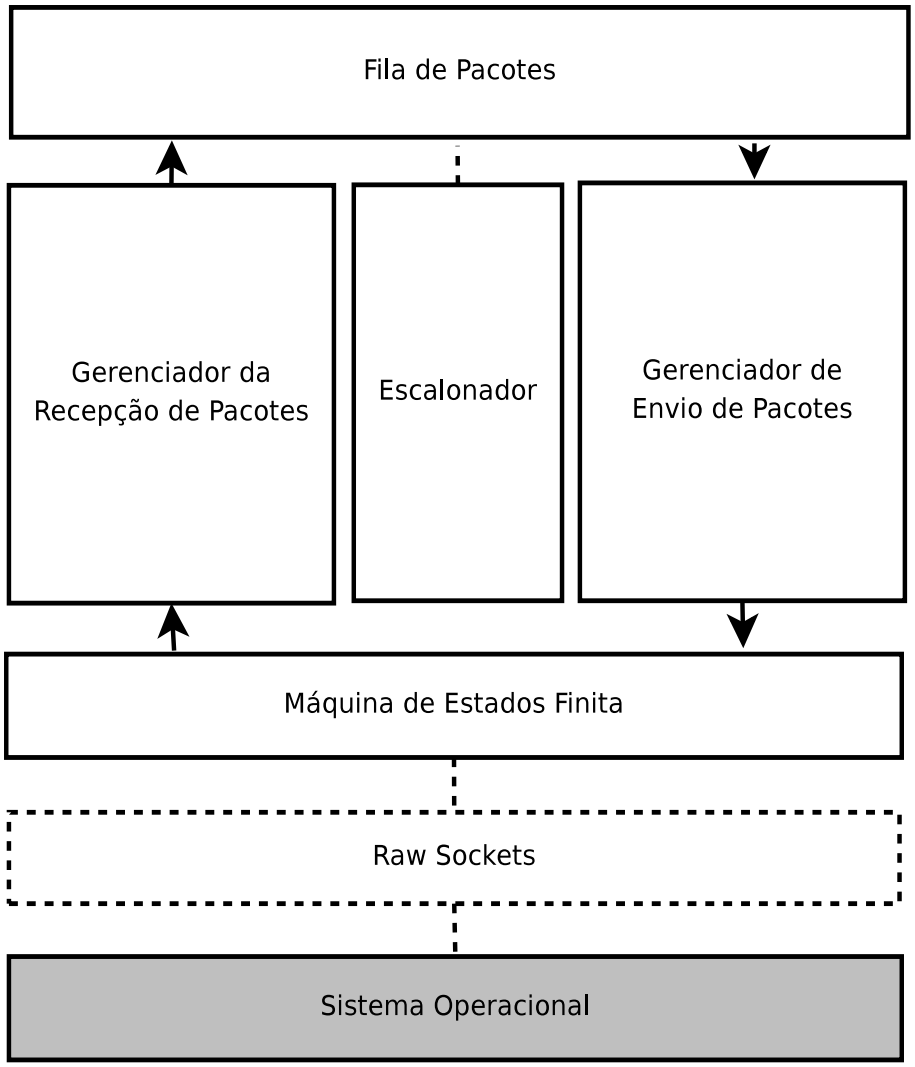

Figura 3.9: Estrutura Funcional do Software

cimento e encerramento da conexão. Interfere no processo de *Three-Way Handshake* e término da sessão TCP para que os valores de RTT sejam "inflados"pelos coeficientes C e D.

- *Gerenciador da Recepção de Pacotes*: Este bloco contém a *thread* de leitura dos segmentos TCP recebidos através do *raw socket*. Também é responsável por identificar os segmentos provenientes da origem e do destino de forma a preencher os campos `dir`, `seq` e `ack` da estrutura para armazenamento do segmento TCP. Este segmento é então inserido na fila de pacotes.

- *Fila de Pacotes*: Este bloco gerência a inserção e deleção das estruturas de armazenamento dos segmentos TCP. Ele prove *APIs* para que o outros blocos acessem as estruturas. Visto que as estruturas representam áreas críticas, ou seja, necessitando

de operações atômicas de inserção e deleção, elas são protegidas por semáforos. A fila de pacotes também é responsável pelo controle de reconhecimento acumulativo, no qual os pacotes que receberam um ACK são removidos da fila. Também é responsável pela detecção de sequências de ACKs consecutivos visando eliminar o acionamento dos mecanismos de congestionamento indevidamente.

- *Escalonador*: O escalonador é um módulo gerenciado por um esquema de *clock* de forma a ordenar a fila de pacotes e que determina o momento em que um segmento deve ser enviado para o gerenciador de envio de pacotes para que ele seja transmitido. Neste módulo também são tratados os casos em que ocorre estouro da temporização devido a falta da recepção do ACK referente ao pacote enviado. Neste caso o valor do campo `expTime` é atualizado em função do valor do RTO aplicando-se o conceito de *backoff*.

- *Gerenciador de Envio de Pacotes*: Este bloco contém uma *thread* para gerenciar a transmissão dos pacotes enviados pela fila de pacotes. Eles são enviados ao destino via o *raw socket*.

3.6 Conclusão

A arquitetura do TCP-HPL não requer alterações nos módulos do Kernel, sendo esta uma importante vantagem com relação as demais soluções apresentadas, tais como o ATCP, variante na qual alguns mecanismos como o processo de *bufferização* e tratamento de ACKs serviram de inspiração.

Ela provê mecanismos que visam inflar os temporizadores do TCP de forma a possibilitar a recuperação de segmentos perdidos devido a perdas de pacotes, principalmente em ambiente de redes sem fio, atuando de maneira semelhante ao RTO fixo mas utilizando coeficientes independentes tanto para o originador quanto o receptor e atualizando os valores de RTT baseados na equação do SRTT, ao contrário do RTO fixo.

Os principais eventos e estados do TCP são controlados internamente sendo o controle de temporizações a parte mais crítica do TCP-HPL, pois este influencia diretamente o desempenho geral na transmissão TCP.

Capítulo 4

Resultados

Neste capítulo serão apresentados os métodos para obtenção dos resultados, os mecanismos e ferramentas utilizadas e os resultados obtidos.

4.1 Cenário de Testes

Um cenário de teste consistindo de 3 computadores em série foi usado para simular um *download* de dados num dispositivo móvel, cujo servidor está localizado na porção de rede cabeada.

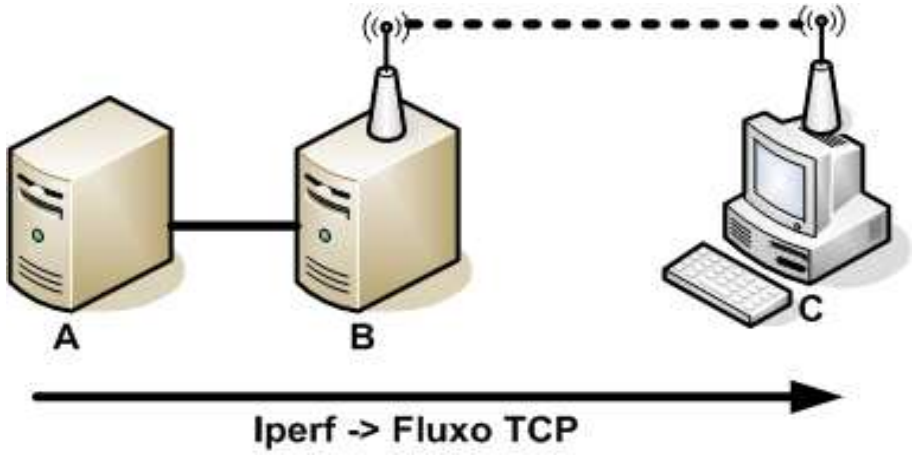

Figura 4.1: Cenário de Testes

A figura 4.1 exemplifica o cenário de teste.

Todos os computadores foram configurados com linux distribuição Ubuntu, versão 8.04LTS, kernel 2.6.24 e placa de rede de 100Mbps. O TCP original se refere a imple-

mentação do protocolo TCP presente no Ubuntu e definida na RFC 793 (INSTITUTE, 1981), RFC 1122 (BRADEN, 1989) e RFC 2001 (STEVENS, 1997) com as extensões NewReno e SACK.

Consideremos os respectivos computadores como `a`, `b` e `c`. O segmento `a-b` simula a porção cabeada da rede e o segmento `b-c` a porção da rede sem fio.

4.2 Geração de Erros Randômicos

Para a geração de erros randômicos foi utilizada uma distribuição uniforme através do uso da função geradora de números pseudo-aleatório `rand()` com a inserção de uma semente para evitar que a mesma sequência randômica seja gerada sempre. A semente foi inserida via comando `srandom(time(NULL))`.

Os erros são inseridos diretamente durante a recepção de um pacote independente do sentido e sempre no segmento `b-c`, o qual representa a porção da rede sem fio.

4.3 Medição do Throughput

A aplicação `iperf` foi usada para gerar tráfego e para medir o *throughput* de dados.

O `iperf` é um programa bastante difundido e utilizado para medir largura de banda entre computadores. Ele pode ser facilmente instalado no linux via pacote de instalação.

Para executar a geração de tráfego basta iniciar o `iperf` na máquina que vai gerar o tráfego com a opção de servidor e iniciar o `iperf` como cliente na máquina que vai receber os pacotes. Nos dois computadores será informado a velocidade alcançada durante a conexão.

A largura de banda foi limitada em 30Mbps baseada nos padrões 802.11a ou 802.11g através do uso de controle de tráfego `tc` no linux via disciplina de filas `qdisc classful htb` para garantir uma abordagem mais real do *throughput* máximo em uma rede sem fio.

Na tabela 4.1 são mostrados os comandos para configuração da disciplina de filas `qdisc`.

Rotas estáticas foram criadas nos computadores `a` e `c` para direcionar o tráfego para

Tabela 4.1: Configuração da qdisc para limitação de largura de banda

Comandos qdisc

`tc qdisc add dev eth1 root handle 1:  htb default 0`
`tc class add dev eth1 parent 1:0 classid 1:1 htb rate 100Mbit`
`tc class add dev eth1 parent 1:1 classid 1:11 htb rate 30Mbit`
`tc filter add dev eth1 parent 1:0 prio 1 protocol ip u32 match ip dst [endereço IP] flowid 1:11`

o TCP-HPL, instalado no computador `b`.

4.4 Cálculo dos Coeficientes

Vários ajustes dos coeficientes foram feitos empiricamente de maneira a otimizar a taxa de transmissão de dados do *proxy* TCP. Os seguintes coeficientes para o cálculo do RTT e RTO foram utilizados:

$$C = 2, D = 5 \tag{4.1}$$

A taxa de amortização do SRTT foi calculada baseada em um α de 0,9.

4.5 Obtenção dos Resultados

Primeiramente o TCP-HPL foi configurado apenas para encaminhar pacotes, mas com a possibilidade de introduzir erros aleatórios, de modo a simular os erros na parte de rede sem fio. Os valores da taxa de perda de pacotes entre 0 e 5% foram utilizados para obter a curva de degração do *throughput* no TCP padrão.

Intervalos de 0.01 foram utilizados na variação da taxa de perdas entre 0 e 5%.

Um outro programa foi desenvolvido para iniciar o `iperf` no computador `a` e variar o intervalo na taxa de perdas de modo síncrono no computador `b`, desta maneira todo o processo de obtenção do valor do *throughput* em função da taxa de perdas ocorre automaticamente.

A cada nova medição da taxa de erros foi gerada uma nova semente.

Após a obtenção dos resultados com o TCP padrão o TCP-HPL foi instalado no computador intermediário e o mesmo procedimento foi realizado para se obter a curva de

degração do *throughput* do TCP-HPL em função da taxa de perda de pacote de 0 a 5%.

A Figura 4.2 mostra os resultados obtidos com ambos o TCP padrão e o TCP-HPL.

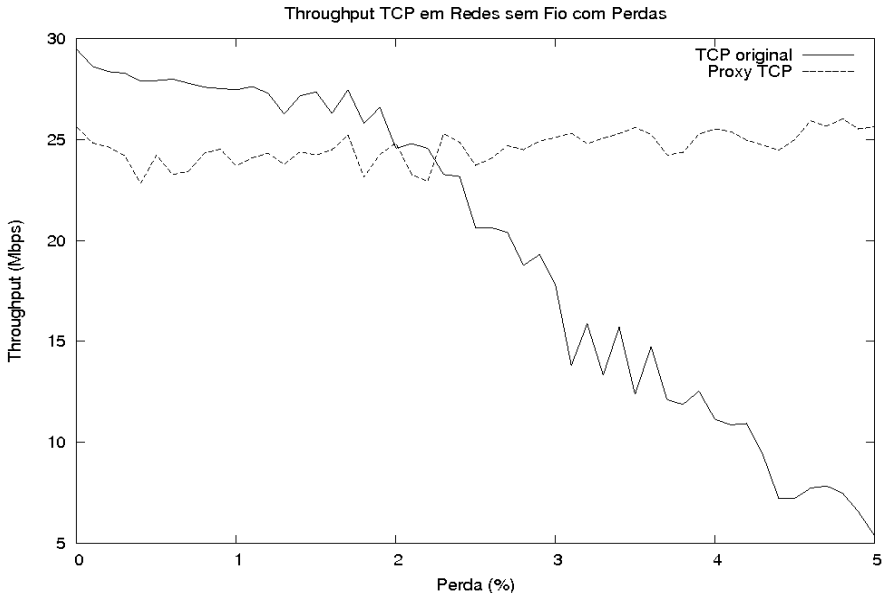

Figura 4.2: Comparação entre o TCP padrão e o TCP-HPL

Nota-se que a taxa de transferência via TCP-HPL mantém-se praticamente inalterada em relação ao aumento da taxa de perdas de pacotes no segmento da rede sem fio, enquanto o TCP padrão sofre uma forte degradação.

4.6 TCP-HPL x Socks

SOCKS é um protocolo de Internet que permite às aplicações cliente servidor usar de maneira transparente os serviços de um *firewall* de rede. SOCKS é uma abreviação de "SOCKetS".

Os clientes que estão protegidos por um *firewall*, os quais precisam acessar servidores externos, podem fazê-lo através da utilização de um servidor proxy SOCKS. Tal servidor proxy possui controle dos clientes que podem fazer tal acesso e repassam a requisição para o servidor externo. SOCKS pode ser usado também da forma contrária, permitindo aos clientes que estão fora do *firewall* se conectar aos servidores de dentro do *firewall*.

O protocolo foi desenvolvido originalmente por David Koblas, um administrador da MIPS Computer Systems. Depois que o MIPS passou a ser controlado pela Silicon Graphics em 1992, Koblas apresentou um artigo sobre SOCKS no Simpósio anual de

segurança Usenix e então o SOCKS chegou a estar disponível publicamente. O protocolo foi estendido à versão 4 por Ying-Dá Lê da NEC.

A extensão não oficial SOCKS 4a acrescenta suporte para resolução de nomes via DNS.

A versão atual do protocolo, o SOCKS v5, é padronizada via RFC 1928 (LEECH M. GANIS; JONES, 1996), e estende a versão prévia suportando UDP, autenticação e IPv6, permitindo ao servidor SOCKS resolver nomes de host para o cliente.

A arquitetura de referência do SOCKS pertence a Permeo Technologies.

De acordo com o modelo OSI o SOCKS é uma camada intermediária entre a camada de aplicação e a camada de transporte.

Inicialmente a proposta do TCP-HPL consistia em atuar em uma versão aberta do SOCKS para implementar os mecanismos sugeridos mas, como ele atua entre a camada de transporte e a camada de aplicação, não é possível atuar diretamente sobre o TCP. Se o SOCKS pudesse ser utilizado para estes fins algumas características que ele possui, tais como autenticação e suporte a DNS seriam agregadas ao TCP-HPL, além dos métodos de controle de acesso.

Mas apesar da mudança de estratégia o TCP-HPL também atua igualmente como um *proxy* para os dispositivos que realizam uma conexão através dele. Ele atua de forma transparente para ambos os dispositivos. Não foram implementados mecanismos de autenticação visto que o principal objetivo é o de melhorar o desempenho do TCP em ambiente de redes sem fio.

Como o TCP-HPL atua como uma camada intermediária entre a camada IP e a de transporte seria possível a união das duas propostas visando criar um mecanismo mais eficiente de *proxy*, agregando as características de ambas as soluções.

4.7 Conclusão

Um delay intrínseco é causado durante a transmissão dos pacotes mas como efeito positivo ele diminui o jitter da conexão.

Também notamos um desempenho pior do TCP-HPL em se tratando de taxa de perdas de pacotes pequena, a qual se deve principalmente a utilização de valores fixos para os coefientes (coeficientes C e D) de ajuste do atraso após armazenamento dos pacotes para

posterior envio ao destino.

Notoriamente o TCP-HPL se destaca na relação do *throughput* com taxa de perda de pacotes mais elevada, atingindo portanto o objetivo de melhora da conexão TCP em ambientes de rede sem fio através das soluções apresentadas neste capítulo.

Capítulo 5

Conclusão

O mecanismo proposto para melhorar o desempenho do TCP em redes sem fio com alta taxa de perda de pacotes garante que a taxa de transferência de dados é mantida mesmo com um significativo aumento da taxa de perda de pacotes, diferentemente do TCP padrão. O TCP-HPL tem um desempenho inferior ao TCP original apenas em condições com taxa de perdas muito pequena devido a utilização de coeficientes fixos.

A principal vantagem do TCP-HPL é ser transparente tanto para o emissor quanto para o receptor, o que significa que a sintaxe do TCP não precisa ser alterada nestes elementos. Ele não requer qualquer mudança no cliente ou servidor TCP ao contrário de outras propostas. Sua instalação é simples como uma aplicação normal.

Também não requer alterações nos módulos do kernel.

Nesta primeira versão valores fixos foram utilizados nos coeficientes para o cálculo do RTT e RTO. Melhorias podem ser feitas para que esses valores possam ser calculados de forma adaptativa de acordo com a taxa de perda de pacotes, tornando o comportamento do TCP-HPL semelhante ao TCP padrão em caso de baixa taxa de perda de pacotes, e com a vantagem de uma baixa degradação na taxa de transferência de dados em enlaces com altas taxas de perda de pacotes.

Como trabalhos futuros poderia haver a integração do TCP-HPL com o protocolo SOCKS, de forma a fornecer um serviço de *proxy* com melhor desempenho principalmente em redes sem fio.

Referências Bibliográficas

AKYILDIZ, G. M. I. F.; PALAZZO, S. Tcp-peach: A new congestion control scheme for satellite ip networks. *IEEE/ACM Trans. Net.*, v. 9, n. 3, p. 307–321, 2001.

AL., T. G. et. Freeze-tcp: A true end-to-end enhancement mechanism for mobile environments. *Proc. IEEE INFOCOM 2000*, p. 1537–1545, 2000.

ALTMAN, E.; JIMéNEZ, T. Novel delayed ack techniques for improving tcp performance in multihop wireless networks. In: *Personal Wireless Communications*. [S.l.]: IEEE, 2003. p. 237–253.

BAKRE, A.; BADRINATH, B. R. I-tcp: indirect tcp for mobile hosts. In: *ICDCS '95: Proceedings of the 15th International Conference on Distributed Computing Systems*. [S.l.: s.n.], 1995. p. 136.

BHARGHAVAN A. DEMERS, S. S. V.; ZHANG, L. Macaw: a media access protocol for wireless lan. *in Proc. of ACM SIGCOMM*, p. 212–225, 1994.

BLUETOOTH Special Interest Group. [S.l.]. Disponível em: <http://www.bluetooth.com>.

BRADEN, R. *Requirements for Internet Hosts - Communication Layers*. [S.l.], 1989. Disponível em: <http://www.faqs.org/rfcs/rfc1122.html>.

BRAKMO, L.; PETERSON, L. Tcp vegas: End to end congestion avoidance on a global internet. *IEEE JSAC*, v. 13, n. 8, p. 1465–1480, 1995.

BRAKMO, L.; PETERSON, L. Tcp veno: Tcp enhancement for transmission over wireless access networks. *IEEE JSAC*, v. 21, n. 2, p. 216–228, 2004.

BROADBAND radio access networks (BRAN): High performance Local Area Network (HiperLAN) type 2. *ETSI TR 101 683*, V1.1.1, 2000–02.

CHANDRAN, K. et al. A feedback based scheme for improving tcp performance in ad-hoc wireless networks. In: *ICDCS '98: Proceedings of the The 18th International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 1998. p. 472.

CHEN, X. et al. A survey on improving tcp performance over wireless networks. In: *In Resource Management in Wireless Networking, Cardei M, Cardei I, Du D-Z (eds.* [S.l.]: Kluwer Academic Publishers, 2005. p. 657–695.

CHIANG, M. Balancing transport and physical layers in wireless ad hoc networks: jointly optimal tcp congestion control and power control. *IEEE JSAC*, v. 23, n. 1, p. 104–116, 2005.

CHIU, D.-M.; JAIN, R. *Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks*. [S.l.], 1989. Disponível em: <http://www.cs.berkeley.edu/ istoica/classes/cs268/papers/cj89.pdf>.

CORDEIRO, S. D. C.; AGRAWAL, D. Copas: Dynamic contention-balancing to enhance the performance of tcp over multi-hop wireless networks. In: *IC3N*. [S.l.]: IC3N, 2003. p. 382–387.

DURST, R. C.; MILLER, G. J.; TRAVIS, E. J. Tcp extensions for space communications. *Wirel. Netw.*, v. 3, n. 5, p. 389–403, 1997.

DYER, T. D.; BOPPANA, R. V. A comparison of tcp performance over three routing protocols for mobile ad hoc networks. In: *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2001. p. 56–66.

FLOYD, S.; HENDERSON, T. *The New Reno Modification to TCP's Fast Recovery Algorithm*. [S.l.], 1999. Disponível em: <http://www.ietf.org/rfc/rfc2582.txt>.

FLOYD, V. J. S. Random early detection gateways for congestion avoidance. *IEEE/ACM Transaction on Networking*, v. 1, n. 4, p. 397–413, 1993.

FLUMINENSE, U. F. *MARFIM - Medicina Assistida por Redes sem FIo Multimidia*. [S.l.], 2008. Disponível em: <http://www.midiacom.uff.br/projeto-marfim>.

FU P. ZERFOS, H. L. S. L. L. Z. Z.; GERLA, M. The impact of multihop wireless channel on tcp throughput and loss. In: *IEEE INFOCOM*. [S.l.]: IEEE, 2003.

FU, Z. et al. The impact of multihop wireless channel on tcp performance. *IEEE Transactions on Mobile Computing*, v. 4, n. 2, p. 209–221, 2005.

GOFF, T. et al. Preemptive routing in ad hoc networks. *J. Parallel Distrib. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 63, n. 2, p. 123–140, 2003.

HANBALI, A. A.; ALTMAN, E. A survey of tcp over ad hoc networks. *IEEE Communications Surveys & Tutorials*, v. 7, p. 22–36, 2005.

HOLLAND, G.; VAIDYA, N. *Analysis of TCP Performance over Mobile Ad Hoc Networks*. [S.l.], 1999. Disponível em: <http://citeseer.ist.psu.edu>.

IEEE 802.11 WLAN standard. [S.l.]. Disponível em: <http://standards.ieee.org/getieee802>.

INSTITUTE, I. S. *Transmission Control Protocol*. [S.l.], 1981. Disponível em: <http://www.faqs.org/rfcs/rfc793.html>.

JONES, C. E. et al. A survey of energy efficient network protocols for wireless networks. *Wirel. Netw.*, v. 7, n. 4, p. 343–358, 2001.

K.FALL, S. F. Simulation-based comparisons of tahoe, reno, and sack tcp. *Comp. Commn Rev.*, p. 5–11, 1996.

KIM, C. T. D.; CHOI, Y. Tcp-bus: Improving tcp performance in wireless ad hoc networks. *Journal of Communications and Networks*, v. 3, n. 2, p. 175–186, 2001.

KLEMM, F.; KRISHNAMURTHY, S. V.; TRIPATHI, S. K. Alleviating effects of mobility on tcp performance in ad hoc networks using signal strength based link management. In: *in Proc. of the Personal Wireless Communications*. [S.l.: s.n.], 2003. p. 611–624.

KOPPARTY S. KRISHNAMURTHY, M. F. S.; TRIPATHI, S. Split tcp for mobile ad hoc networks. In: *in Proc. of IEEE GLOBECOM*. [S.l.]: IEEE, 2002.

LEECH M. GANIS, Y. L. R. K. D. K. M.; JONES, L. *SOCKS Protocol Version 5*. [S.l.], 1996. Disponível em: <http://www.ietf.org/rfc/rfc1928.txt>.

LEFEVRE, G. V. F. Understanding tcpś behavior over wireless links. *Proc. Communications and Vehicular Technology*, v. 2000 SCVT- 200, p. 123–130, 2000.

LI, S.; ANSARI, N. *TCP-Jersey over High Speed Downlink Packet Access*. [S.l.], 2005. Disponível em: <http://web.njit.edu/ ansari/papers/05GC_Shupeng.pdf>.

LIM, K. X. H.; GERLA, M. Tcp performance over multipath routing in mobile ad hoc networks. In: *IEEE ICC*. [S.l.]: IEEE, 2003.

LIU, J.; SINGH, S. Atcp: Tcp for mobile ad hoc networks. *IEEE JSAC*, v. 19, n. 7, p. 1300–1315, 2001.

LUNDGREN, H.; NORDSTRö, E.; TSCHUDIN, C. Coping with communication gray zones in ieee 802.11b based ad hoc networks. In: *WOWMOM '02: Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*. New York, NY, USA: ACM, 2002. p. 49–55.

MASCOLO, S. et al. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In: *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2001. p. 287–297.

MATHIS J. MAHDAVI, S. F. M.; ROMANOW, A. *TCP Selective Acknowledgement Options*. [S.l.], 1996. Disponível em: <http://www.ietf.org/rfc/rfc2018.txt>.

PAXSON, V.; ALLMAN, M. *Computing TCPś retransmission timer*. [S.l.], 2000. Disponível em: <http://www.ietf.org/rfc/rfc2988.txt>.

RAMAKRISHNAN, S. F. K.; BLACK, D. *The Addition of Explicit Congestion Notification (ECN) to IP*. [S.l.], 2001. Disponível em: <http://www.ietf.org/rfc/rfc3168.txt>.

STEVENS, W. *TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms*. [S.l.], 1997. Disponível em: <http://www.ietf.org/rfc/rfc2001.txt>.

TIAN, K. X. Y.; ANSARI, N. *TCP in Wireless Environments: Problems and Solutions*. [S.l.], 2005. Disponível em: <http://web.njit.edu/ ansari/papers/05ComMag_Tian.pdf">.

TOBAGI, F.; KLEINROCK, L. Packet switching in radio channels: Part ii - the hidden terminal problem in carrier sense multiple-access modes and the busy-tone solution. *IEEE Transactions on Networking*, v. 23, n. 12, p. 1417–1433, 1975.

TOH, C.-K. Associativity-based routing for ad hoc mobile networks. *Wirel. Pers. Commun.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 4, n. 2, p. 103–139, 1997.

V.JACOBSON. Congestion avoidance and control. *Proc. ACM SIGCOMM*, p. 314–329, 1988.

WANG, F.; ZHANG, Y. Improving tcp performance over mobile ad-hoc networks with out-of-order detection and response. In: *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2002. p. 217–225.

XU, K. et al. Enhancing tcp fairness in ad hoc wireless networks using neighborhood red. In: *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2003. p. 16–28.

XU, M. K.; BAE, S. Effectiveness of rts/cts handshake in ieee 802.11 based ad hoc networks. *Ad Hoc Networks Journal*, v. 1, n. 1, p. 107–123, 2003.

XU, Y. T. K.; ANSARI, N. Tcp-jersey for wireless ip communications. *IEEE JSAC*, v. 22, n. 4, p. 747–756, 2004.

XU YE TIAN, N. A. K. Tcp-jersey for wireless ip communications. *IEEE Journal on Selected Areas in Communications*, v. 22, n. 4, p. 747–756, 2004.

YANG, L.; SEAH, W. K.; YIN, Q. Improving fairness among tcp flows crossing wireless ad hoc and wired networks. In: *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*. New York, NY, USA: ACM, 2003. p. 57–63.