Resumo da P3 de MC822

Matéria: Capítulos 4.5 até 6.5

4.5 Algoritmos de Roteamento 1

4.5.1 O Algoritmo de Roteamento de Estado de Enlace (LS) 2

4.5.2 O Algoritmo de Roteamento de Vetores de Distâncias (DV) 2

Um Exemplo Detalhado 3

Mudanças no Estado do Enlace 5

Adição de Reversão Envenenada 6

Algoritmo de LS X VP 6

Provas antigas 6

Material de Estudo Alternativo: 11

4.5 Algoritmos de Roteamento

Dado um conjunto de roteadores conectados por enlaces, a finalidade de um algoritmo de roteamento é descobrir um bom caminho entre o roteador de fonte e o roteador de destino. Normalmente um bom caminho é aquele que tem o **menor custo**. Porém, os algoritmos de roteamento podem ser influenciados por questões políticas. Por exemplo, a empresa X é dona do roteador Y e não deixa o roteamento de sua concorrente Z passar pelo seu roteador.

Classificação 1:

Global ou Centralizado: calcula todos os possíveis caminhos para determinar qual o melhor. Precisa da informação completa de todos os links e hosts da rede. Um dos exemplos é o Algoritmo de Dijkstra. Veja este video para entender o algoritmo.

Descentralizado: Usa um método iterativo para calcular o menor caminho, onde cada roteador tem a informação de custo apenas de seus vizinhos. Por exemplo: Vetor de Distâncias.

Classificação 2:

Estático: este tipo de algoritmo raramente muda a topologia das conexões, normalmente por interação humana.

Dinâmico: voltado para mudar constantemente as conexões da rede, em resposta a mudança no custo de uma determinada conexão ou mesmo em períodos predeterminados.

Classificação 3:

Sensível à carga: o custo entre rotas varia dinamicamente para refletir ao congestionamento atual do link.

Insensível à carga: o congestionamento não afeta diretamente o custo entre rotas.

4.5.1 O Algoritmo de Roteamento de Estado de Enlace (Link State)

Um algoritmo de estado de enlace (ex: Algoritmo de *Djikstra* ou Algoritmo de *Prim*) é um algoritmo iterativo, ou seja, a cada iteração é determinado o menor caminho para um nó. Além disso, possui o conhecimento da topologia da rede e de todos os custos de enlaces. Isso é possível pois existe uma transmissão, chamada de **transmissão broadcast de estado de enlace,** onde pacotes de um nó são retransmitidos para todos os outros, chegando ao custo de cada enlace.

A ideia por trás do algoritmo pode ser definida em cinco passos:

1. Descobrir seus vizinhos e aprender seus endereços de rede
2. Medir o roteador ou custo até cada um de seus vizinhos
3. Criar um pacote que informe tudo o que ele acabou de aprender
4. Enviar esse pacote a todos os outros roteadores
5. Calcular o caminho mais curto até cada um dos outros roteadores

Desvantagens:

- Cada nó só computa sua tabela
- Um nó pode avisar custo incorreto de um link
- Convergência **$O(n^2)$** e pode ter oscilações (todos os nós descobrem ao mesmo tempo onde está congestionado e mudam pra onde não está, congestionando outro lugar)

4.5.2 O Algoritmo de Roteamento de Vetores de Distâncias (Distance Vector)

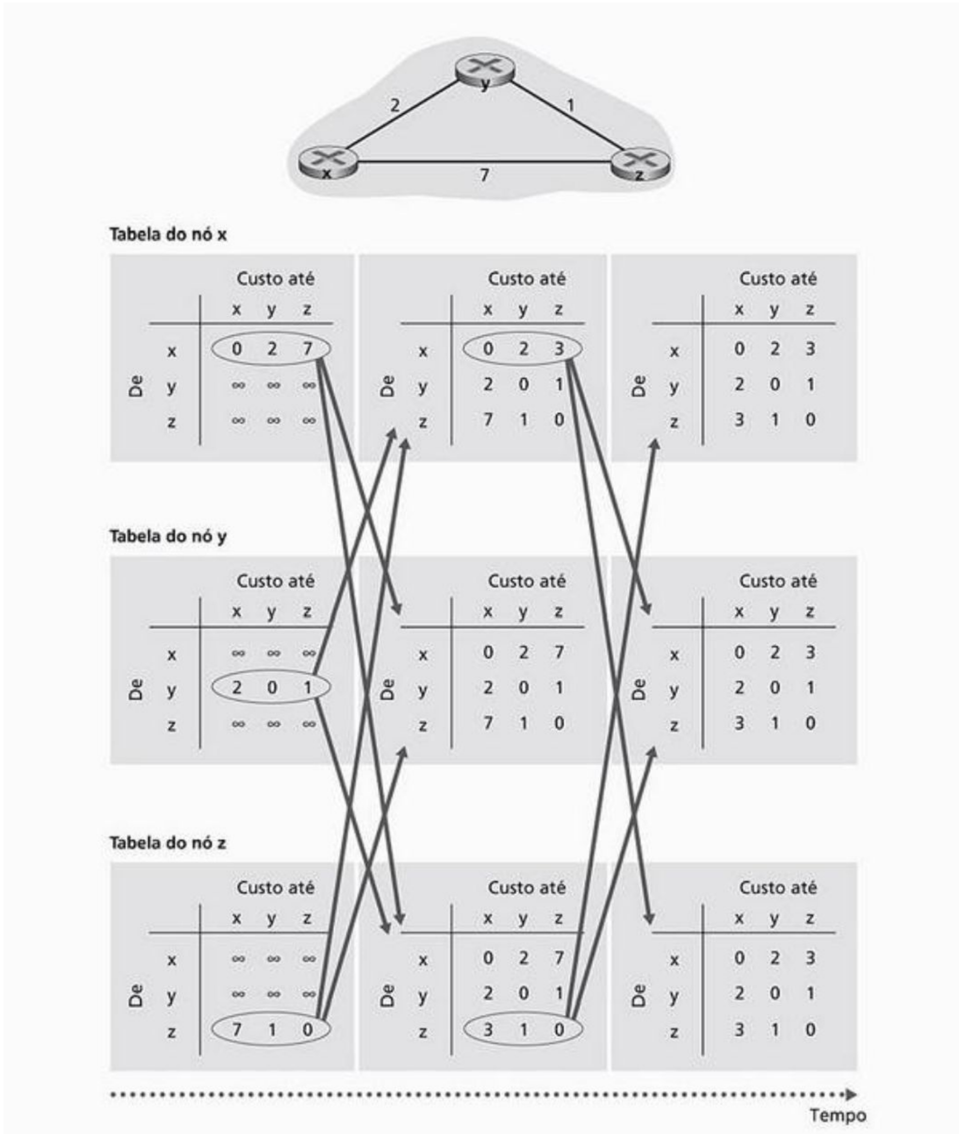
Neste algoritmo, cada nó armazena três tipos de dados:

1. O **custo** até cada um de seus nós vizinhos.
2. O **vetor de distâncias** do nó atual (isto é, as estimativas dos custos de todas as distâncias do nó atual até todos os outros destinos possíveis)
3. Os **vetores de distâncias** de todos os seus **nós vizinhos** (ou seja, as estimativas de custos dos nós vizinhos também)

Em certos intervalos de tempo, cada nó manda o seu vetor de distâncias (tabelão com os custos) pros seus vizinhos. Assim, se o vizinho notar alguma diferença no vetor de distâncias, ele atualiza o seu próprio vetor de distâncias e ainda repassa para todos os seus próprios vizinhos, que por sua vez atualizam os seus vetores de distâncias e por aí vai, repassando nó após nó, pior que corrente de facebook.

Esse algoritmo é iterativo, assíncrono e distribuído. **Iterativo** porque o processo continua até que mais nenhuma informação seja trocada entre os seus vizinhos. **Assíncrono** porque não requer que todos os nós rodem simultaneamente. **Distribuído** porque cada nó, depois que recebe as informações de seus vizinhos, realiza cálculos e distribui os resultados dos seus cálculos para os seus vizinhos.

Um Exemplo Detalhado



Nessa tabela acima, você tem que ler da esquerda pra direita, como se os três quadrados em cada coluna fossem uma coisa só. (Como assim?) Assim, ó:

1. **Quadrado da linha 1, coluna 1:**
   Inicialmente, o nó x só tem as informações dos custos até ele mesmo ($D_x(x) = 0$) e seus vizinhos ($D_x(y) = 2$ e $D_x(z) = 7$). Assim, o resto tá marcado como infinito. Até aí beleza né?
2. **Quadrado da linha 2, coluna 1:**
   De forma semelhante ao quadrado anterior, o y só tem as informações dos custos dele mesmo $D_y(y) = 0$ e dos seus vizinhos ($D_y(x) = 2$ e $D_y(z) = 1$.
3. **Quadrado da linha 3, coluna 1:**
   A mesma coisa dos anteriores, só que pro Z.

4. **Quadrado da linha 1, coluna 2:**
Agora, o x pega as informações calculadas nos passos (2) e (3) pra preencher as suas demais linhas, que antes estavam com símbolos de infinito e agora serão [2, 0, 1] e [7, 1, 0]. Com essas novas informações, ele recalcula o seu próprio vetor de distâncias, que antes era:

$$[\,D_x(x)\,,\; D_x(y)\; D_x(z)\,] = [0, 2, 7].$$

E agora será recalculado segundo a fórmula de Bellman-Ford.

$$D_x(y) = min\{\, c(x,y) + D_y(y)\,,\; c(x,z) + D_z(y)\}$$

Dessa forma, temos:

$D_x(x)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.

$D_x(y) = min\{\, c(x,y) + D_y(y),\; c(x,z) + D_z(y)\}$ = min{ 2 + 0, 7 + 1 } = 2

$D_x(z) = min\{\, c(x,z) + D_z(z),\; c(x,y) + D_y(z)\}$ = min{ 7 + 0, 2 + 1 } = 3

Portanto, o novo vetor de distâncias do x será:

$$[\; D_x(x)\,,\; D_x(y)\; D_x(z)\,] = [0, 2, 3]$$

5. **Quadrado da linha 2, coluna 2:**
O y faz o mesmo que o x fez no passo anterior, porém ele pega as informações do x e do z ao mesmo tempo que o x fez isso (lembra que eu falei que era pra ler como se os três quadrados em cada coluna fossem uma coisa só?) ou seja, o x ainda não atualizou seus valores ainda. Portanto o y pega o valor antigo do vetor de distâncias do x: [0, 2, 7] .

Recalculando o vetor de distâncias do y:

$D_y(x) = min\{\, c(y,x) + D_x(x),\; c(y,z) + D_z(x)\}$ = min{ 2 + 0, 1 + 7 } = 2

$D_y(y)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.

$D_y(z) = min\{\, c(y,z) + D_z(z),\; c(y,x) + D_x(z)\}$ = min{ 1 + 0, 2 + 7 } = 1

Portanto o vetor de distâncias do y não muda, continua [2, 0, 1] e todo esse trabalho foi em vão. (Pois é, não fique triste... é a vida).

6. **Quadrado da linha 3, coluna 2:**
O z faz o mesmo que o y e x fizeram nos passos anteriores.

Recalculando o vetor de distâncias do z:

$D_z(x) = min\{\, c(z,x) + D_x(x),\; c(z,y) + D_y(x)\}$ = min{ 7 + 0, 1 + 2 } = 3

$D_z(y) = min\{\, c(z,y) + D_y(y),\; c(z,x) + D_x(y)\}$ = min{ 1 + 0 , 7 + 2 } = 1

$D_z(z)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.

Portanto, o novo vetor de distâncias do z será:

$$D_z(x)\,,\; D_z(y)\; D_z(z)\,] = [3, 1, 0]$$

7. **Quadrado da linha 1, coluna 3:**
O x recebe os valores dos vetores distância de seus vizinhos já atualizados (no caso do nosso exemplo, o z mudou de [7, 1, 0] pra [3, 1, 0] e o y se manteve a mesma coisa.

$D_x(x)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.
$D_x(y) = min\{\, c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$ = min{ 2 + 0, 7 + 1 } = 2
$D_x(z) = min\{\, c(x,z) + D_z(z),\ c(x,y) + D_y(z)\}$ = min{ 7 + 0, 2 + 1 } = 3

Ou seja, mantém-se o valor [0, 2, 3] para o vetor de distâncias do x.

8. **Quadrado da linha 2, coluna 3:**
O y recebe os valores dos vetores distância de seus vizinhos já atualizados (no caso do nosso exemplo, o x mudou de [0, 2, 7] pra [0, 2, 3] e o z mudou de [7, 1, 0] pra [3, 1, 0].

$D_y(x) = min\{\, c(y,x) + D_x(x),\ c(y,z) + D_z(x)\}$ = min{ 2 + 0, 1 + 3 } = 2
$D_y(y)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.
$D_y(z) = min\{\, c(y,z) + D_z(z),\ c(y,x) + D_x(z)\}$ = min{ 1 + 0, 2 + 3 } = 1

Ou seja, mantém-se o valor [2, 0, 1] para o vetor de distâncias do y.

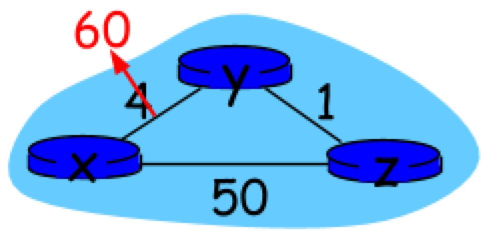
9. **Quadrado da linha 3, coluna 3:**
O z recebe os valores dos vetores distância de seus vizinhos já atualizados (no caso do nosso exemplo, o x mudou de [0, 2, 7] pra [0, 2, 3] e o y se manteve o mesmo.

$D_z(x) = min\{\, c(z,x) + D_x(x),\ c(z,y) + D_y(x)\}$ = min{ 7 + 0, 1 + 2 } = 3
$D_z(y) = min\{\, c(z,y) + D_y(y),\ c(z,x) + D_x(y)\}$ = min{ 1 + 0 , 7 + 2 } = 1
$D_z(z)$ permanece a mesma coisa, pois a distância dele a ele mesmo continua zero.

Ou seja, mantém-se o valor [3, 1, 0] para o vetor de distâncias do z e está terminado o algoritmo (graças a Deus).

Mudanças no Estado do Enlace

Boas notícias viajam rápido, portanto se diminuir algum custo até algum vizinho, o nó em questão rapidamente avisa os vizinhos e estes recalculam seus vetores de distâncias para assim escolhem o menor caminho. Porém, más notícias demoram a chegar: se o custo de um nó aumenta consideravelmente, como na figura abaixo, surge o **problema da contagem ao infinito.**



Inicialmente c(z,x) = c(z,y) + c(y,x) = 4 + 1 = 5, passando por y.

Quando c(x,y) aumenta para 60, o nó y sabe que chega em Z com custo 1 e que c(z,x) é 5. Então, o y fala que c(y,x) = 6.

Depois, o nó z verifica que chega em y com custo 1 e que c(y,x) = 6, portanto c(z,x) = 7 e assim por diante, até cagar tudo de uma vez.

Adição de Reversão Envenenada

É uma técnica para evitar **loops de roteamento**. Se a rota de Z para chegar a X passa por Y, então o Z "mente" pro Y que a sua distância ao X é infinita ($D_z(x) = \infty$). Essa técnica não funciona para loops com nós com mais de dois vizinhos e nem resolve o **problema da contagem ao infinito.**

Algoritmo de LS X VP

Provas antigas

1. Duas Redes Ethernet E1 e E2 estão ligadas via um roteador R. Uma máquina H1 em E1 deseja enviar um pacote de dados para a máquina H2 em E2. Suponha que em H1 a tabela ARP tenha informação para o IP do roteador (que roteia para o IP de H2), e que em R a tabela ARP não tenha informação para o IP de H2. Quantos quadros trafegam em E1 e em E2? Monte uma tabela mostrando para cada quadro, qual o MAC fonte, MAC destino, IP fonte, IP destino e tipo do quadro. H1 tem MAC1 e IP1, H2 tem MAC2 e IP2, a interface de R em E1 tem MACr1 e IPr1 e a interface R em E2 tem MACr2 e IPr2. (valor 3,0)

(Questão Resolvida)

Em E1: Como a tabela ARP de H1 possui a informação (mapeamento entre IP e endereço MAC) para o IP do roteador, H1 conhece o endereço MAC do roteador (MACr1) e, assim, só precisa enviar um frame com a mensagem a ser entregue para H2.

Em E2: /Como a tabela ARP do roteador não tem a informação para o IP de H2, ele precisa enviar uma mensagem broadcast solicitando o endereço MAC de H2. Então, H2 responde seu endereço MAC (MAC2) e, por fim, o roteador envia a mensagem de H1 para H2. Totalizando, assim, 3 frames.

No total, são 4 quadros. O primeiro em E1 e o restante em E2.

Nº do frame	Tipo do frame	MAC fonte	MAC destino	IP fonte	IP destino
1	dados	MAC1	MACr1	IP1	IP2
2	ARP	MACr2	FF-FF-FF-FF-FF-FF	IPr2	IP2
3	ARPACK	MAC2	MACr2	IP2	IPr2
4	dados	MACr2	MAC2	IP1	IP2

(Questão Resolvida, mas não sei se foi resolvida da maneira correta)

2. O CSMA/CD usa um mecanismo de contenção para diversas estações obterem acesso ao meio compartilhado. Uma outra solução seria dividir o canal em múltiplos sub-canais (divisão no tempo ou frequência) e alocá-los estaticamente. Da teoria de filas para tempos de chegada e serviço de Poisson temos:
$T = 1 / (uC - L)$
onde T é o atraso médio para um canal de capacidade C bps, com uma taxa de chegada de L quadros/s, cada quadro tendo um comprimento médio de 1/u b/quadro (com função de densidade de probabilidade exponencial).

a) Qual o atraso T se a capacidade C é 100 Mbps, o comprimento médio do quadro é 1/u = 10000 b e a taxa de chegada de quadros L é 5000 quadros/s?

$T = 1/(10^{-4}.10^{8}-5.10^{3}) = 1/(10.10^{3}-5.10^{3}) = 1/(5.10^{3}) = 2.10^{-4}$ s/quadro

b) Se dividirmos o canal em N =10 sub-canais independentes, cada um com capacidade de C/N bps, qual será o novo atraso T?

Primeiramente, tentei trocar só o C da fórmula por C/N, mas aí o atraso deu negativo.
Então acho que mais alguma coisa irá mudar ao dividir em N subcanais. Além disso, acho q não faz sentido alterar a quantidade de bits por quadro (1/u). Então, na prova, chutaria que a taxa de chegada (L) também seria dividida por N, assim teríamos:
$T = 1/(10^{-4}.10^{7}-5.10^{2}) = 1/(10.10^{2}-5.10^{2}) = 1/(5.10^{2}) = 2.10^{-3}$ s/quadro

c) Compare os dois valores de T e explique o que ocorre com o atraso quando dividimos o canal em sub-canais. (valor 2,5)

O atraso ficou 10 vezes maior, pois cada canal terá uma capacidade N=10 vezes menor. Observe, entretanto, que se N=10 frames fossem enviados (um em cada canal), o atraso total seria 2.10^{-3} s. Portanto, seria equivalente ao atraso para enviar 10 frames antes de dividir o canal, afinal estaria

usando a mesma capacidade (100% do canal original) para enviar a mesma quantidade de dados (10 frames).
Além disso, se não houver 10 frames para serem transmitidos ao mesmo tempo, o canal será subutilizado, gerando um atraso maior do que se utilizasse o canal antes da divisão, o que é esperado.

(Questão Resolvida)

3. Um nó móvel H1 (com H1MAC) está associado a um AP (com APMAC) através do protocolo IEEE 802.11. O AP está conectado a um roteador R1 (com R1MAC) através do protocolo IEEE 802.3. Ao H1 enviar uma mensagem para a Internet, mostre os endereços de enlace (fonte, destino, etc) do quadro IEEE 802.11 entre H1 e AP, e do quadro IEEE 802.3 entre AP e R1. (valor 2,0)

Entre H1 e AP (802.11):

- Endereço 1 ⟶ APMAC (endereço MAC do host ou AP q vai receber o frame pelo protocolo IEEE 802.11)
- Endereço 2 ⟶ H1MAC (endereço MAC do host ou AP transmitindo o frame)
- Endereço 3 ⟶ R1MAC (endereço MAC do roteador conectado ao AP)

Entre AP e R1 (802.3):

- Endereço fonte ⟶ H1MAC
- Endereço destino ⟶ R1MAC

(Questão Resolvida)

1. Considere uma subrede composta por 5 roteadores (A, B, C, D, e E) que utilize roteamento baseado em Vetores de Distância. O roteador C tem como vizinhos os roteadores B, D e E. Os seguintes vetores com os custos para A,B,C,D e E chegam no roteador C:

(1,0,3,7,15) de B; (8,2,4,0,5) de D; e (5,14,8,2,0) de E. 5+2+4
8

Os atrasos medidos para B, D e E são 5, 2 e 10, respectivamente. Qual será a nova tabela de roteamento (enlace de saída, atraso esperado) de C? Justifique a resposta. (valor 2,5)

O roteador *C* usa as informações recebidas para recalcular seu próprio vetor de distâncias através da fórmula de Bellman-Ford. Assim, temos:

$$D_C(A) = min\{\, c(C,B) + D_B(A),\ c(C,D) + D_D(A),\ c(C,E) + D_E(A) \,\}$$
$$D_C(A) = min\{\, 5 + 1,\ 2 + 8,\ 10 + 5 \,\} = 6$$

$$D_C(B) = min\{\, c(C,B) + D_B(B),\ c(C,D) + D_D(B),\ c(C,E) + D_E(B) \,\}$$
$$D_C(B) = min\{\, 5 + 0,\ 2 + 2,\ 10 + 14 \,\} = 4$$

$D_C(C) = 0$, pois a distância dele a ele mesmo continua zero.

$$D_C(D) = min\{\, c(C,B) + D_B(D),\ c(C,D) + D_D(D),\ c(C,E) + D_E(D) \,\}$$
$$D_C(D) = min\{\, 5 + 7,\ 2 + 0,\ 10 + 2 \,\} = 2$$

$$D_C(E) = min\{ c(C,B) + D_B(E),\ c(C,D) + D_D(E),\ c(C,E) + D_E(E) \}$$
$$D_C(E) = min\{ 5 + 15,\ 2 + 5,\ 10 + 0 \} = 7$$

Assim, o novo vetor de distâncias de C será:

$$[D_C(A),\ D_C(B),\ D_C(C),\ D_C(D),\ D_C(E)] = [6, 4, 0, 2, 7]$$

Portanto, a nova tabela de roteamento (enlace de saída, atraso esperado) de C será:

	Atraso de A	Atraso de B	Atraso de C	Atraso de D	Atraso de E
Enlace de saída: C	6	4	0	2	7
Enlace de saída: B	1	0	3	7	15
Enlace de saída: D	8	2	4	0	5
Enlace de saída: E	5	14	8	2	0

(Questão Resolvida)

Quarta Questão:

Descreva o funcionamento do protocolo ARP das redes ethernet (valor 2.0).

O protocolo ARP (Address resolution protocol) é utilizado para obter o endereço MAC dos nós a partir de seu endereço IP. Cada nó possui uma tabela ARP, cujas entradas realizam o mapeamento entre endereço IP e endereço MAC para os nós conhecidos.

Se um nó A precisa obter um endereço MAC desconhecido, ele envia uma mensagem broadcast tendo como IP destino o IP para o qual ele deseja conhecer o MAC; e, como MAC destino, FF-FF-FF-FF-FF-FF, indicando que é um broadcast. Esta mensagem é recebida por todos os nós da rede, mas respondida apenas pelo nó cujo IP é compatível com o IP destino. Este nó envia uma mensagem unicast, tendo, como MAC destino, o MAC de A e, como MAC origem, o seu MAC. Assim, o nó A recebe esta mensagem e adiciona uma entrada em sua tabela ARP.

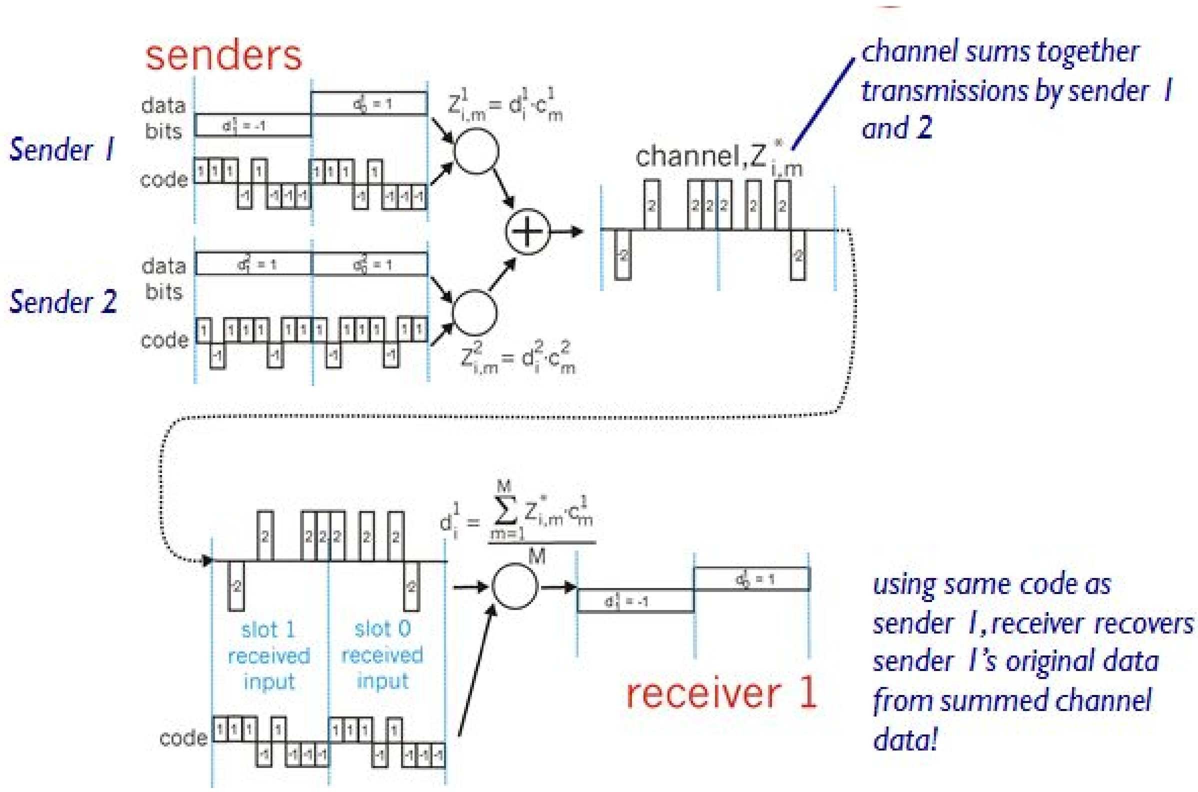
(Questão Resolvida)

Quinta Questão:

Em redes CDMA duas transmissões simultâneas não significa colisão dos quadros. Diga a razão pela qual os receptores de quadros transmitidos simultaneamente são capazes de recuperar integralmente o quadro enviado pelo transmissor (valor 1.0).

Em redes CDMA, cada usuário possui um código para codificação dos dados (“chipping” sequence), a qual é realizada através do produto entre seu código e os dados. Se mais de um usuário transmitir ao mesmo tempo, os sinais se somam.
Se os códigos dos usuários forem ortogonais, posteriormente será possível recuperar os dados. Para isso, basta realizar o produto interno do sinal recebido e o código utilizado.

Exemplo:



Material de Estudo Alternativo:

Resumo do Danilo Charântola:
https://docs.google.com/document/d/1-ePYPFtGlwMeTlAdT8YSyCeW8shuawckDB7zkEwYeiY/edit?usp=sharing