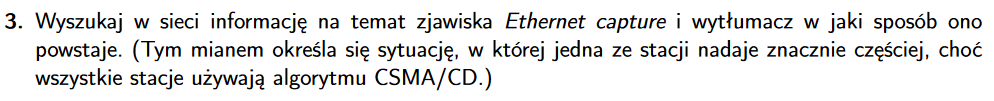
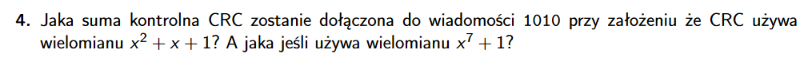
**Sieci Komputerowe – Ćwiczenia 2**



Ethernet capture powstaje wtedy, gdy jeden z użytkowników danego medium raz po raz „wygrywa” użycie medium względem pozostałych użytkowników, czyli w dużym stopniu używa łącza. W protokole Ethernet jeśli dwa hosty wysyłają w tym samym czasie komunikat i powstanie kolizja, obaj wycofują się i czekają losową ilość czasu (zakres losowanego czasu dla obu nich jest taki sam) zanim spróbują po raz kolejny wysłać komunikat. Zjawisko „przechwycenia” (capture) medium występuje, gdy „wygrywać” będzie cały czas jeden host. Po każdych 16 kolizjach następujących po sobie przedział losowania czasu czekania hosta (tego, który przegrał) przed wysłaniem kolejnego pakietu zwiększa się. Przykładowo jeśli przy pierwszej kolizji losowaliśmy czas z przedziału <0,1>, przy wydłużonym czasie będziemy losować czas z przedziału <0,3>.

Załóżmy, że mamy dwa hosty: A oraz B. A oraz B w tym samym czasie wysyłają komunikat. Następuje kolizja, więc oba hosty czekają losową ilość czasu z przedziału <0,1>. Host A wybrał mniejszy czas, więc wysyła komunikat szybciej. Po upływie swojego czasu czekania komunikat próbuje wysłać host B. Jednak A nadaje, więc B wycofuje się. Tak się dzieje 16 razy. Host B zwiększa swój zakres czekania do kolejnej próby wysłania komunikatu. Czyli losuje z większego przedziału. Może powstać sytuacja, gdzie A cały czas będzie wygrywał i przejmie łącze, przez co B będzie cały czas czekał na swoją kolej. Zjawisko jest mniej prawdopodobne przy większej ilości hostów (więcej hostów jest w stanie przejąć łącze podczas oczekiwania hosta, który wcześniej je przejął).



1. **G(x) = x2 + x + 1**

Korzystam z wiadomości z wykładu. Mamy więc:

G(x) = x2 + x + 1

m = 1010 -> x3 + x = M(x)

r = 2 = st(G)

Dzielimy xr \* M(x) = x2 \* M(x) = x5 + x3 przez G(x).   
Otrzymujemy: x2 \* M(x) = (x3 – x2 + x) \* (x2 + x + 1) + x =(mod 2)= (x3 + x2 + x) \* (x2 + x + 1) + x.  
Z teorii z wykładu wynika, że reszta to jest nasze S(x), czyli S(x) = x. Suma kontrolna powinna mieć st(G) = 2bity, więc s = 10(2).

1. **G(x) = x7 + 1**

Ponownie korzystam z wiadomości z wykładu. Czyli:

G(x) = x7 + 1

m = 1010 -> x3 + x = M(x)

r = 7 = st(G)

Dzielę x7 \* M(x) = x10 + x8 przez G(x).

Otrzymujemy: x7 \* M(x) = (x3 + x) \* (x7 + 1) + (x3 + x) =(mod 2)= (x3 + x2 + x) \* (x2 + x + 1) + x.  
Z teorii z wykładu wynika, że reszta to jest nasze S(x), czyli S(x) = (x3 + x). Suma kontrolna powinna mieć st(G) = 7bitów, więc s = 0001010(2).



To co musimy pokazać to identyczne zachowanie sumy kontrolnej CRC-1 z bitem parzystości. Czyli jeśli wiadomość ma parzystą liczbę zapalonych bitów, suma kontrolna ma wynosić 0, w przeciwnym wypadku 1. Konstrukcja obu sposobów jest taka, że „doklejamy” na koniec wiadomości odpowiedni bit (0 lub 1). W takim razie musimy tylko pokazać, że suma kontrolna będzie taka sama w obu przypadkach.

Załóżmy, że mamy wiadomość (m) zapisaną bitowo. Ma ona n elementów (są to elementy ze współczynnikami 1, elementy ze współczynnikami 0 pomijamy, ponieważ nic nie wnoszą).

Wiadomość ta będzie miała wielomian: M(x) = xk1 + xk2 + xk3 + … + xkn.

Używając metody CRC-1 musimy znaleźć s, które dokleimy do wiadomości. Aby znaleźć s, musimy podzielić wielomian M(x) przez G(x) = x + 1. Jednak nie znamy do końca postaci wielomianu, a potrzebujemy znać tylko resztę z dzielenia. Z twierdzenia o reszcie wielomianu mamy, że:

„Reszta z dzielenia wielomianu W(x) przez dwumian x-a jest równa W(a).”

Podstawiając pod każdy x cyfrę 1 (normalnie podstawialibyśmy -1, ale modulo 2 liczby 1 i -1 to to samo) uzyskujemy:

1. **Jeśli n jest parzyste** – parzystą liczbę jedynek. Suma parzystej ilości jedynek jest parzysta. Wykonując na tej sumie operację modulo 2 uzyskujemy 0.

Metoda bitu parzystości da nam taki sam efekt (dodajemy zgaszony bit, bo mamy już parzystą ilość zapalonych bitów w wiadomości).

1. **Jeśli n jest nieparzyste** – nieparzystą liczbę jedynek. Suma nieparzystej ilości jedynek jest nieparzysta. Wykonując na tej sumie operację modulo 2 uzyskujemy 1.

Metoda bitu parzystości da nam taki sam efekt (dodajemy zapalony bit, bo nie mamy parzystej ilości zapalonych bitów w wiadomości).

Czyli metoda CRC-1 oraz metoda bitu parzystości działają identycznie.