Trabalho Final

Telecomunicações e redes 1

Vitor de Araújo Vieira 11/0067151

Departamento de ciências das Computação - CIC

Universidade de Brasília. UNB

Brasília, Brasil

Vitor.ecomp@gmail.com

Mateus Mendelson da Silva 11/0017579

Departamento de ciências das Computação - CIC

Universidade de Brasília. UNB

Brasília, Brasil

m.mendelson.unb@gmail.com

# Resumo

O objetivo desse relatório é expor a implementação dos algoritmos da camada de enlace discutidos em sala de aula, essa implementação foi feita baseada nos conceitos definidos de cada algoritmo, tentando simular da forma mais adequada a simular o funcionamento real destes algoritmos. Através dessas simulações será possível validar o que foi aprendido em aula, validando desde algoritmos, até regras que determinam eficiência do algoritmo. Nesse relatórios estão os dados reais catalogados através da simulações, e também as conclusões que foram extraídas desses dados. Os pontos de maior discussão são as eficiências encontradas já que demonstram que cada algoritmo dependendo do caso de uso e das variáveis do sistema.

# Introdução

Em redes de comunicação é de grande importância a entrega dos pacotes de formas correta e ordenada, pois isso e a base para que exista um comunicação entre dois pontos da rede, para que essa seja mantida de forma concisa e correta existem protocolos que visam garantir essa integridade, essa são protocolos de controle de erro. Esses normalmente tem um estrutura básica simples, que normalmente possui as seguintes técnicas:

* Envio da confirmação de recebimento do pacote.
* Retransmissão dos dados em caso de erro.
* Uso de temporizadores, para o controle dos algoritmos.
* Pacotes normalmente enviados em ordem.

Porem existem diversos protocolos que implementam essa regras iremos falar sobre os mais comuns, talvez os mais utilizados desses protocolos, que são o Stop and Wait(ARQ), Go-Back N ARQ e Selective Repeat ARQ, esse são bastante simples e foram implementados para esse trabalho de forma a permitir a validação desses, para a implementação destes deve-se ter em mentes o funcionamento básico destes que e dados abaixo:

### Stop and wait

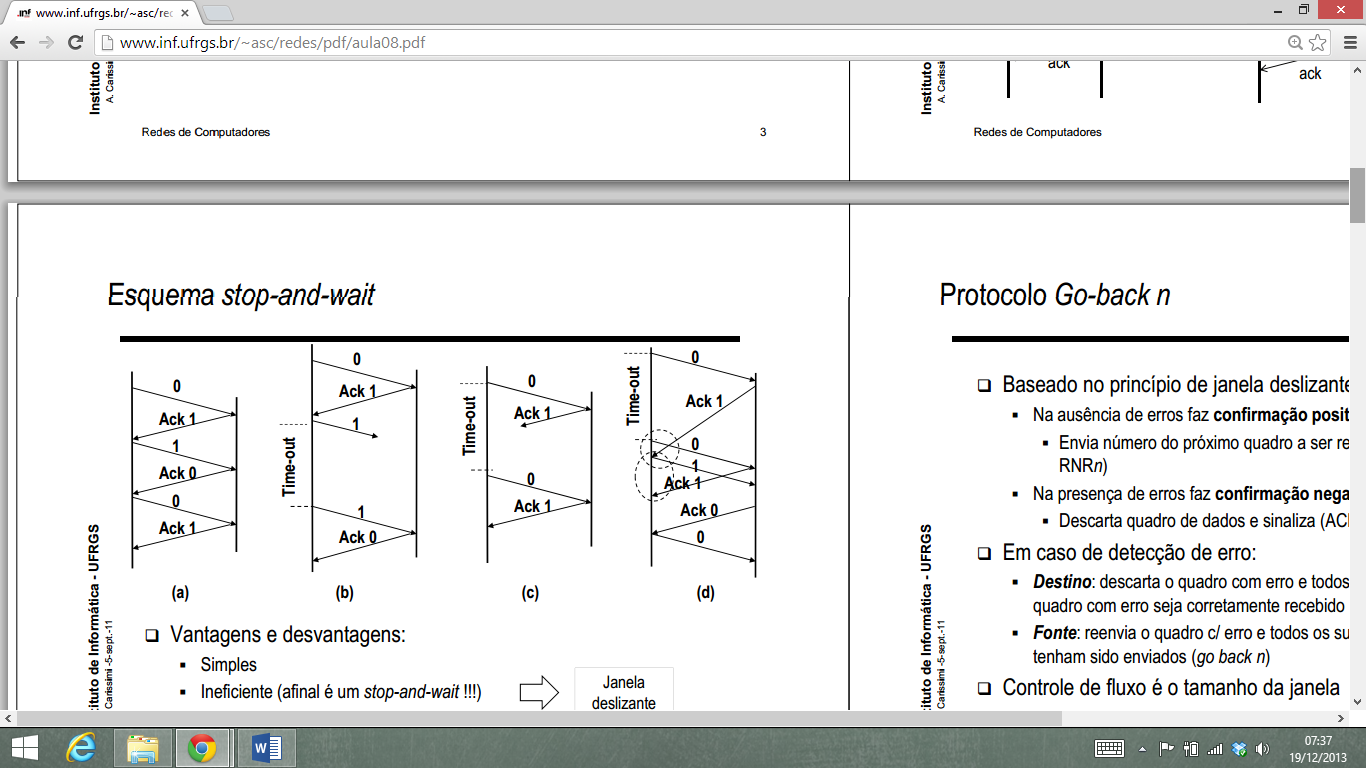
O stop and wait consiste basicamente em um simples algoritmo que envia e aguarda resposta de confirmação de chegada. Aguardar um tempo máximo até essa chegada, se essa não chegar, reenviar o pacote, se ela enviar o próximo pacote. Esse fica melhor explicado com as seguintes figuras:

Figura 1:

Essa representa o funcionamento comum do sistema o transmissor envia a mensagem o receptor recebe essa sem erros envia o ack que chega o transmissor sem erros, quando essa chega o transmissor envia o próximo quadro.

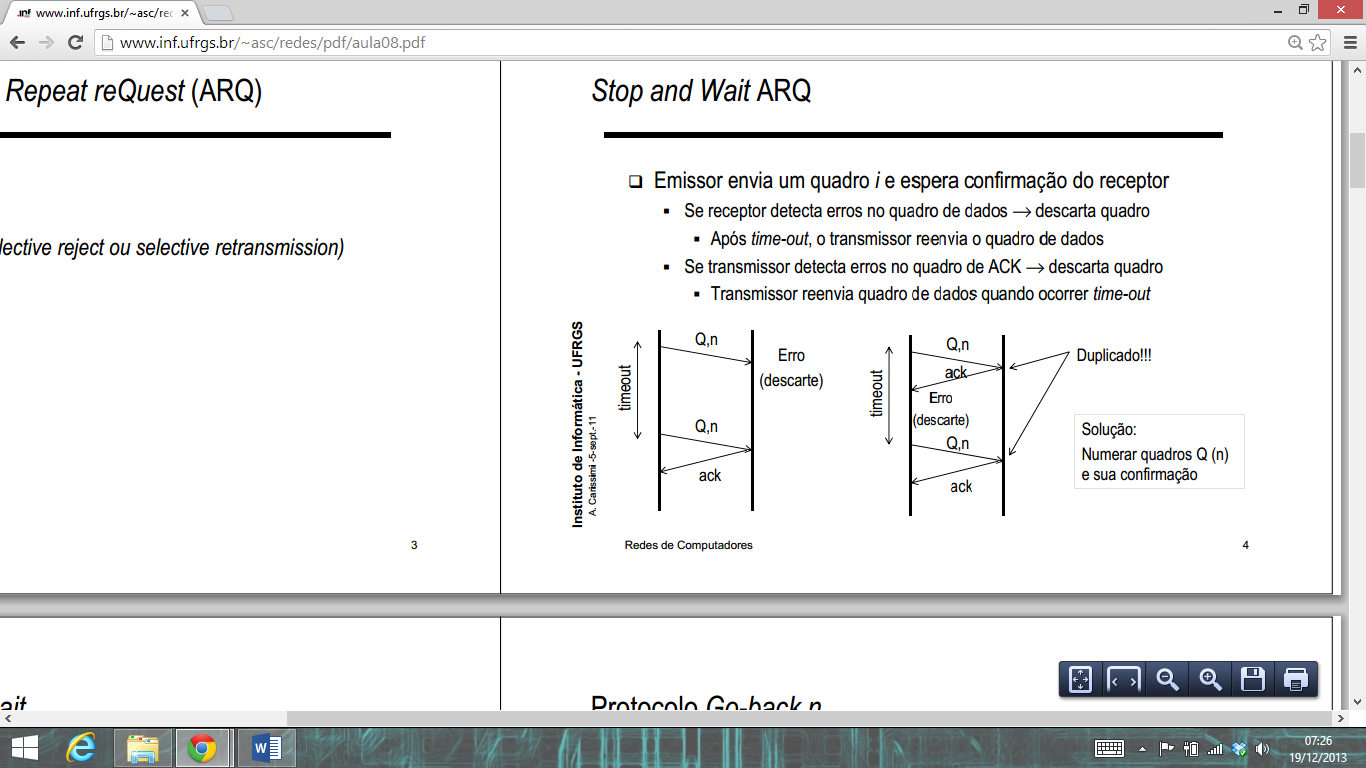


Figura 2:

Nessa figura vemo um caso comum que e o caso de no recebimemto do pacote, oque faz com que o receptor não envie o pacote de ack, e com isso não recebendo o ack após o time-out o transmissor ira retransmitir o quadro.

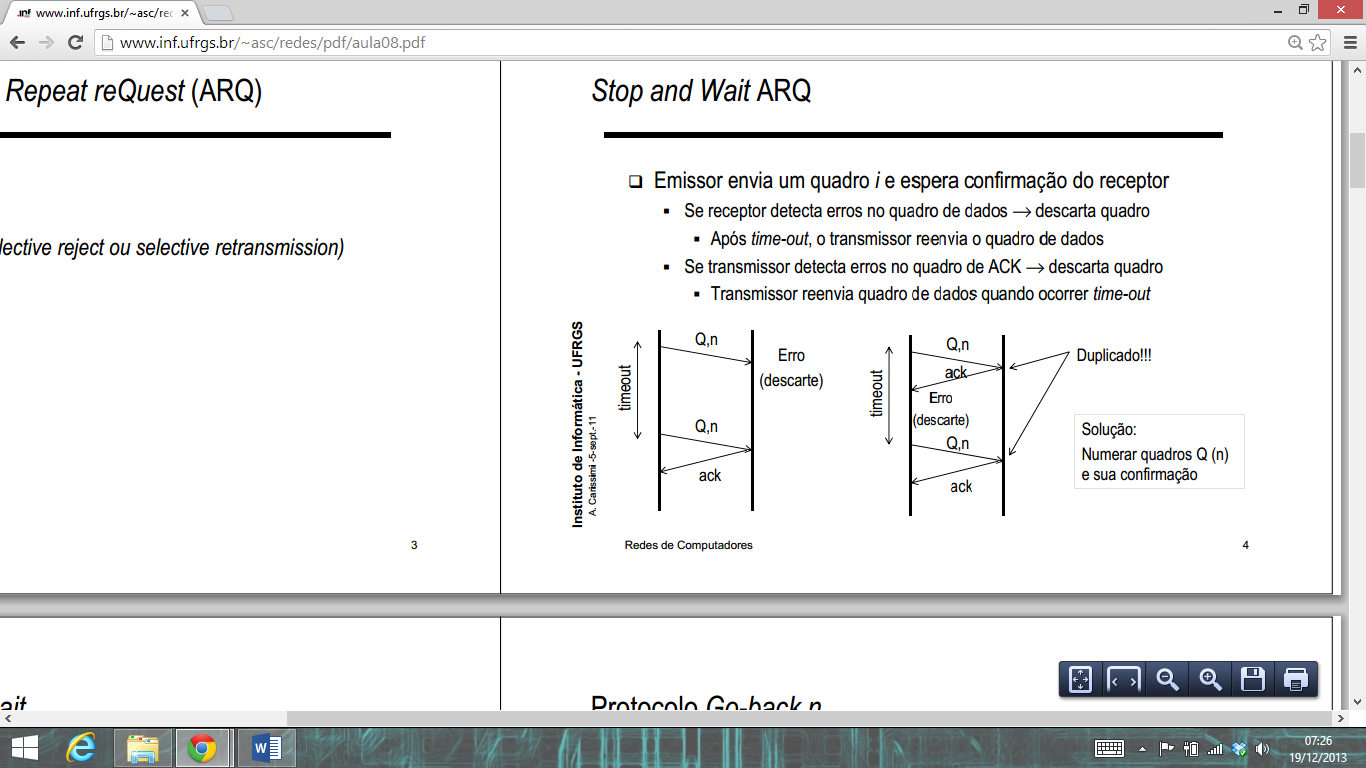


Figura 3:

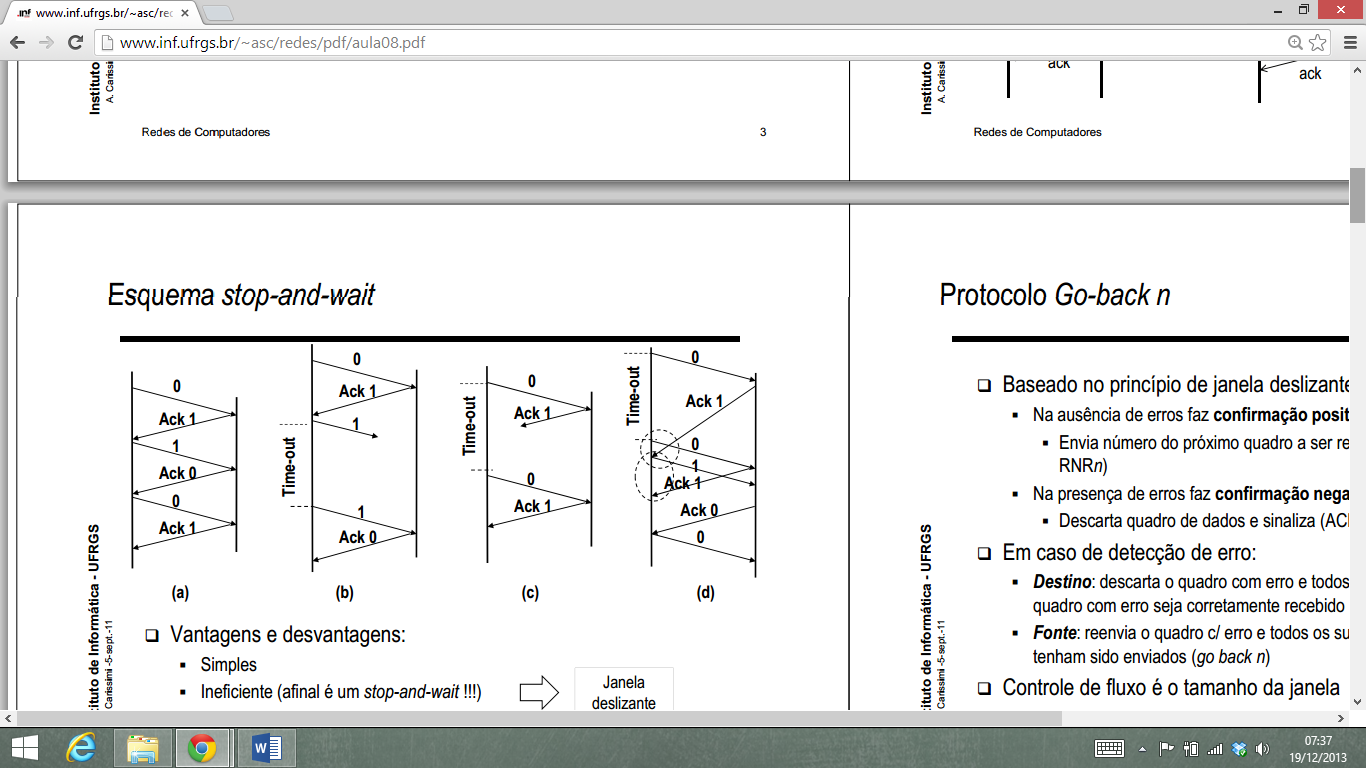
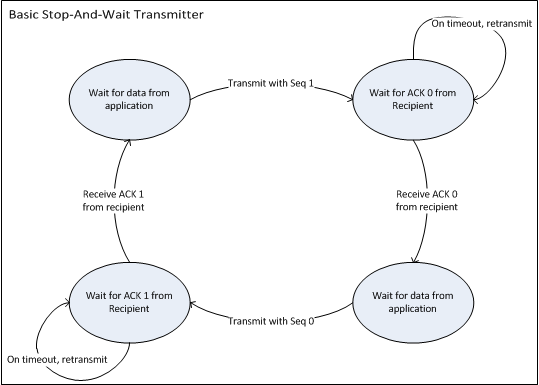
Nessa figura vemos o caso de quando ocorre o erro no ack, quando isso ocorre o receptor já recebeu o quadro porem porro perda de pacotes ou erros no canal o pacote ack não chega integro ao transmissor o que faz com que ele reenvie o pacote que já foi envido, por esse motivo os pacotes devem ser enumerados pois senão não forem o receptor interpretara o pacote n duplicado como o pacote n+1.

Figura 4:

Nessa figura vemos um erros que pode acontecer por uma falha no próprio algoritmo, aqui está representado quando o timeout ficou menor que o tempo de chegada de um ack, o que faz com que eu envie um pacote já recebido e que o ack está vindo em bom estado.

 Figura 5:

Aqui está representada a mef (máquina de estados finitos) que representa o stop and wait e que foi implementada no trabalho com algumas modificações.

Podemos agora discutir as vantagens e desvantagens do stop and wait. A principal vantagem do stop and wait, e que ele e bem simples com uma mef bem pequena, isso faz com que esse possa ser implementado em hardware de forma barata e rápida. Já as desvantagens, são que ele não faz uso da banda máxima do canal por que esse fica ocioso por muito tempo, e ocorrem muitos casos onde o pacote chegou e mesmo assim ele reenvia o pacote.

### Go-Back N

O Go Back n e um algoritmo que complementa o stop and wait tentando fazer um uso mais eficiente do canal, para isso esse implementa um conceito básico que são pacotes de ack para confirmações de recebimento, e ack para recebimentos ruins e também uma enumeração dos pacotes. Em caso de erros o receptor ficar descartando os pacotes até o recebimento do correto, e o receptor reenvia o quadro com erro e todos o subsequentes a esse novamente.

Porem existe um problema que e dado pelo quantidade de números necessários para enumerar os quadros, para resolver isso foi definido o conceito de janela deslizante que define que durante um período de tempo não existiram mais de n pacotes no enlace logo será necessário só n números para enumera-los e depois se recomeça a contagem. Podemos ver nas imagens abaixo uma maior detalhamento desse algoritmo.

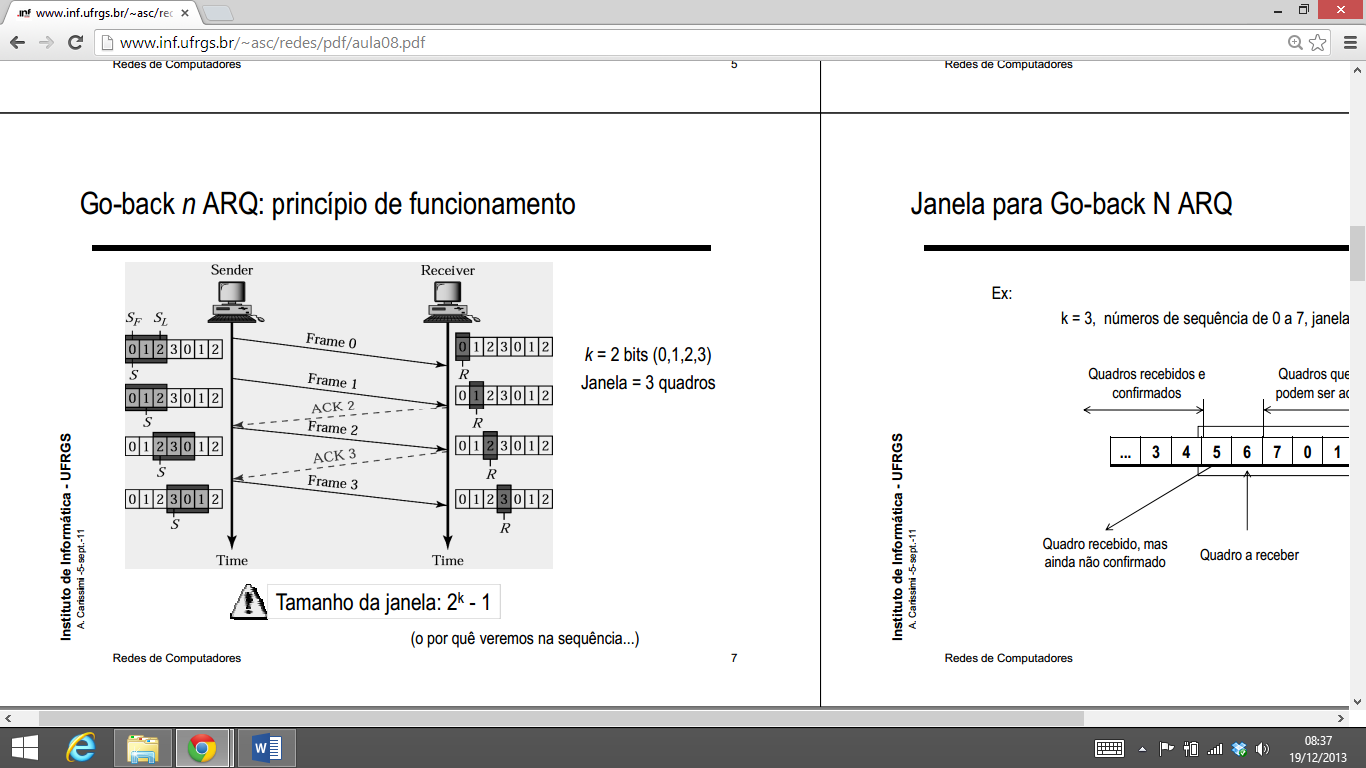
Aqui demostramos o funcionamento normal do go back N onde ele recebe todos os quadros corretos e vai enviando os ack corretos dos dizendo que não houve perda de quadros.

Figura – Go Back N funcionando

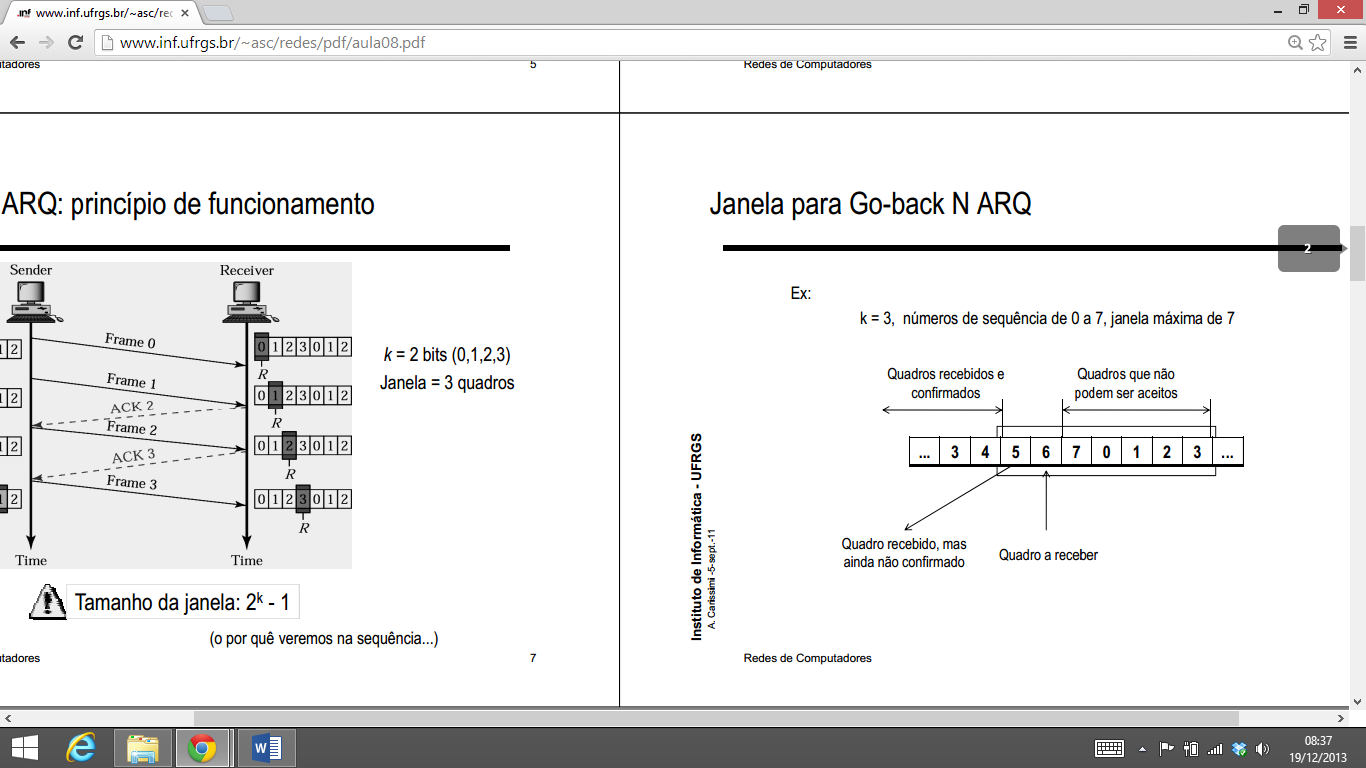
O interessante do go-back n e que esse consegue continuar recebendo quadros sem ainda ter confirmados os anteriores isso pode ser visto na imagem ao lado pois o receptor já está pronto para receber o 6, mesmo ainda sem a confirmação do 5 não ter chegado ao transmissor. Outras das vantagens desse protocolo e a confirmação negativa que garante uma menor dependência aos time outs, que podem ser um problema quando são muito grandes pois deixam o canal muito tempo ocioso, e também não podem ser muito pequenos pois podem gerar falsas situações de falha. Na Figura 3 e possível demostrar isso no exemplo ‘a’ e ‘b’ o canal fica muito tempo ocioso enquanto no ‘c’ esse não fica ocioso por nenhum tempo.

Figura - Janela Go Back N

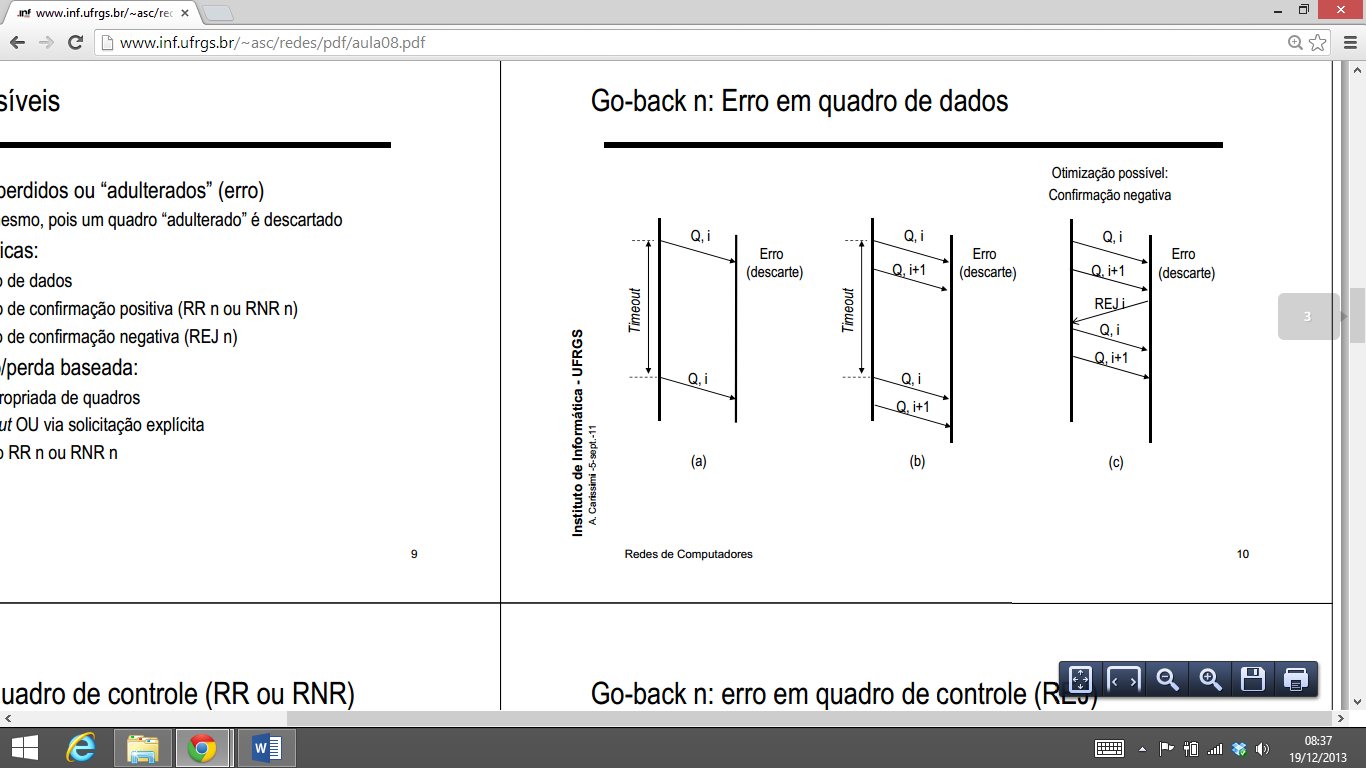
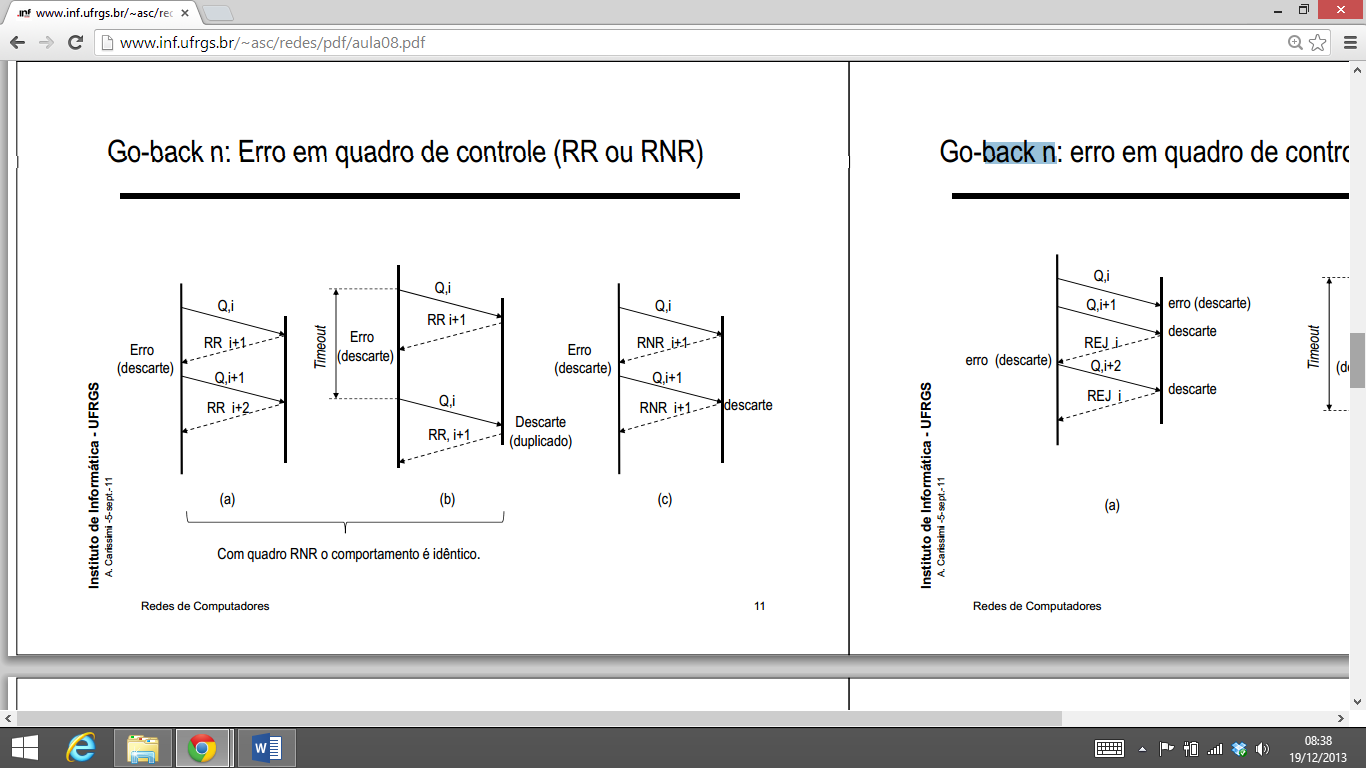
O go back n, não lida bem com erros nos pacotes de confirmação ou rejeição. Isso e demonstrado nas figura 4 e 5 na figura quatro ocorre um erro em um pacote rejeição com essa não chegou o transmissor continua enviando isso faz com que os pacotes continuem chegando errados nos receptores, e esses continuem mandando mensagens de rejeição. Já na figura 5 vimos o erro no ultimo quadro da janela assim o transmissor não tem nada a fazer a não ser esperar até o recebimento de um ack para a continuação da transmissão ou um time out para reenviar o quadro.

Figura 4 - Erro em cabecalho de rejeiçao.

Figura 3 - Mensagem de erro

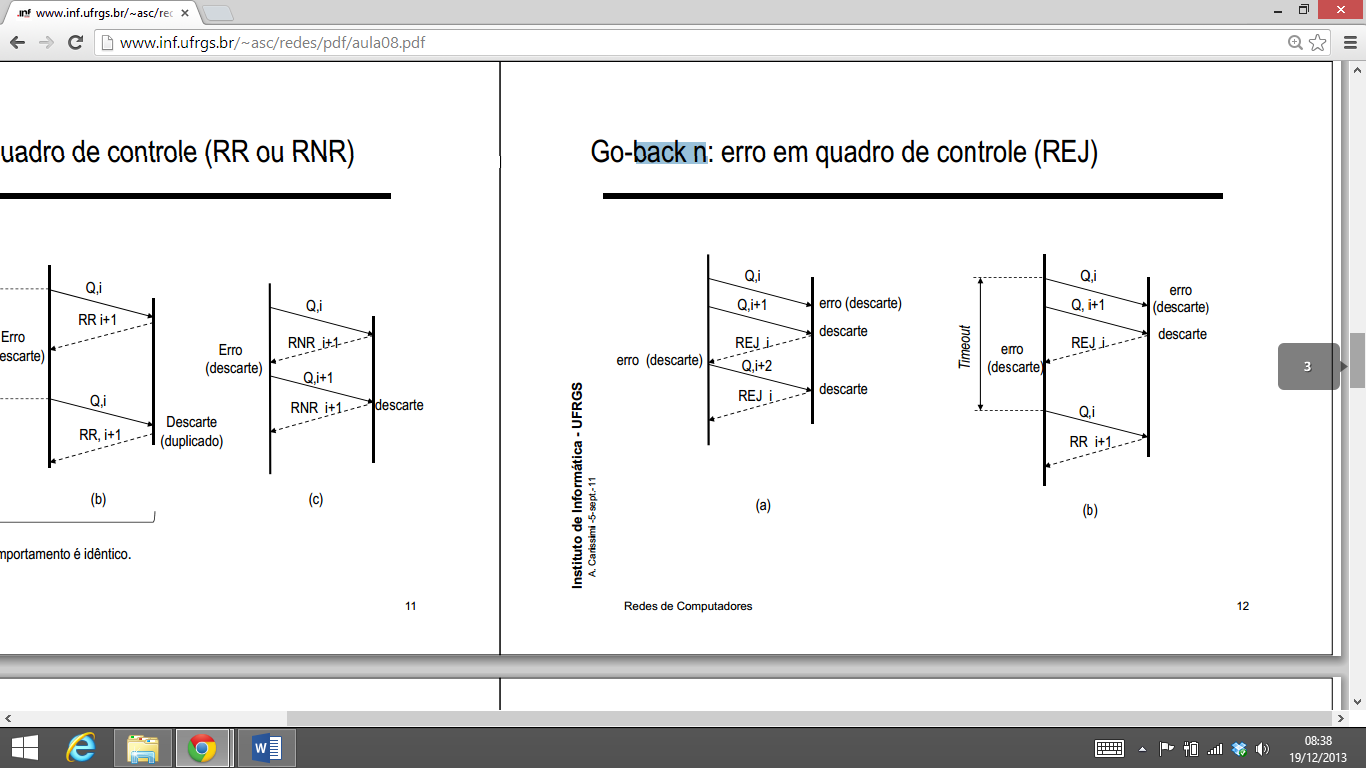
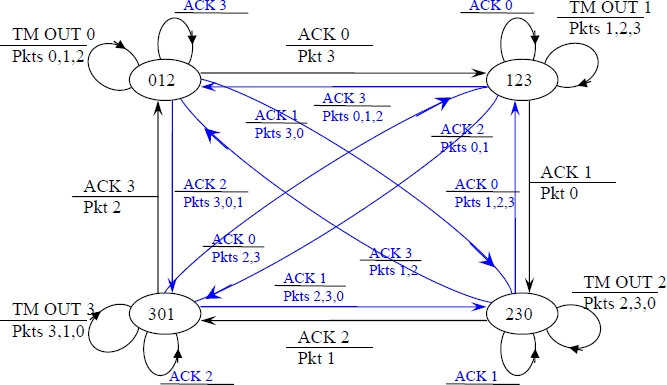


Figura 5 – Erro ultimo quadro da janela

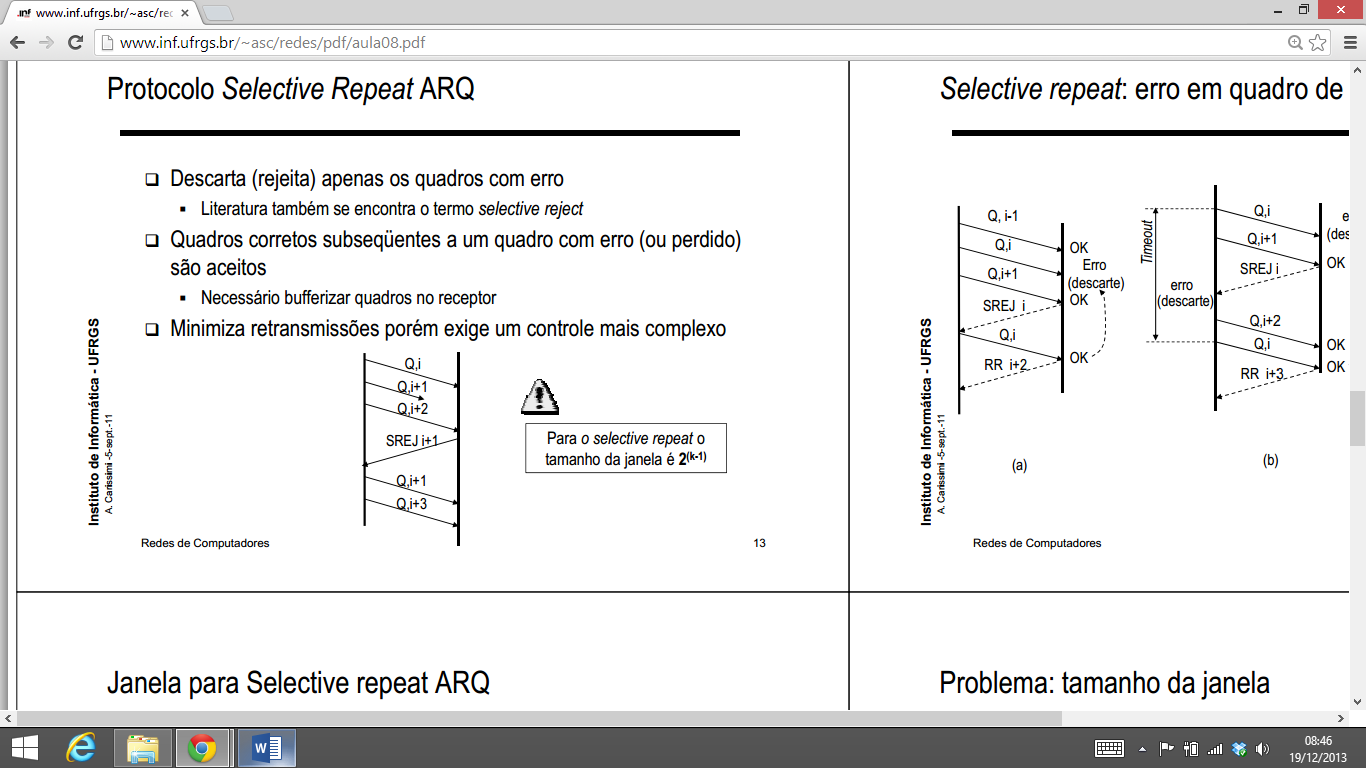
Por fim chegamos a explicação da implementação do go back n essa se dá pela mef descrita na figura 6, também com algumas modificações para implementação de número de janelas variáveis. As principais vantagens desse algoritmo são, a não necessidade de ficar esperando resposta e a menor dependência quanto a time-outs, enquanto a maior desvantagem e a necessidade de reenvio de todos os quadros seguintes a um errado.



6- MEF GO BACK N

### Selective Repeat

O selective repeat e o mais complete algoritmo de controle de fluxo explicado aqui ele e implementado para superar as desvantagens dos algoritmos stop and wait, e o go back n, e principal característica desse e deixa o transmissor sempre enviando e informar apenas pacotes que não foram recebidos, ou recebidos com erros, para que esses possam ser reenviados.



Para o selective repeat o algoritmo transmissor fica apenas enviado dados se se preocupar com acks, porem quando chega uma acks, esse interpreta esse ack e reenvia apenas o quadros indicado pelo ack no próximo envio. Porem esse algoritmo possui uma dependência em

Figura 1 – Selective Report

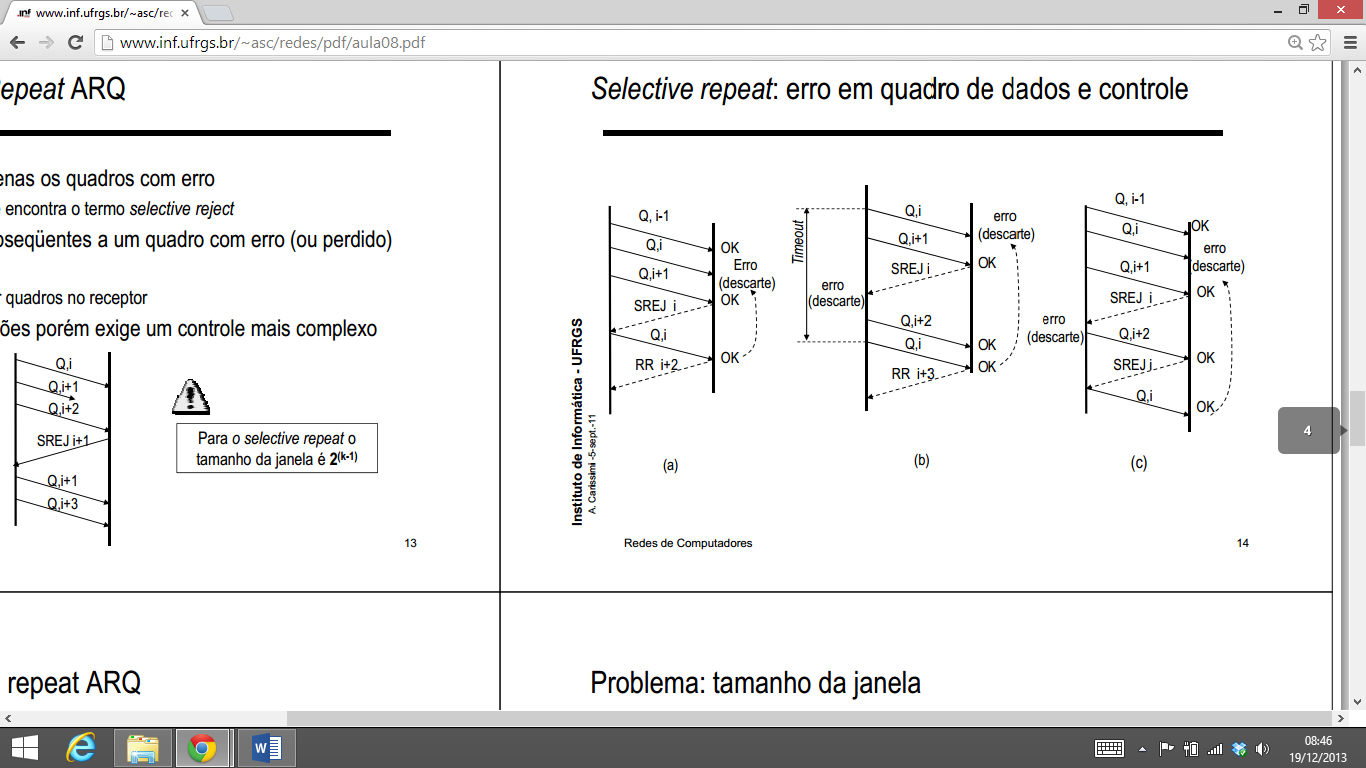
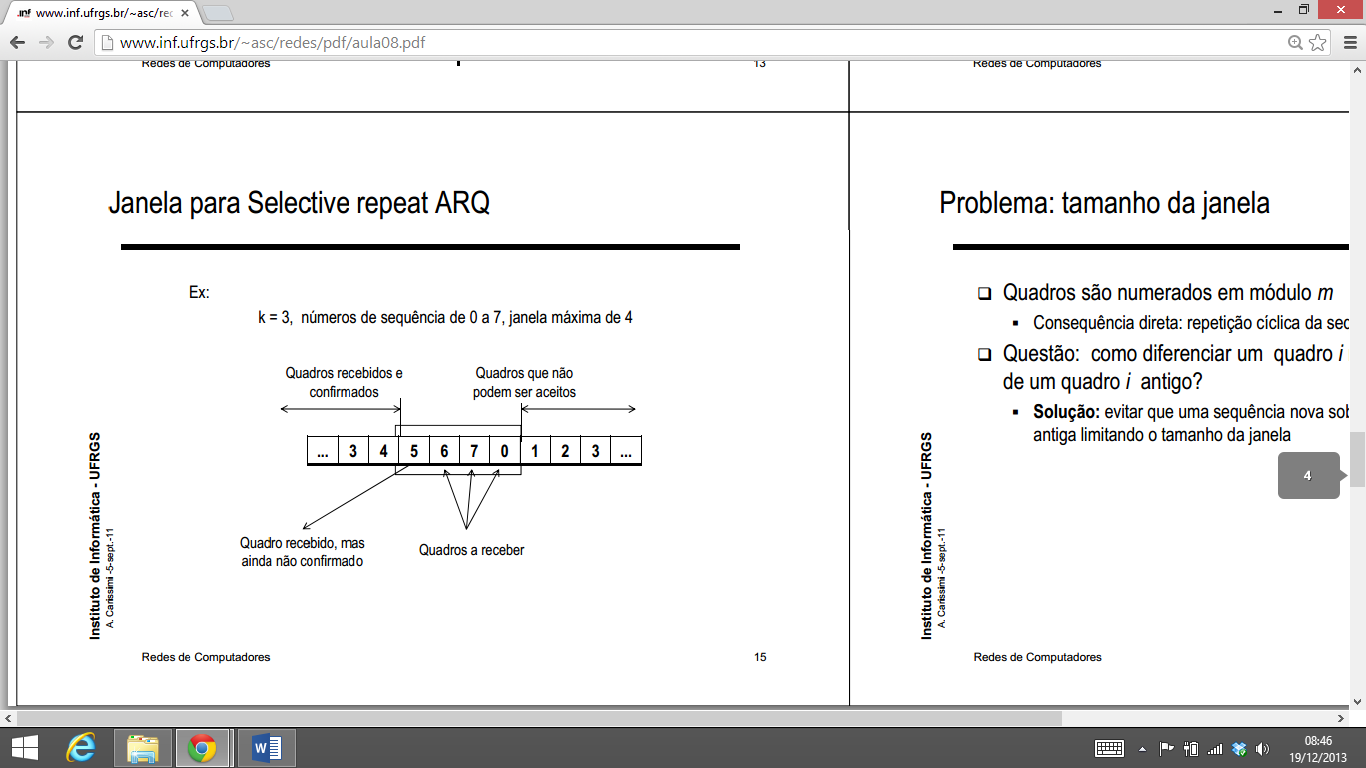


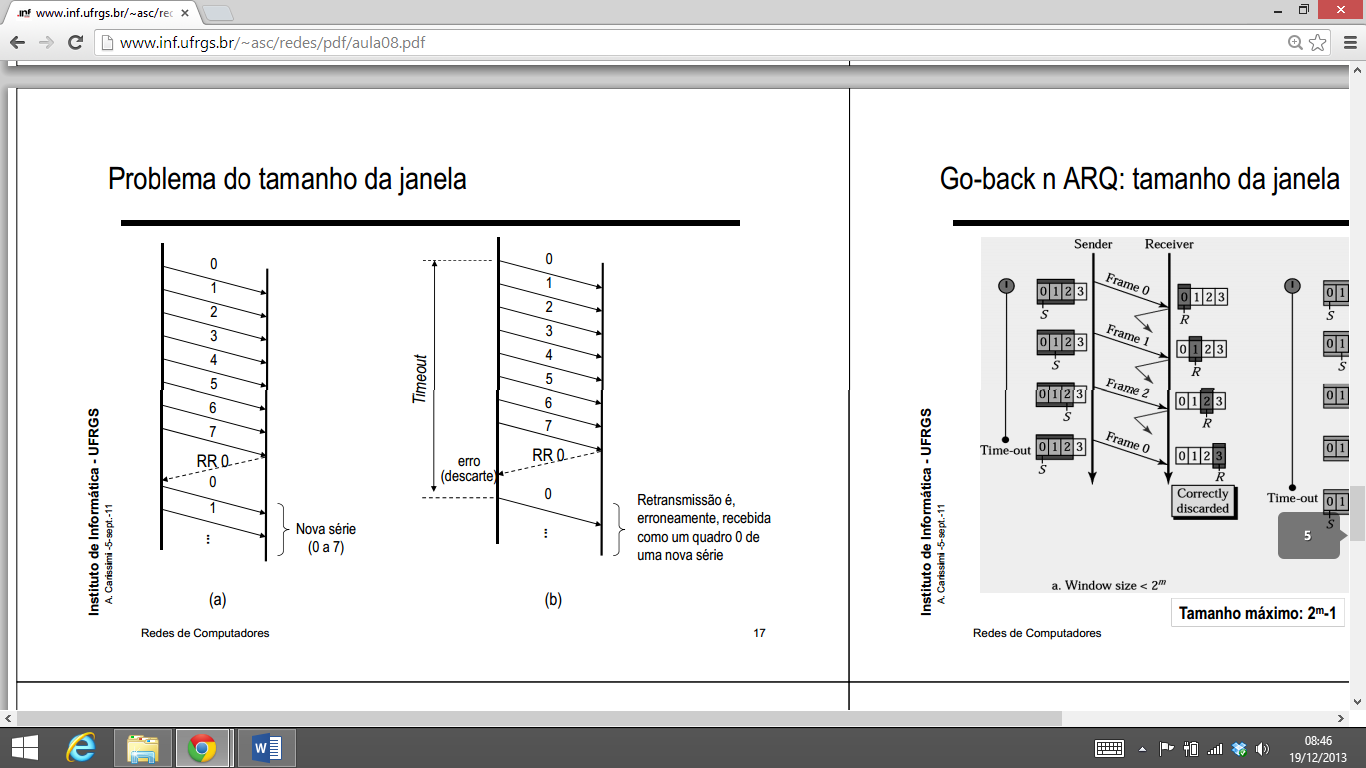
Figura 2 – Fim de janela

. Essa se da por essa não poder transmitir uma outra janela se a confirmação que e o primeiro termo chegou , para isso existe um time out entre o incio de duas janelas tentando garantir assim que uma janela não sobreponha outra causando assim uma erros de interpretação pois os quadros de uma janela seriam interpretados como de outra.

E possível perceber que nesse algoritmo o receptor esta apto a receber tudo logo essas mensagens fora de ordem devem ser armazenadas em um buffer para garantir que não precisaram ser enviadas novamente.

A implementação da mudança de janela para aumentar a eficiência e mandar um quadro de ack para confirmar o recebimento do ultimo quadro da janela como e indicado na figura.

Figura - Janelas



Por fim vimos que esse algoritmo e bastante eficiente em redes erros, porem esse algoritmo se torna bastante complexo de ser implementado pela quantidade de controle que deve ser inserido, pois esse não depende apenas do estado atual. Porem esse tem suprime todas as desvantagens dos algoritmos apresentados anteriormente, e matem as vantagens.

### Equipamentos e ferramentas utilizadas.

Para desenvolvimento da ferramenta de teste foi necessário uma linguagem de baixo nível, no qual o nível de eficiência não influencie nos testes, por isso foi utilizado c++ já que possui esses requisitos e estrutura orientado a objeto facilita a integração. Usado com o compilador g++ v4.6 sem diretivas de otimização em sistema operacional Linux Ubuntu 12.04.

### Restrições da Implementação

A implementação prevê que o usuário irá entrar com dados válidos e que esses dados serão os esperados pelo sistema como, por exemplo, o usuário deve tomar os cuidados necessários para que o número de bits gerados seja um múltiplo inteiro da quantidade de bits por pacote. Ainda, assume-se que o usuário irá escolher valores com apenas duas casas decimais para a probabilidade de erro.

# Procedimentos e Resultados obtidos

Vamos usar Ws = tamanho da janela de transmissão e Wr = tamanho da janela do receptor. Sendo assim, temos:

Ws + Wr = 2^m = 2^2 = 4

sabe-se, ainda, que o tamanho ideal para a janela de transmissão é exatamente a metade da quantidade de números que podemos representar com 2^m bits. Sendo assim, utilizamos Ws = 2.

Stop and Wait Simulado:

O teste foi feito transmitindo 1024 bits, enviados em 128 pacotes de 8 bits cada. Ainda, foram utilizados os seguintes valores: n0 (número de bits de overhead) = 7; nf (número de bits em um frame) = 8 + n0 = 8 + 7 = 15; na (número de bits para o acknoledge) = 1; tempo de propagação = 2 milisegundos; tempo de processamento = 1 milisegundo; tempo de transmissão = 1 milisegundo.

Para o timeout, foi usado um valor suficientemente grande para cobrir 2 tempos de propagação + 2 tempos de transmissão + 1 tempo de processamento :

timeout = 2\*Tprop + 2\*Ttrans + Tproc + 4 = 2\*2 + 2\*1 + 1 + 4 = 11 milisegundos.

Foram realizados dois testes:

1. Sem erros: obteve-se uma eficiência de 60 % (esse valor é o simulado, precisa fazer o teórico com a fórmula que tem nos slides. Para fazer as contas teórica, use R = 0.8 bps. A conta teórica que eu fiz aqui em casa deu aproximadamente 61,3 %, eu só não estou com tempo para escrever essa conta aqui, aí estou deixando para você).
2. Com uma probabilidade de erro de 30%: obteve-se uma eficiência de 41,6 % (esse valor é o simulado, precisa fazer o teórico – é só multiplicar o valor da simulação “sem erros” por 1 – (probabilidade de erro), acho que dá mais ou menos 42 %).

GoBack-N Simulado:

O teste foi feito com os mesmos valores utilizados na simulação do Stop and Wait. Ainda, utilizou-se 2 bits para representar os quadros a serem enviados.

Foram realizados dois testes:

1. Sem erros: obteve-se uma eficiência de 99 %.
2. Com uma probabilidade de erro de 30 %: obteve-se uma eficiência de 37,3 %

# Conclusão

De acordo com os valores obtidos e os valores teóricos, pudemos confirmar que os valores são razoavelmente compatíveis. O Stop and Wait é o protocolo mais lento em casos gerais, com taxas de erro não tão baixas. O Go Back-N, que faz bom uso do tempo que estaria ocioso na transmissão do Stop and Wait, apresenta desempenho mais alto. Entretanto, é possível concluir que o Stop and Wait é mais eficiente do que o Go Back-N para taxas de erro muito altas e tempos de propagação pequenos (em transmissões confiáveis). Para taxas de erro baixas e maiores tempos de propagação, o Go Back-N se torna mais eficiente do que o Stop and Wait. Apesar de não validada a implementação do select repeat, porem para os casos em que exercitada de forma correta ele se mostrou mais eficiente que as duas outras implementações.