Jakub Sobieszek 232838
Adrian Śnieżek 235984
prow. Piotr Semberecki

04.06.2019
Wrocław

Niezawodność i Diagnostyka Układów Cyfrowych 2

Projekt

1.Cele Projektu
Tematem Projektu była symulacja i implementacja systemu ARQ – Automatic Repeat Request
Zadaniem Aplikacji było wczytanie zdjęcia z pliku, podzielenie go na paczki bitów, następnie symulowanie przesyłania paczek do odbiorcy poprzez funkcje wprowadzające błędy na podstawie różnych modeli błędów. Przesyłanie odbywa się poprzez jeden z systemów detekcji i korekcji błędów, który na podstawie sumy kontrolnej decyduje czy przyjąć paczkę, czy zażądać jej ponownego przesłania. Ostatecznie, przyjęte paczki scalane były z powrotem do formy obrazu. W zależności od wybranych modeli błędów, protokołów detekcji i korekcji, sumy kontrolnej, oraz innych parametrów programu można było obserwować stopień zniekształcenia obrazu wyjściowego. W programie zastosowano:

Modele błędów:
- -Binary Symmetric Channel
- -Gilbert's model

Protokoły detekcji i korekcji:
- -Stop and Wait
- -Go back n
- -Selective Repeat

Sumy kontrolne:
- -parity bit
- -crc32

dodatkowymi parametrami programu były:
- -prawdopodobieństwo wystąpienia błedu
- -ilość możliwych ponownych przesłań, po wyczerpaniu których odbiorca zapisywał sztywno ustaloną paczkę (np. paczkę składającą się z samych zer)
- -wielkość ramki wg której przesyłane będą ramki (dotyczy protokołów Selective Repeat oraz Go back n)

Binary Symmetric Channel(BSC) – Prosty model błędu, zakładający że przy przesyłaniu każdej informacji istnieje stałe prawdopodobieństwo zniekształcenia danych.
Przykład: prawdopodobieństwo błędu = 0.05
Przesyłana paczka danych w 5/100 przypadków zostanie zniekształcona.
W/w prawdopodobieństwo tyczy się każdego przesyłanego bitu danych, bez względu na wielkość paczki. Można więc założyć że 5/100 bitów zostanie przekłamanych w trakcie transmisji.

Model Gilberta – Model, zakładający kumulowanie się błędów. Prawdopodobieństwo wystąpienia błędu staje się prawdopodobieństwiem zmiany stanu (z poprawnego na błędny i vice-versa). Dla niskiej wartości prawdopodobieństwa istnieje mała szansa wystąpienia błędu. Jeśli natomiast pojawi się on, możemy oczekiwać wielu błędów, zanim wrócimy do stanu poprawnego

Stop and Wait(SAW) – prosty protokół operujący na jednej paczce jednocześnie. Przeznaczona do wysłania paczka jest transmitowana, protokół czeka z wysłaniem kolejnej, aż otrzyma potwierdzenie, że wysłana paczka dotarła w poprawnej postaci do odbiorcy.

Go Back N(GBN) – Efektywniejszy od SAW protokół operujący na kilku paczkach jednocześnie. Zakładany jest rozmiar ramki określający ilość paczek wysyłanych jednocześnie. Paczki w tej ilości są następnie wysyłane jedna po drugiej, bez czekania na informację zwrotną od odbiorcy. Po otrzymaniu informacji o poprawnie przesłanej paczce, ramka „przesuwa się" o jedno miejsce. Przykładowo jeśli rozmiar ramki=5; wysyłane są paczki *n,...,n+4*. Kiedy nadawca otrzyma potwierdzenie poprawnego przesłania paczki *n* – ramka przesuwa się by obejmować paczki *n+1,...,n+5*. Kiedy dla danej paczki otrzymana zostanie informacja o niepoprawnej transmisji – ponawiane jest wysyłanie wszystkich paczek znajdujących się w obecnej ramce. Może to spowodować ponowne przesyłanie paczek, które zostały już poprawnie przesłane.

Selective Repeat(SR) – Wydajniejsza wersja protokołu Go Back N. Tak samo jak w/w wysyła kilka paczek jednocześnie w zależności od rozmiaru ramki. Po wysłaniu wszystkich paczek, czeka na informacje zwrotne. Po ich otrzymaniu: jeśli paczka została przesłana poprawnie – jest usuwana z paczki a na jej miejsce dodawana jest następna w kolejności. Jeśli wystąpił błąd – paczka zostaje w ramce i następuje ponowna próba jej przesłania. Eliminuje to problem GBN gdzie bez potrzeby może wystąpić retransmisja poprawnie przesłanej paczki.

Parity bit – suma kontrolna polegająca na sprawdzeniu ilości bitów „1" w paczce. Jeśli paczka zawiera parzystą ilość bitów „1", bit parzystości=1, w przeciwnym wypadku 0. Jest to prosta metoda kontroli poprawności. Jak łatwo można zauważyć, w przypadki kiedy przekłamana zostanie parzysta ilość bitów w paczce – w trakcie testu nie zostanie wychwycony błąd transmisji, mimo iż nastąpiło przekłamanie.

Suma kontrolna CRC32 - bardziej zaawansowana metoda kontroli poprawności. Polega na sprawdzeniu, czy otrzymana paczka generuje taką samą resztę z dzielenia, przez pewien ustalony wielomian, jak paczka wyjściowa. Wykorzystana suma kontrolna działa na stringach. Bity każdej ramki zostają zapisywane jako napis, następnie przekazywane dalej, do obliczenia reszty. Obliczenia wykonują się poprzez dopisanie zera a ustalony dzielnik jest stały i jest równy 1011. Sprawdzenie polega na zadaniu wyliczonej reszty i porównanie z pierwotnym wielomianem.

2.Realizacja

Paczka zrealizowana została jako klasa, z atrybutami (m.in.):

-macierzy jednowymiarowej zawierającej w sobie ciąg bitów (dane obrazu)

-sumy kontrolnej

oprócz w/w używane były atrybuty sytuacyjne potrzebne w zależności od używanego protokołu.

Po uruchomieniu programu użytkownik jest proszony o wybór parametrów programu: prawdopodobieństwa błędu, sumy kontrolnej, modelu błędów etc. Następnie zaczytane zdjęcie zostaje przesłane wedle wybranych parametrów. Po zakończeniu transmisji generowany jest log operacji ze statystykami, oraz zapisany oraz wyświetlony zostaje obraz wyjściowy.

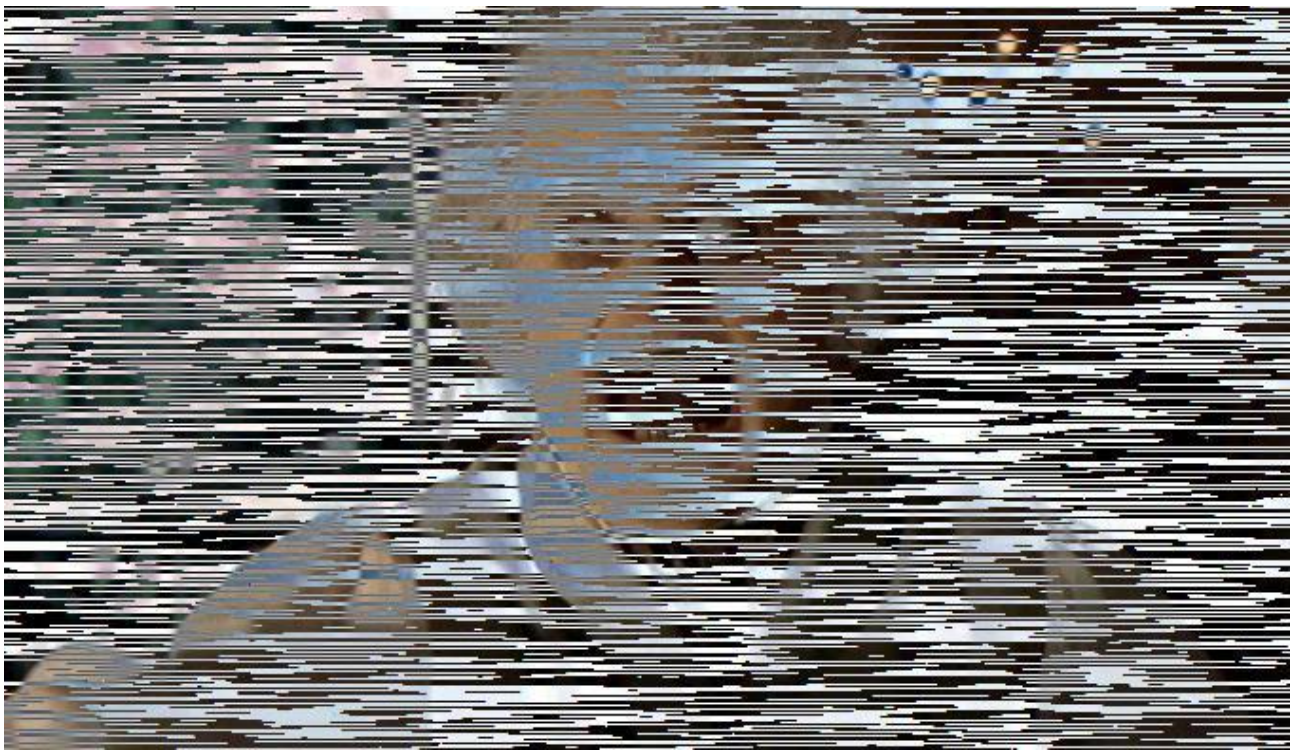
3.Przykładowe wyniki

Parametry:
Prawdopodobieństwo błędu: 0.02
Ilość możliwych ponownych przesłań: 5
Protokół Stop and Wait
Model błędów Gilberta
Suma kontrolna crc32
Poprawnie przesłane paczki: 79.99%
Wykryte i naprawione błędy: 83.33%
Niewykryte błędy: 16.77%
Czas 109.09 s

Model błędów Gilberta zakłada kumulowanie się błędów. Widać dużą ilosc błędów powstałych w trakcie transmisji obrazu (~20% zniekształcenia). Istotne jest też, że błędy występujące wg modelu Gilberta są poważniejsze, niż w przypadku BSC. Jest to spowodowane tym, że kiedy symulujemy błędy modelem Gilberta i wystąpi pierwszy błąd, możemy się spodziewać po nim długiego ciągu kolejnych błędów. Efektem tego są błędy, które nawet jeśli występowały by tak często jak przy BSC, to są o wiele poważniejsze.

Parametry:
prawdopodobieństwo błędu: 0.02
ilość możliwych ponownych przesłań: 5
Protokół Stop and Wait
Model błędów BSC
suma kontrolna crc32
Poprawnie przesłane paczki: 97.1%
Wykryte i naprawione błędy: 83.33%
Niewykryte błędy: 1.83%
Czas: 67.73 s

Obraz przesłany za pomocą symetrycznego modelu błędów. Błędy przypominają równomiernie rozłożony „szum”, w przeciwieństwie do poprzedniego zdjęcia gdzie układały się one w skupiska. O wiele mniejszy jest też stopień zniekształcenia (~3% paczek przesłanych niepoprawnie). Same błędy, oprócz tego że występują rzadziej, są też mniej widoczne. Powodem tego jest mniejsza ilość przekłamanych bitów w paczce



Parametry:
Prawdopodobieństwo błędu 0.0005
brak możliwości ponownego przesłania – zerowa korekcja błędu
protokół Stop and Wait
model błędów Gilberta
suma kontrolna Crc32
Poprawnie przesłane paczki: 54.73%
Wykryte i naprawione błędy: 0%
Niewykryte błędy: 0.12%
Czas: 69.89 s

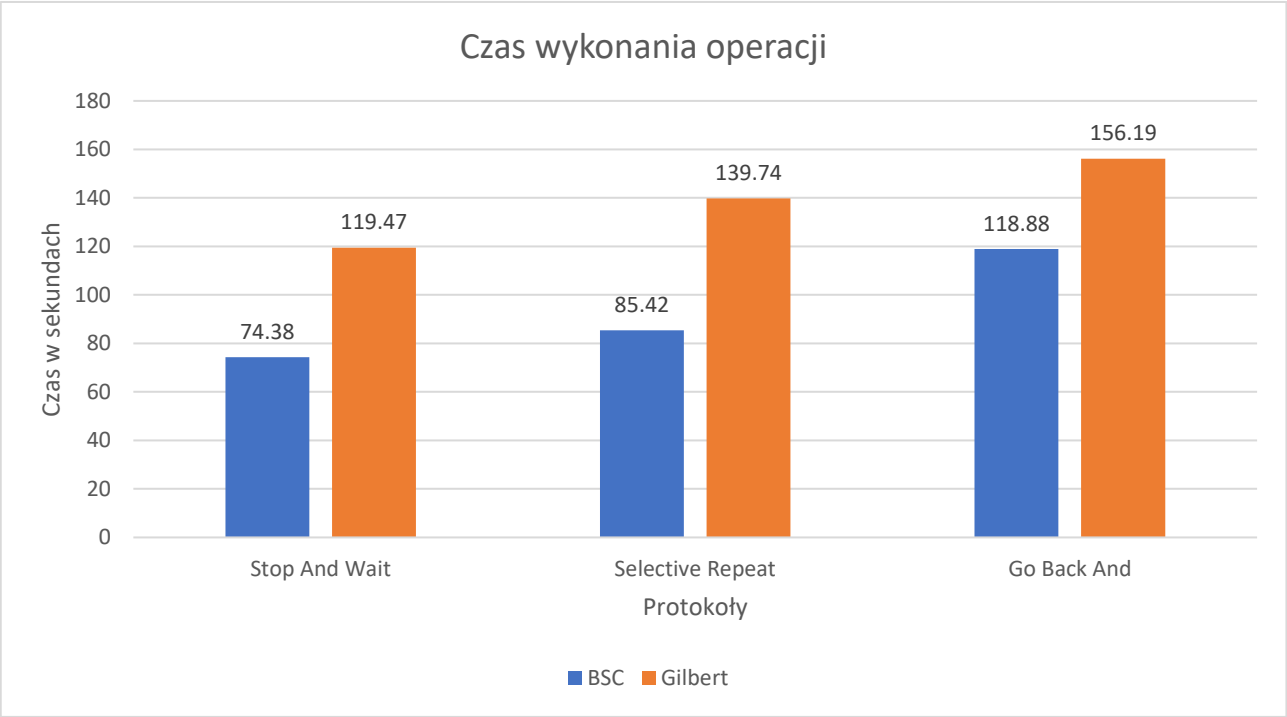
Obraz przesłany za pomocą modelu Gilberta z niewielkim prawdopodobieństwem wystąpienia błędu. Skututkuje to zniekształceniami, które pojawiają się rzadko, ale w dużych skupiskach. Dzięki parametrom nie pozwalającym na korekcję błędów można zaobserwować to w badzo wyraźnie. Zdjęcie wygląda, jak gdyby zostało „przecięte" fragmentami będącymi jego negatywem. Wynika to z dużych skupisk niepoprawnie przesłanych ramek gdzie w danej ramce został przekłamany każdy bit.

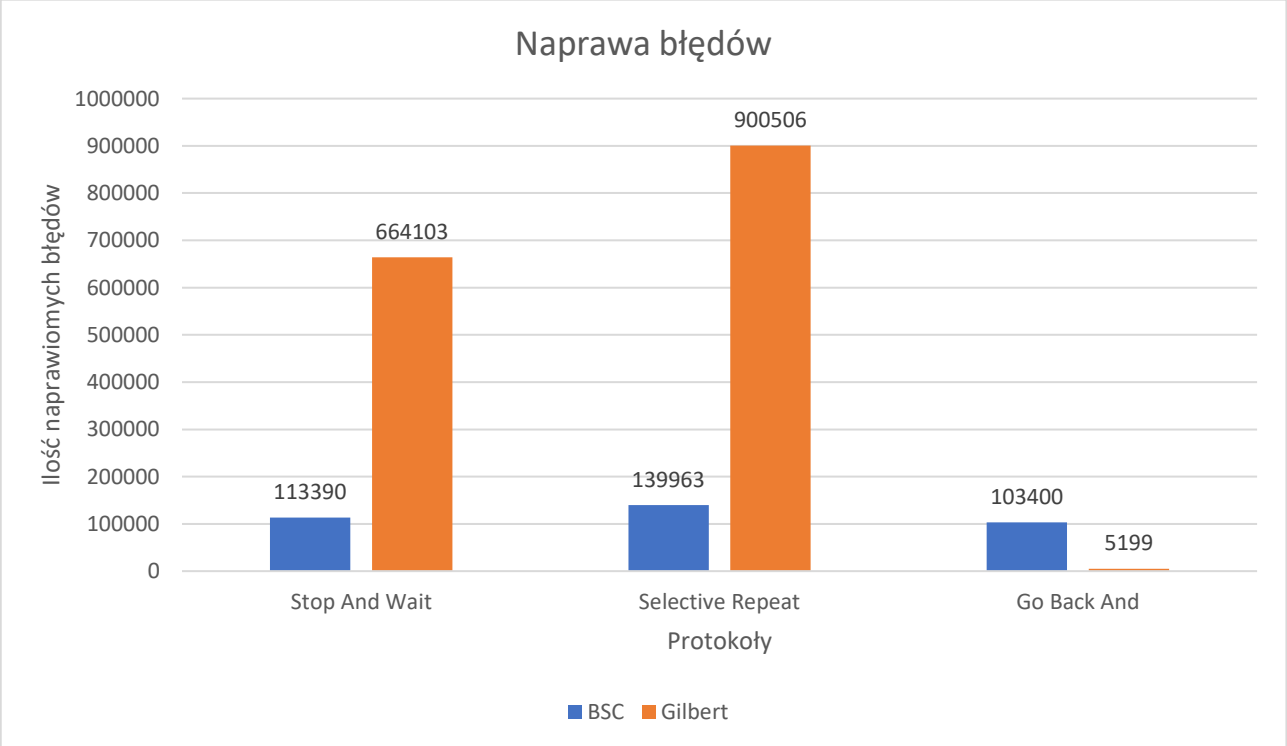
4. Czas i błędy

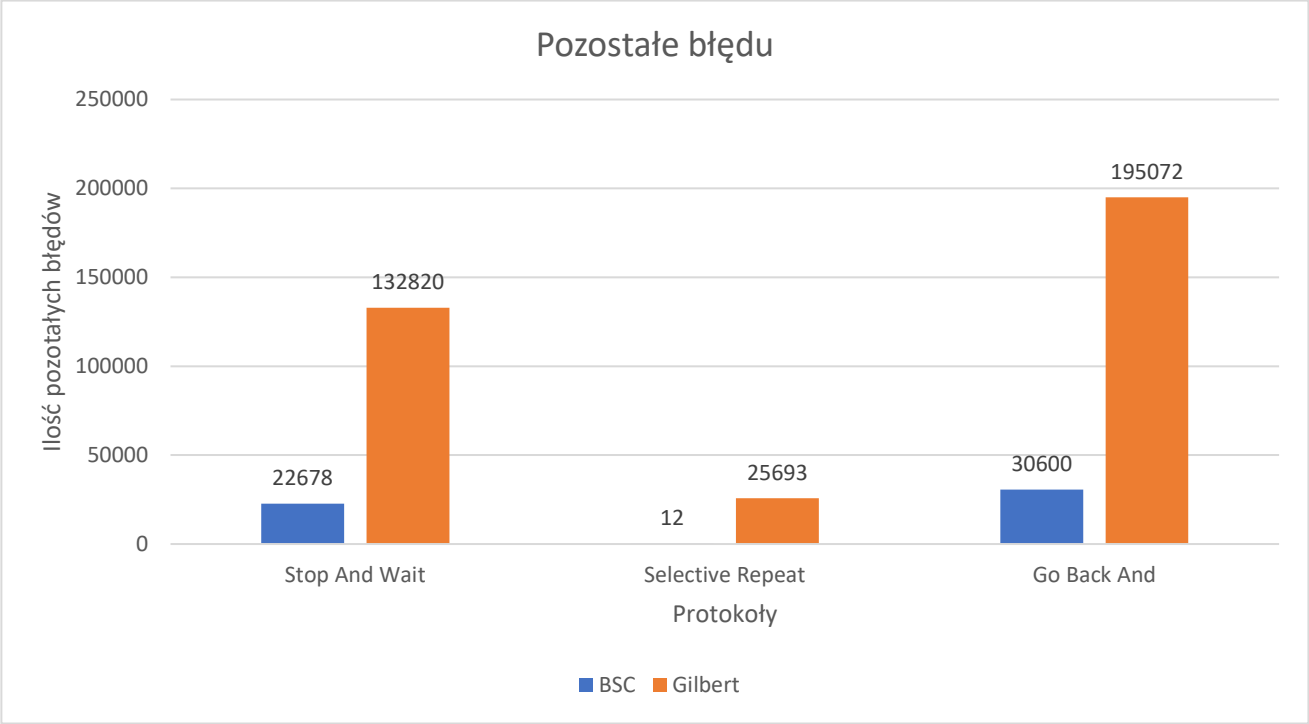
Na podstawie serii testów została wyznaczona średnia z zebranych wyników
(czasu, ilości ramek poprawionych oraz nie poprawionych) podsumowując jakość i prędkość działania poszczególnych protokołów.

Parametry zastosowane do testów:
- prawdopodobieństwo błędu 2%
- 5 możliwych ponownych przesłań
- paczka zawierająca 10 ramek

Zestawienie wyników na wykresach







Na podstawie powyższych wykresów można stwierdzić, że najkorzystniejszym protokołem jest Selective Repeat. Osiąga on najlepsze wyniki w poprawianiu błędów mimo, że nie jest najszybszy. W kwestii prędkości wygrywa Stop and Wait, który naprawia błędy podobnie do Selective Repeat'a porównując przypadek BSC. Patrząc na błędy powstałe przez model Gilberta, Selective Repeat radzi sobie o wiele lepiej niż pozostałe protokoły. Najmniej korzystnym protokołem okazuje się Go Back N co nie jest wielkim zaskoczeniem. Protokół ten wysyła ramki paczkami i jeżeli przekroczy limit możliwych ponownych przesyłów, cała paczka jest kodowana jako źle wysłana. Zmniejszenie wartości ilości ramek w jednej paczce powoduje wzrost wydajności protokołu, ale wciąż nie zbliża się do jakości i potencjału Selective Repeat.

Błędy niekorygowalne czyli takie które nie są wykrywane przez CRC i bit parzystości występują i rozpoczynają się na poziomie prawdopodobieństwa 10^{-3}. Przetestowane zostały różne konfiguracje na wszystkich protokołach – ważne jest, że w każdym z tych testów nie było możliwości przesłania ramki ponownie. Na tym poziomie prawdopodobieństwa występują pojedyncze błędy tj. najczęściej od jednego do zaledwie kilku.

Stopień 5% błędów nienaprawialnych jest osiągany dla różnych prawdopodobieństw w stosunku do wybranego protokołu. Stałymi parametrami były:
- CRC32 / bit parzystości
- liczba możliwych ponownych przesłań = 3
- paczka zawierająca 5 ramek

Dla BSC protokoły osiągają 5% błędów nienaprawialnych przy prawdopodobieństwie :
- Stop and Wait = 2,5%
- Selective Repeat = 4%
- Go back N = 2%

Dla modelu Gilberta protokoły osiągają 5% błędów nienaprawialnych przy prawdopodobieństwie :
- Stop and Wait = 2%

- Selective Repeat = 3%
- Go back N = 1%

Oczywiście stopa wystąpienia błędów nienaprawionych jest uzależniona najbardziej od ilości możliwych przesłań ramki. Porównywając jednak dla takich samych parametrów można stwierdzić, że przy BSC potrzeba większej szansy na błąd aby osiągnąć próg 5% błędów niż w modelu Gilberta.

5. Porównania parametrów

P. błędu: 0.001
Możliwe ponowne przesłania: 10
Protokół Selective Repeat
Model błędów Gilberta
Suma kontrolna Crc32

Rozmiar ramki	Przesłane paczki	Poprawne paczki	Niepoprawne paczki	Wykryte błędy	Naprawione błędy	Nie naprawione błędy	Nie wykryte błędy
5	797328	746420	50908	676312	626835	49477	1431
8		752138	45190	681916	638171	43745	1445
10		756091	41237	685702	645916	39786	1451
16		765029	32299	694234	663404	30830	1469
32		781122	16206	710953	696271	14682	1524
50		789544	7784	720045	713778	6267	1517
100		794529	2799	724657	723424	1233	1566
200		795429	1899	725454	725181	273	1626

Przy protokole Selective Repeat, wraz ze zwiększaniem rozmiaru ramki, zmniejsza się ilość błędnie zapisanych paczek, oraz wykrytych i nie naprawionych błędów. Ciekawe jest że ilość nie wykrytych błędów zdaje się rosnąć razem ze zwiększaniem rozmiaru ramki.

P. błędu: 0.0001
Możliwe ponowne przesłania: 1
Protokół Selective Repeat
Model błędów Gilberta
Suma kontrolna Crc32

Rozmiar ramki	Przesłane paczki	Poprawne paczki	Niepoprawne paczki	Wykryte błędy	Naprawione błędy	Nie naprawione błędy	Nie wykryte błędy
5	797328	641645	155683	312016	156457	155559	124
8		641871	155457	312016	156685	155331	126
10		641961	155367	312196	156953	155243	124
16		641964	155364	313036	157803	155233	131
32		643104	154224	313864	159767	154097	127
50		643776	153552	315157	161761	153396	156
100		646497	150831	318586	167967	150619	212
200		652641	144687	323041	178658	144383	304

Po zmniejszeniu prawdopodobieństwa błędu, oraz ilości możliwych ponownych przesłań 10krotnie, obserwujemy podobną tendencję do zmniejszania się błędów razem ze zwiększającym się rozmiarem ramki. Spadek ten jest jednak o wiele mniejszy.

P. błędu: 0.01
Możliwe ponowne przesłania: 1
Protokół Go Back n
Model błędów BSC
Suma kontrolna Bit parzystości

Rozmiar ramki	Przesłane paczki	Poprawne paczki	Niepoprawne paczki	Wykryte błędy	Naprawione błędy	Nie naprawione błędy	Nie wykryte błędy
5	797328	723438	73890	116297	44507	71790	2100
8		659009	138319	173097	36673	136424	1895
10		618730	178598	209466	32666	176800	1798
16		516804	280524	303634	24594	279040	1484
32		352623	444705	458731	15051	443680	1025
50		264556	532772	542834	10834	532000	772
100		157856	639472	645396	6396	639000	472
200		86707	710621	713751	3551	710200	421

Przy użyciu protokołu Go Back N, wraz ze zwiększaniem rozmiaru ramki, rośnie ilość błędów. Wynika to z faktu iż jeśli pierwsza paczka w ramce osiągnie swój limit przesłań, wszystkie paczki w ramce zostają transmitowane jako niepoprawne. Długość ramki, wpływa więc na to, jak duży efekt ma każdy błąd.

6. Wnioski

Najefektywniejszym protokołem korekcji ARQ okazuje się być Selective Repeat. Oferuje on największą wydajność pod kątem ilości skorygowanych błędów. Aby zwiększyć jego skuteczność należy użyć sumy Crc32, gdyż bit parzystości jest niezwykle podatny na błędy niewykrywalne. Zwiększenie rozmiaru ramki, oraz ilości możliwych ponownych przesłań pozwalają jeszcze podnieść wydajność tego protokołu, szczególnie jeśli rozpatrujemy model błędów Gilberta. Stop And Wait oferuje dobry stosunek wydajności/szybkości do skuteczności, jeśli używany jest do korekcji błędów modelu BSC. Zwiększanie ilości możliwych ponownych przesłań odgrywa kluczową rolę w zwiększeniu jego osiągów. Najgorzej sprawuje się Go Back N który, co prawda operuje na wielu paczkach jednocześnie, ale dokonuje to w sposób, który sprawia, że ustępuje pod każdym względem SAW oraz SR. Wskazane jest używanie do kontroli sumy CRC o jak najdłuższym wielomianie, pozwala to zminimalizować ilość niewykrywalnych błędów. Bit parzystośći jest niezwykle podatny na tworzenie sie niewykrywalnych błędów.