Réseaux 6e édition

Andrew Tanenbaum, Nick Feamster, David Wetherall

Corrigés des exercices

Chapitre 3 : La couche Liaison de données

ISBN : 978-2-3260-0239-5

01. Ethernet n'utilise pas le remplissage de bits pour éviter l'apparition du préambule dans la partie des données. Lorsque cela se produit, la trame est perdue et le récepteur se resynchronise au début de la trame suivante. Ce problème survient rarement car le préambule est long (64 bits, y compris le SFD).

02. Après remplissage, on obtient : A B ESC ESC C ESC ESC ESC FLAG ESC FLAG D.

03. La surcharge maximale se produit lorsque la charge utile ne se compose que des octets ESC et FLAG. Dans ce cas, elle est de 100 %.

04. 0110 0111 1101 1110 1111 11.

05. C'est possible. Supposons que l'original contienne la séquence binaire 01111110 en tant que donnée. Après remplissage de bits, la séquence devient 011111010. Si le deuxième 0 est perdu suite à une erreur de transmission, le récepteur reçoit 01111110, qu'il interprète comme marqueur de fin de trame. Il revient alors en arrière pour trouver la somme de contrôle juste avant la fin de trame afin de la vérifier. Si la somme est sur 16 bits, il y a une chance sur 2^{16} qu'elle soit correcte par accident, ce qui fait accepter une trame qui aurait dû être rejetée. Plus la somme de contrôle est longue, plus faible est le risque de laisser passer une erreur indétectée, mais la probabilité n'est jamais nulle.

06. Comme chaque trame a une probabilité de 0,8 d'arriver intacte, la probabilité qu'un message global arrive intact est de 0,810, soit de l'ordre de 0,107. Appelons *p* cette valeur. Le nombre moyen de transmissions *E* pour un message global est de :

$$E = \sum_{i=1}^{\infty} ip(1-p)^{i-1} = p\sum_{i=1}^{\infty} i(1-p)^{i-1}$$

Pour réduire cette équation, on utilise la formule bien connue de la somme d'une série géométrique infinie :

$$S = \sum_{i=1}^{\infty} \alpha^i = \frac{1}{1-\alpha}$$

La dérivée de cette expression relativement à α donne :

$$S' = \sum_{i=1}^{\infty} i\alpha^{i-1} = \frac{1}{(1-\alpha)^2}$$

Considérons alors $\alpha = 1 - p$ pour obtenir, après réduction, E = 1/p. On obtient alors E = 1/0,107, ce qui correspond à 9,3 transmissions.

07. Si les délais de propagation sont très longs (liaison vers une sonde spatiale à proximité d'une planète comme Mars ou Vénus), une autocorrection d'erreur (utilisant, par exemple, un code de Hamming) est préférable. Elle est également appropriée dans certaines situations (militaires par exemple), où celui qui reçoit ne veut pas réémettre pour ne pas se faire repérer. Tout va bien, dans ce cas, si le taux d'erreur est relativement faible pour ne pas entraîner de perturbation sur les performances. Enfin, dans les systèmes temps réel, le temps d'attente relatif à la retransmission est tout à fait intolérable.

08. Le fait de ne changer qu'un seul bit sur un caractère valide n'entraîne aucune correspondance avec un autre caractère valide, étant donné la nature même du bit de parité. En revanche, la modification de deux bits en contrôle de parité paire ou impaire entraîne une correspondance avec un caractère valide. D'où une distance de Hamming de 2 pour ce type de code.

09. L'envoi en double de chaque message correspond à une distance de Hamming de 2 : un changement se produisant au même endroit dans la première et dans la seconde partie du message peut générer un autre mot de code valide. Le taux de code est égal à 1/2. La situation est bien meilleure en ajoutant un seul bit de parité : la distance de Hamming est aussi égale à 2, mais le taux de code devient $m/(m + 1)$.

10. Lors de l'envoi en double du message, la probabilité dépend du nombre de bits du message *m* :

$$\frac{m}{\binom{2m}{2}}$$

Un bit de parité unique ne permet pas de détecter les double erreurs. Leur probabilité de passer inaperçues est égale à 1. En émettant le message deux fois, on capte plus d'erreurs.

11. La valeur codée est 101001001111.

12. 1010 0011 1010

13. Supposons que la numérotation des bits se fasse de gauche à droite, à partir du bit 1. Dans l'exemple, le bit 2, qui est un bit de parité, est incorrect. Les 12 bits transmis (après encodage de Hamming) avaient pour valeur 0xA4F. La valeur hexadécimale d'origine était donc 0xAF.

14. Non, ils ne le peuvent pas. Avec une distance de 3, tous les mots de code altérés sont à une inversion de bit de distance d'un mot de code valide. Si nous utilisons ces mots incorrects pour la correction, nous n'en avons plus pour la détection. Il nous faut donc une distance de 4 pour pouvoir corriger les erreurs sur un bit et détecter les erreurs sur deux bits. Pour une distance de Hamming *n*, jusqu'à $n < 1$ erreurs de bits pourront être détectées et jusqu'à $n/2$ erreurs de bits corrigées, mais pas simultanément.

15. Oui. Les codes de Hamming permettent de corriger les erreurs uniques en utilisant la limité théorique basse du nombre de bits de redondance, c'est-à-dire $(m + r + 1) \geq 2^r$. La conversion d'octets en bits donne $(128 + 8 + 1)\ 2^8$. Ainsi, un octet suffit.

16. Une erreur simple provoque des contrôles de parité erronés, en horizontal et en vertical. Deux erreurs peuvent être également détectées. En effet, si elles sont dans des lignes différentes, le contrôle de parité vertical les détecte. Si elles sont dans la même ligne, le contrôle de parité horizontal les détecte.

Les erreurs triples peuvent aussi être détectées. Si elles sont dans la même ligne ou la même colonne, le contrôle de parité horizontal ou vertical les détectera. S'il y a deux erreurs sur la même ligne, le contrôle de parité vertical de la troisième fera apparaître une erreur. De même s'il y a deux erreurs sur la même colonne.

Mais une erreur quadruple portant sur 4 bits situés aux quatre coins d'un rectangle ne serait pas détectée.

17. Une erreur d'un seul bit entraîne une faute de parité sur la ligne et sur la colonne. Le bit défectueux peut donc être trouvé en cherchant l'intersection ligne/colonne de cette parité en défaut. Par ailleurs, si deux bits de la même ligne sont en erreur, les deux colonnes concernées auront une faute de parité. Ce mécanisme peut donc corriger des erreurs sur 1 bit et détecter des erreurs sur 2 et sur 3 bits, comme expliqué dans la réponse précédente.

18.

$$\binom{m+r}{2} + m + r + 1 \leq 2^r$$

19. Chercher à détecter de plus en plus d'erreurs sur un bit provoque une chute sensible du débit. L'adoption de méthodes probabilistes complexes maintient le taux de code assez élevé, au risque de rendre la détection d'erreur moins implacable.

20. L'équation 3.1 indique que si l'on veut utiliser un code de Hamming, il faut 10 bits de contrôle par bloc d'où 1 010 bits transmis par bloc. Avec un mécanisme de détection d'erreur, on ne transmet qu'un bit de parité par bloc. Supposons que le taux d'erreur soit de x par bit. Un bloc peut alors avoir un bit erroné 1 000x fois. Chaque fois que l'on a une erreur, il faut retransmettre 1 010 bits. Au total, on transmet 1 001 + 1 000x × 1 001 bits. Pour que la détection d'erreur suivie d'une retransmission soit plus efficace, il faut que 1 001 + 1 000x × 1 001 < 1 010, ce qui nous conduit à un taux d'erreur qui doit être inférieur à 9×10^{-6}.

21. Considérons les différentes configurations erronées sous forme d'une matrice de n lignes et k colonnes, dans laquelle tout bit correct est représenté par un 0 et tout bit erroné par un 1. Avec quatre erreurs par bloc, chaque bloc doit avoir exactement quatre bits à 1. Combien y a-t-il alors de tels blocs ? Il y a nk façons de choisir où placer le premier bit à 1, $nk - 1$ façons de choisir où placer le deuxième, etc. Donc le nombre de blocs est $nk(nk - 1)(nk - 2)(nk - 3)$. Les erreurs non détectées ne surviennent que lorsque les 4 bits à 1 sont aux quatre coins d'un rectangle. Si on utilise les coordonnées cartésiennes, chaque bit à 1 est à (x, y), où $0 \leq x < k$ et $0 \leq y < n$. Supposons que le bit le plus près de l'origine (l'angle en bas à gauche du rectangle) soit à (p, q). Le nombre de rectangles légaux est $(k - p - 1)(n - q - 1)$. On peut trouver le nombre total de rectangles en faisant une sommation sur tous les p et q possibles. La probabilité d'avoir une erreur non détectée est donc le nombre de ces rectangles divisé par le nombre de façons de distribuer les quatre bits :

$$\frac{\sum_{p=0}^{k-2}\sum_{q=0}^{n-2}(k-p-1)(n-q-1)}{nk(nk-1)(nk-2)(nk-3)}$$

22. Pour obtenir la somme de contrôle, il faut calculer la somme des mots en complément à un, ce qui est la même chose que la somme modulo 24 avec débordement des bits de poids forts sur les bits de poids faible :

    0011 + 1010 = 1101

    1101 + 1100 = 1001 + 1 = 1010

    1010 + 1001 = 0011 + 1 = 1100.

    La somme de contrôle Internet est donc 1100.

23. Le reste est : $x^2 + x + 1$.

24. La trame à transmettre est 10011101. Le générateur vaut 1001. Le polynôme obtenu, après ajout de trois 0 à la trame, est 10011101000. Le reste de la division du polynôme par le générateur est 100. La chaîne de bits effectivement transmise par l'émetteur est 10011101100. Le flux binaire reçu, qui présente une erreur sur le troisième bit à partir de la gauche, est 10111101100. En divisant ce polynôme par 1 001, on obtient 100 comme reste (donc différent de 0). Le récepteur détecte l'erreur et demande alors une retransmission.

26. Le code de contrôle CRC est calculé pendant la transmission pour être ajouté en fin de flux de sortie juste après le dernier bit de contenu. Si le CRC avait été placé dans l'en-tête, il aurait fallu lire la trame une première fois pour calculer le code avant de le transmettre, puis une seconde fois pour la transmettre. En ajoutant le CRC à la fin, on divise le travail par deux.

27. Oui, c'est possible si l'acquittement arrive correctement mais après que le temporisateur de l'émetteur a expiré. Cela peut se produire en particulier si le récepteur, dont l'UC est surchargée par d'autres tâches, met longtemps à envoyer son acquittement.

28. L'efficacité est de 50 % lorsque le temps de transmission de la trame est égal au délai de propagation aller-retour sur la ligne. Cela correspond, pour une vitesse de transmission de 4 bit/ms, à transmettre 160 bits (en 40 ms). Dans ce cas, plus le nombre de bits à transmettre est supérieur à 160, plus le protocole de type « arrêt et attente » est efficace.

29. Si une fenêtre de taille 20 permet d'obtenir presque 100 % d'efficacité de bande passante, $2BD + 1 \approx 20$. Le protocole arrêt-attente suppose une taille de fenêtre d'envoi de 1 qui apporte à peu près $1/20 = 5$ % d'efficacité de bande.

30. En cas d'utilisation du protocole arrêt-attente, 25 % d'efficacité de bande passante signifie

$$\frac{t_s}{t_r + t_s} = \frac{1}{4}$$

avec t_s le temps requis pour placer une trame dans le canal et t_r le délai de propagation aller-retour.

$$\frac{t_s}{100 + t_s} = \frac{1}{4} \rightarrow t_s = \frac{100}{3}$$

S'il faut 100/3 de secondes pour envoyer une trame de 900 bits sur le canal, la bande passante de ce canal est de 27 kbit/s.

31. $t_s / (t_s + t_r) = 6/10$. L'envoi d'une trame de 300 bits sur un canal à 50 kbit/s requiert 6 ms. En résolvant $6 / (6 + t_r) = 6/10$, on obtient $t_r = 4$. Le délai de propagation en un sens est de 2 ms.

32. $t_s = 800 / 1200 = 2/3$. $t_r = 16$.

$$\frac{2/3}{16 + 2/3} = \frac{2}{50} = 4\ \%$$

33. Pour l'ancien canal, $2BD + 1 = 3 \rightarrow BD = 1$. $(250d)/1000 = 1 \rightarrow d = 4$, avec d le délai de propagation en un sens. La bande passante est doublée à 500 kbit/s, mais le délai reste de 4 ms. $BD = 500 \times 4 = 2000$ bits, soit 2 trames. $2BD + 1 = 5$ trames. Sur le nouveau canal, l'efficacité est de $3/5 = 60\%$.

34. Cela peut arriver. Supposez que l'émetteur transmette une trame et qu'un acquittement trompeur arrive très rapidement en retour. La boucle principale du programme (du protocole) est exécutée une deuxième fois, une trame est alors à nouveau transmise, bien que le *timer* soit déjà activé.

35. Pour obtenir une efficacité maximale, le numéro de séquence (en fait, la taille de la fenêtre) doit être tel que l'émetteur puisse transmettre jusqu'à ce qu'il reçoive le premier acquittement. Le délai de propagation est de 18 ms. Sur un lien T1, à 1,536 Mbit/s (si on exclut le premier bit d'en-tête), une trame de 64 octets est transmise en 0,3 ms. La première trame est intégralement reçue par le récepteur au bout de 18,3 ms. L'acquittement de cette trame, transmis immédiatement par le récepteur, est reçu par l'émetteur 18 ms plus tard. Il faut ajouter un très court temps (négligeable) permettant à l'acquittement d'être intégralement reçu. Cela représente un temps global de 36,3 ms. L'émetteur dispose ainsi de 36,3 ms pour transmettre des trames avant de recevoir l'acquittement de la première trame transmise. Comme une trame est transmise en 0,3 ms, cela lui permet de transmettre 121 trames. Le numéro de séquence (la taille de la fenêtre) doit être alors de 7 bits (autorisant jusqu'à 128 trames).

36. Supposons que la fenêtre de l'émetteur soit (Sl, Su) et celle du récepteur (Rl, Ru). Considérons que la taille de la fenêtre est W, alors les relations qui doivent être satisfaites sont :

$0 \leq S_u - S_l + 1 \leq W$

$R_u - R_l + 1 = W$

$S_l \leq R_l \leq S_u + 1$

37. OLD25 Oui. Cela conduirait à un interblocage (*deadlock*). Supposez qu'un groupe de trames arrive correctement et qu'elles soient toutes acceptées. Le récepteur avance sa fenêtre d'autant. Maintenant, supposons que tous les acquittements transmis par le récepteur se perdent. L'émetteur verrait, éventuellement, son timer expirer et renverrait alors la première trame du groupe. En réponse à cet envoi, le récepteur transmettrait un NAK. Faisons le point. On se trouve dans une situation où l'émetteur a retransmis une trame qui avait déjà été acceptée mais le récepteur l'ignorait. L'utilisation d'un timer auxiliaire permet de lever l'ambiguïté, en offrant au récepteur la possibilité de transmettre un acquittement correct et de se resynchroniser avec l'émetteur.

38. Cela conduirait à un interblocage car c'est le seul endroit dans le programme où les acquittements reçus sont traités. Sans ces lignes de code, l'émetteur passerait son temps à attendre l'expiration de son timer sans aucune progression possible.

39. Pour une taille de fenêtre d'envoi w, le taux d'utilisation du lien est $w/(1 + 2BD)$. Avoir 100 % d'utilisation nous conduit à une fenêtre de $w = 150001$.

40. Cela ôterait la raison d'être des acquittements négatifs NAK, et obligerait à recourir à des timeouts. Les performances en souffriraient, mais pas la protection contre les erreurs. Les NAK ne sont pas essentiels.

41. Considérons le scénario suivant : *A* envoie la trame 0 à *B*, *B* la reçoit et transmet en retour ACK à *A*, mais cet acquittement se perd. *A* voit le délai de son timer expirer et retransmet la trame 0. *B* s'attend à recevoir la trame 1 ; quand il reçoit la trame 0, il répond par un envoi de NAK. En revanche, si *A* avait retransmis *r.ack* + 1, il aurait envoyé la trame 1 qui n'avait pas encore été transmise.

42. Supposez que *A* transmette une trame à *B* qui soit correctement reçue, mais que *B* n'ait aucune trame à envoyer. À l'expiration du délai de son timer, *A* retransmet la trame précédente. *B* constate qu'une erreur de séquencement s'est produite car le numéro de la trame attendue est inférieur à *trame_attendue*. En conséquence, *B* envoie NAK avec le numéro de trame attendue. Chaque trame est ainsi transmise deux fois.

43. Non. Cette implémentation ne fonctionne pas. Avec *MAX_SEQ* = 4, on a *NB_BUFS* = 2. Une trame paire utilise le buffer 0 et une trame impaire le buffer 1. Cela signifie que les trames 0 et 4 emploient le même buffer. Supposez que les trames 0 à 3 soient transmises, reçues correctement et acquittées. La fenêtre du récepteur contient 4 et 0. Supposez que la trame 4 soit perdue et que la trame 0 arrive. Elle est stockée dans le buffer 0. `Arrivé[0]` passe à `true`. La boucle `Arrivée trame` est exécutée une fois et un message d'erreur de séquencement est transmis à l'émetteur. Le protocole nécessite que *MAX_SEQ* soit impair pour fonctionner correctement. D'autres implémentations de protocoles à fenêtre ne présentent pas cette propriété.

44. Soit $t = 0$ l'instant de début de transmission. À $t = 1$ ms, la première trame a été envoyée. À $t = 271$ ms, elle est intégralement reçue. À $t = 272$ ms, l'acquittement de la première trame a été envoyé. À $t =$ 542 ms, l'acquittement est intégralement reçu. Le cycle de transmission/acquittement d'une trame est donc de 542 ms. Un ensemble de k trames sont transmises en 542 ms, avec une efficacité de $k/542$.

    Ainsi :

    Avec $k = 1$, l'efficacité est de : 1/542 = 0,18 % ;

    Avec $k = 7$, l'efficacité est de : 7/542 = 1,29 % ;

    Avec $k = 4$, l'efficacité est de : 4/542 = 0,74 %.

45. Non. Lorsqu'un acquittement négatif est perdu, l'émetteur croit à tort que sa trame a été bien reçue.

46. Soit t = 0, l'instant de début de transmission. À t = 4 096/64 000 = 64 ms, le dernier bit est transmis. À t = 334 ms, le dernier bit arrive au satellite et un court ACK est transmis. À t = 604 ms, l'ACK parvient à la station terrienne. La vitesse de transmission est de 4 096 bits en 604 ms, ce qui correspond à 6 781 bit/s. Avec une fenêtre d'anticipation de 7, l'émetteur transmet 7 trames en 448 ms puis arrête de transmettre. Après 604 ms, le premier ACK parvient à l'émetteur, un nouveau cycle peut recommencer. Ce sont ainsi 7 × 4 096 = 28 672 bits qui ont été transmis en 604 ms. La vitesse de transmission est alors de 47 470,2 bit/s. La transmission en continu est possible si l'émetteur est en train de transmettre lorsque le premier ACK lui parvient à t = 604 ms. En d'autres termes, si la taille de la fenêtre de transmission est supérieure ou égale à 604 ms, l'émetteur travaille à pleine vitesse et sans arrêt. Avec une fenêtre de 10 trames (ou plus), cette condition est remplie. Pour des tailles de fenêtre de 10 trames ou plus (par exemple 15 ou 127 trames), la vitesse de transmission est de 64 kbit/s.

47. La vitesse de propagation sur le câble est de 200 000 km/s ou 200 km/ms. Sur un câble de 100 km, elle est de 500 µs. Une trame T1 compte 193 bits qui sont transmis en 125 µs. C'est ainsi que 4 trames T1 peuvent se trouver sur un câble de 100 km, ce qui correspond à 772 bits sur ce câble.

48. PPP est un protocole qui a été conçu typiquement pour n'être implémenté que sous forme logicielle, et non sous forme matérielle, comme peut l'être le protocole HDLC. PPP ne travaille qu'avec des octets et non avec des bits. En outre, PPP a été conçu pour être utilisé avec des modems qui ne manipulent des unités de données que sous forme d'octets et non directement de bits.

49. Au minimum, une trame PPP compte 2 octets de fanion, 1 octet de protocole et 2 octets de total de contrôle. Cela représente un surcoût total minimum de 5 octets par trame. Au maximum, on trouvera 2 octets de fanion, 1 octet pour l'adresse et 1 octet de contrôle, 2 octets pour le protocole et 4 octets de total de contrôle. Cela nous conduit à 10 octets de surcoût.

50. Chaque octet est neutralisé (échappé) par ESC puis l'octet est traité en OU exclusif avec 0x20. La séquence d'octets transmise est donc :

    01111101 01011101 01111101 01011110 01111101 01011110 01111101 01011101

51. Une trame AAL5 compte 2 octets de protocole PPP, 100 octets de charge utile PPP, quelques octets de bourrage et 8 octets d'en-queue. Pour que sa taille soit un multiple de 48, il faut 34 octets de bourrage. On a alors une trame AAL5 de 144 octets qui occupera 3 cellules ATM. La première contiendra 2 octets du protocole PP et 46 octets du paquet IP, la deuxième contiendra les 48 octets suivants du paquet IP et la troisième les 6 derniers octets du paquet IP, 34 octets de bourrage et les 8 octets de l'en-queue AAL5.

52. Exercice pratique. Appréciation entre collègues ou par un enseignant.