CAMADA DE ENLACE

1) Qual a finalidade do protocolo ARP? Por que suas requisições são enviadas dentro de um quadro de *broadcast* e a resposta é enviada com um endereço MAC específico?

Sua finalidade é traduzir endereços IP em endereços MAC. O *broadcast* é usado porque não se sabe para quem se deve enviar a solicitação específica, então se envia para todos. Aquele que possuir o IP desejado retorna seu MAC ao ARP.

2) No CSMA/CD, em redes locais *ethernet*, explique por que é necessário estabelecer uma distância máxima entre duas estações, bem como um tamanho mínimo de quadros. Ilustre sua resposta calculando a distância máxima entre duas estações em uma rede *ethernet* 10mbps, com quadros mínimos de 64 bytes (512 bits) e velocidade de propagação do sinal igual a $2{,}3*10^8$ m/s.

O tempo de transmissão pode ser calculada como L/R, sendo L o tamanho do quadro e R a taxa de transmissão. Assim, o tempo de transmissão é 64bytes/10mbps, o que é 512bits/10^7bits.

Pela fórmula física, V = S/T. Assim, se queremos saber S (que seria a distância), bastaria fazer S = VT. No entanto, deve-se lembrar que, nesse protocolo, a distância é de ida e volta, então a fórmula é 2S = VT, ou 2d = vt.

Sabemos que a velocidade é $2{,}3*10^8$ e que o tempo é $512/10^7$. Assim, a distância é:

$$2D = (2{,}3*10^8)*(\frac{512}{10^7}) \Rightarrow 2D = 2{,}3*512*10 \Rightarrow D = 5888m \Rightarrow D \approx 5{,}9km.$$

3) Se todos os enlaces da internet proverem confiabilidade, o serviço de transmissão confiável do TCP seria redundante? Por quê?

Não, pois a finalidade da confiabilidade na camada de enlace é corrigir erros localmente, e não fim a fim. Por exemplo, pacotes que cheguem fora de ordem poderão ser descartados caso não haja confiabilidade em nível de transporte.

4) Suponha que dois nós comecem a transmitir, ao mesmo tempo, um pacote de comprimento de L bits sobre um canal que permita múltiplos acessos. Seja a taxa de transmissão deste enlace igual a R. Denote por dprop o atraso de propagação entre os dois nós. Se $d_{prop} < L/R$, vai existir colisão detectável? Por quê?

O tempo de transmissão é L/R. D_{prop} é o tempo que o bit levará para chegar ao outro lado. Como os dois nós transmitem ao mesmo tempo, uma colisão será detectada, pois, desta forma, os bits chegam

antes do outro começar a transmitir. Se ambos não transmitissem ao mesmo tempo, teria que ser duas vezes a distância de propagação.

5) Na seção de protocolos de múltiplo acesso, foram listadas quatro características desejáveis para um canal *broadcast*. Quais destas o *slotted aloha* possui? Quais o *token passing* possui?

Slotted ALOHA: único nó ativo envia à taxa total; é descentralizado; é simples.

Token Passing: único nó ativo envia à taxa total; é descentralizado; M nós ativos têm vazão próxima à R/; é simples.

6) Mostre que a eficiência máxima do ALOHA puro é 1/(2e.)

A probabilidade de que um nó tenha transmissão bem-sucedida é $p(1-p)^{2(N-1)}$ e, levando N ao infinito, chega-se à eficiência 1/(2e), que é metade daquela do slotted aloha.

7) São respectivamente consideradas características das redes (L/M/W)AN:

Anulada pelo Márcio por não constar em lugar algum.

c) Velocidade entre 10Mbps e 1Gbps; abrangem tipicamente a área de uma universidade; roteadores conectam diferentes tipos de tecnologia de rede normalmente.

8) Indique V ou F a respeito da camada de enlace:

- Um datagrama pode ser manipulado por apenas um protocolo de enlace no caminho da fonte ao destino.

FALSO. Pode ser manipulado por vários protocolos.

- É responsabilidade de um protocolo da camada de enlace, entre outras, realizar o controle de fluxo e o acesso aleatório.

VERDADEIRO.

- Os quadros de controle CTS e RTS são usados pelo protocolo IEEE 802.11.

VERDADEIRO.

- O protocolo de acesso aleatório usado nas redes *ethernet* é o CSMA/CA.

FALSO. É o CSMA/CD.

- No ALOHA, as estações transmitem quadros sempre que existem dados e depois de monitorar o estado do meio da transmissão.

FALSO. Ele não monitora o estado do meio da transmissão.

9) Em relação ao ARP, é correto afirmar que:

a) O ARP mapeia de IP para endereços MAC apenas para hospedeiros na mesma sub-rede.

CAMADA DE REDE

1) Suponha que um roteador tenha n portas de entrada com velocidades de linha idênticas; n portas de saída com velocidades de linha idênticas, e que a velocidade de linha de uma porta de saída seja, no mínimo, n vezes a velocidade da porta de entrada. Suponha ainda que a velocidade do elemento de comutação é, no mínimo, n vezes maior do que a velocidade de uma porta de entrada. Quando essas condições acontecem não existe formação de fila nem na porta de entrada e nem na porta de saída. Explique o porquê em cada um dos casos, dando um exemplo dos piores casos que podem acontecer (tanto na porta de entrada quanto na porta de saída) e mostrando que, mesmo nestes casos, não tem formação de fila.

O pior caso para entrada seria se chegasse um pacote para cada uma das portas de entrada ao mesmo tempo. Mas o comutador tem velocidade n vezes maior do que a de uma porta de entrada, então ele conseguirá tratar todos os pacotes que chegaram antes que cheguem outros, não formando fila.

O pior caso para a saída seria se o comutador mandasse todos os pacotes para a mesma porta de saída ao mesmo tempo. Como a velocidade da porta de saída é n vezes maior que a de entrada, o comutador sempre conseguirá transmitir pacote na linha de saída antes que cheguem outros, sem fila.

2) O que acontece em cada um dos algoritmos *link-state* (estado do enlace) e *distance-vector* (vetor de distâncias) quando muda o custo de um enlace ligado a um determinado nó?

Link-state: a mudança no custo é notificada para todos os nós da rede, com broadcast. Então, quando um nó executar o algoritmo, o novo valor vai ser conhecido e suas tabelas serão alteradas.

Distance-vector: o nó detecta que o custo do enlace local mudou, e então atualiza sua tabela de distâncias. Quando é o menor custo do caminho, ele avisará a seus vizinhos.

6) A máscara da sub-rede de uma rede na Internet é 255.255.248.0. Qual é o número máximo de hosts que ela pode manipular?

Como 255 é 2^8 (e ele aparece duas vezes), e 248 é 2^8-2^3, então o número de bits da máscara é 8 + 8 + (8 - 3), que dá 21. Como um endereço IPV4 tem, no máximo, 32 bits, e sendo 21 bits para rede, fica 32-21 bits para *host*, isto é, 11 bits para *host.* Assim, o número máximo de *hosts* que ela pode manipular é 2^{11}, ou seja, é 2048.

5) O IPv6 utiliza endereços de 16 bytes (128 bits). Se um bloco de um milhão de endereços for alocado a cada picossegundo (10^{-12} seg.), qual será a duração desses endereços?

Um endereço de 128 bits permite 2^{128} endereços. Um milhão de endereços é 10^6. Um milhão de endereços por segundo pode ser representado por $10^6/10^{-12}$, que dá 10^{18}. Assim, a duração, em segundos, destes endereços, pode ser representada por $2^{128}/10^{18}$. Eu pararia a conta por aqui, porque não faria idéia de como resolver mais. Mas, de acordo com o Márcio, 2^{128} é $3.4*10^{38}$, então levaríamos 10^{13} anos para esgotar todos os endereços IPV6.

3) Um grande número de endereços IP consecutivos está disponível a partir de 198.16.0.0. Suponha que quatro organizações, A, B, C, B, solicitem 4.000, 2.000, 4.000 e 8.000 endereços, respectivamente, e nessa ordem. Para cada uma delas, forneça o primeiro endereço IP atribuído, o último endereço IP atribuído e a máscara na notação w.x.y.z/s.

a) É impossível alocar exatamente 4000 endereços, 2000 endereços etc. Assim, precisamos alocar endereços em um número da base de 2. O menor número possível, dessa forma, é 2^{12} = 4096 endereços. Se temos 12 bits para host, e o máximo de bits que um IPV4 pode ter é 32, então, para a máscara, teremos 32-12=20 bits. Logo, a máscara será 198.16.0.0/20, ou 255.255.240.0 ($2^8.2^8.(2^8-2^4).0$, conforme explicação da questão 5.)

Assim, o primeiro endereço IP, conforme a máscara, será 198.16.0.0. Para calcularmos o último, passamos a parte dos endereços de *host* para binário, deixando tudo que é da parte do *host*, e não da máscara, como 1: 198.16.0000|1111.11111111 = 198.16.15.255.

b) Não tem como alocar 2000 endereços, só 2^{11}, que é 2048. Assim, 32-11 é 21, logo, a máscara é /21. O endereço anterior era 198.16.0000|1111.11111111; corrigindo o bit adicional da máscara, temos que ele é 198.16.00001|111.11111111. Somando 1 a ele para pular ao próximo endereço, temos: 198.16.00010|000.0, que é 198.16.16.0.

Assim, a máscara é 198.16.16.0/21, seu primeiro endereço é 198.16.16.0 e seu último é, respectivamente: 198.16.00010|111.255, que é 198.16.23.255.

c) A máscara será novamente /20, pois serão 4096 endereços alocados. Assim, temos o último endereço anterior, que é 198.16.00010|111.255, e devemos corrigi-lo para nova máscara, ficando como: 198.16.0001|0111.255. Somando 1 a ele para pular para o próximo endereço, teríamos algo como 198.16.24.0. Este endereço, no entanto, não é válido, pois é impossível conseguir um endereço 24 com os 4 bits da máscara. Só é possível 128 (máscara 1000|0000), 64 (0100|0000), 32 (0010|0000), 16 (0001|0000), 0 (0000|0000), e somas entre esses algarismos decimais.

Desta forma, o endereço mais perto possível para essa máscara é o 32. Então, o primeiro endereço será 198.16.0010|0000.0, que é 198.16.32.0 (e a máscara será 198.16.32.0/20.) O último é 198.16.0010|1111.255, ou 198.16.47.255.

d) Para alocar 8000 endereços, temos 2^{13}, que é 8192. Assim, a máscara será /19, pois 32-13=19. O último endereço foi 198.16.47.255. Não poderíamos pular pois 198.16.48.0 não seria compatível com a máscara /19. Observe: 192.16.000|00000.0. Com esses 3 bits que sobraram da máscara, é impossível se ter 48. Apenas 128 (100|00000), 64 (010|00000), 32 (001|00000) e 0 (000|00000), e combinações da soma destes algarismos decimais.

Ora, endereços começando com 0 já foram tomados pela letra A; os começando em 32, também, pela letra C. Resta, então, pegar a máscara que dê 64: 192.168.010|00000.0, que é 192.168.64.0

Destarte, o primeiro endereço seria 192.16.64.0 (e a máscara 192.16.64.0/19), e o último seria 192.16.010|11111.255, ou seja, 192.16.95.255.

4) Um roteador tem as seguintes entradas em sua tabela de roteamento:

Endereço/Máscara	**Hop**
135.46.56.0/22	**i0**
135.46.60.9/22	**i1**
192.53.40.0/23	**r1**
Padrão	**r2**

Para cada um dos endereços IP a seguir, o que o roteador fará se chegar um pacote com esse endereço?

a) 135.46.63.10 **c) 135.46.52.2** **e) 192.53.56.7**

b) 135.46.57.14 **d) 192.53.40.7**

135.46.56.0/22 é a mesma coisa que dizer que:
135.46.001110|00.0, pois a máscara é de 22 bits, logo, ela pega os 16 primeiros bits (do 135.46) e mais 6 adicionais para chegar a 22. Desses 6 adicionais, deve-se ver quais bits precisarão estar ativos (=1) para dar 56.
Agora, para saber o resto, basta colocar 1 em todos os demais campos que não a máscara:
135.46.001110|11.255, que é 135.46.59.255.
Portanto, qualquer endereço entre 135.46.56.0 a 135.46.59.255 irá para a Interface 0.
Pela mesma lógica, qualquer endereço entre 135.46.60.9 a 135.46.63.255 irá para a Interface 1.
E, igualmente, entre 192.53.40.0 a 192.53.41.255 irá para o Roteador 1; demais irão ao Roteador 2.

Com isso:

a) Irá para a I1. c) Irá para R2. e) Irá para R2.

b) Irá para I0. d) Irá para R1.

IMPORTANTE: Se estiver podendo ir para dois lugares diferentes, deve-se escrever o endereço de forma binária e ver qual tem mais bits da máscara batendo. Ele será encaminhado àquele lugar que tiver mais bits parecidos.

APÊNDICE – TABELA DE REPASSE

Roteadores têm uma tabela de repasse. Esta tabela de repasse é calculada pelo processador de roteamento, mas uma cópia dela é comumente armazenada na porta de entrada dos roteadores e é atualizada pelo processador de roteamento quando necessário. Esta descentralização permite que decisões sejam tomadas localmente, evitando gargalo de processamento de repasse.

O processamento da porta de entrada deve ser feito em velocidade de linha, ou seja, deve-se examinar a porta de saída do pacote em tempo menor do que o necessário para receber um pacote na porta de entrada. Este valor é absurdo, então uma busca linear na tabela de repasse é impossível.

Por conta disto, o registro das tabelas de repasse são armazenados em formato de árvore. Desta forma, realiza-se uma busca binária: se o primeiro bit do endereço for 0, o registro da tabela de repasse para o endereço de destino estará à esquerda; se for 1, estará à direita, e assim sucessivamente.

Mesmo essas árvores, porém, têm baixa velocidade se comparada à necessária. Assim, há outras formas de realizar os exames de tabela de repasse, como **memórias de conteúdo endereçáveis**, e algumas melhorias como realizar cache de registros recém-acessados.

Um pacote sai da porta de entrada e, examinada a tabela, é repassado ao elemento de comutação. Às vezes, este pode estar ocupado, então o pacote sai da porta de entrada e é colocado em uma fila do elemento de comutação.

A comutação pode ser por memória: realizada por uma CPU, como se esta realizasse operações de entrada e saída, com interrupções e tudo.

Comutação por barramento: o pacote é transferido diretamente da porta de entrada à de saída por meio do barramento; a única limitação de velocidade é a velocidade do próprio barramento, mas que costuma ser alta o suficiente para LANs e redes corporativas. Se o barramento estiver ocupado, o pacote é colocado na fila da porta de entrada.

Comutação por *crossbar*: usa uma rede sofisticada de interconexão para superar as limitações de velocidade dos barramentos. Nela, os barramentos formam algo parecido com uma matriz: há

barramentos horizontais e verticais para cada porta de entrada e saída. De sua porta de entrada, um pacote percorre um barramento horizontal até que chegue ao barramento vertical de sua porta de saída. Se este estiver bloqueado, o pacote ficará na fila na porta de entrada.

Se as filas nas portas de entrada ou de saída se tornarem muito grande, o espaço de *buffer* do roteador se acaba e ocorre perda de pacote. Os piores casos de filas já foram descritos neste resumo, na questão 1 do teste de camada de rede. O livro fala mais coisas, mas não sei se o Marcio as cobraria…

APÊNDICE – ALGORITMOS DE ROTEAMENTO

A finalidade do algoritmo de roteamento é descobrir um bom caminho do roteador fonte ao destino; geralmente medido pelo menor custo. O algoritmo pode ser global, isto é, o cálculo de menor custo é feito utilizando-se conhecimento completo e global sobre a rede. O cálculo pode ser rodado em um só lugar, ou replicado em vários. A característica principal, no fim, é que ele tem informação completa sobre conectividade e custo de enlaces.

O algoritmo também pode ser descentralizado, sendo o cálculo realizado de forma iteraativa e distribuída. Um nó só sabe os custos dos enlaces ligados diretamente a ele. Por meio de troca de informações com os vizinhos, um nó gradualmente calcula o caminho de menor custo até um destino ou conjunto de destinos.

Estes algoritmos podem ainda ser classificados como estáticos (mudam lentamente ao longo do tempo, geralmente com alguém mudando manualmente as rotas), ou dinâmicos (mudam os caminhos de roteamento à medida que muda a carga de tráfego ou a topologia da rede; são mais sucestíveis a problemas de loops de roteamento e oscilação.)

Por fim, o algoritmo pode ser sensível à carga (varia o custo para refletir o nível de congestionamento), mas a maioria dos algoritmos atuais usados são insensíveis à carga.

Algoritmo de roteamento de estado de enlace (LS, link-state): é um algoritmo de roteamento global. As informações da topologia da rede são conseguidas por meio de uma transmissão *broadcast* de estado de enlace. Assim, todos os nós têm uma visão idêntica e completa da rede. Este algoritmo também é chamado de algoritmo de Dijkstra. Ele é iterativo, e, após a k-ésima iteração, conhece todos os caminhos de menor custo para k nós de destino. Algoritmo complexo demais pra botar aqui, ignorei.

Algoritmo de roteamento de vetor de distâncias (DV, distance-vector): é iterativo, assíncrono e distribuído. Cada nó recebe informação de seus vizinhos diretamente ligados; realiza cálculos; distribui seus resultados para seus vizinhos. Este processo continua até que mais nenhuma informação seja trocada. Os nós não precisam rodá-lo simultaneamente. Basicamente, cada nó envia, regularmente, uma cópia do seu vetor de distâncias a cada um dos vizinhos. Quando um nó recebe um novo vetor de distâncias, ele armazena este vetor e então usa a equação de Bellman-Ford para atualizar seu próprio vetor de distâncias.

VALEU: ALEXIA, CAMILA, NICHOLAS E SIDNEY PELA AJUDA COM ESTE RESUMO!