1. Линейные коды (кодирование)

1.1 **NRZ**:

0 » 0, 1 » 1

Space: 0 » переход, 1 » сохранение

Mark: 0 » сохранение, 1 » переход

1.2 **RZ** – с возвращением к нулю в середине такта

0 » 0, 1 » 1

IRZ (inverted) 0 » 1, 1 » 0

1.3 **Манчестерский**

0 » спад в середине, 1 » фронт в середине

Inverted Манчестерский

0 » фронт в середине, 1 » спад в середине

1.4 **Miller**

0 » сохранение в середине, 1 » переход в середине

!!! между 00 обязательный переход между тактами

1.5 **Split Phase**

Space: 0 » инверсия последнего перехода в середине, 1 » повтор последнего перехода в середине

Mark: 0 » повтор последнего перехода в середине, 1 » инверсия последнего перехода в середине

1.6 **Biphase –** всегда смена уровня между тактами

Space: 0 » переход в середине, 1 » сохранение в середине

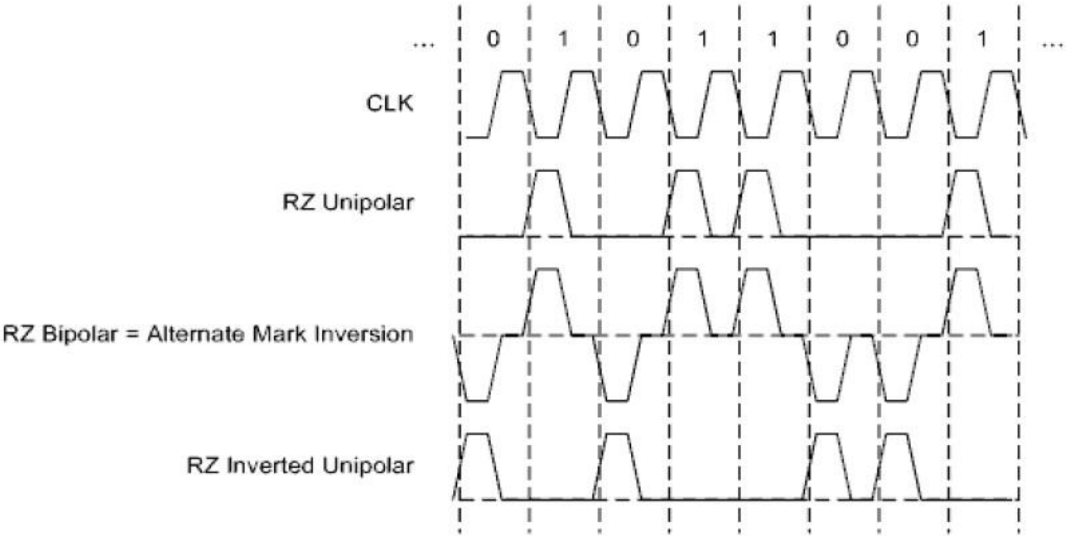
Mark: 0 » сохранение в середине, 1 » переход в середине

1.7 **MLT (например 3) –** три уровня -1, 0, 1

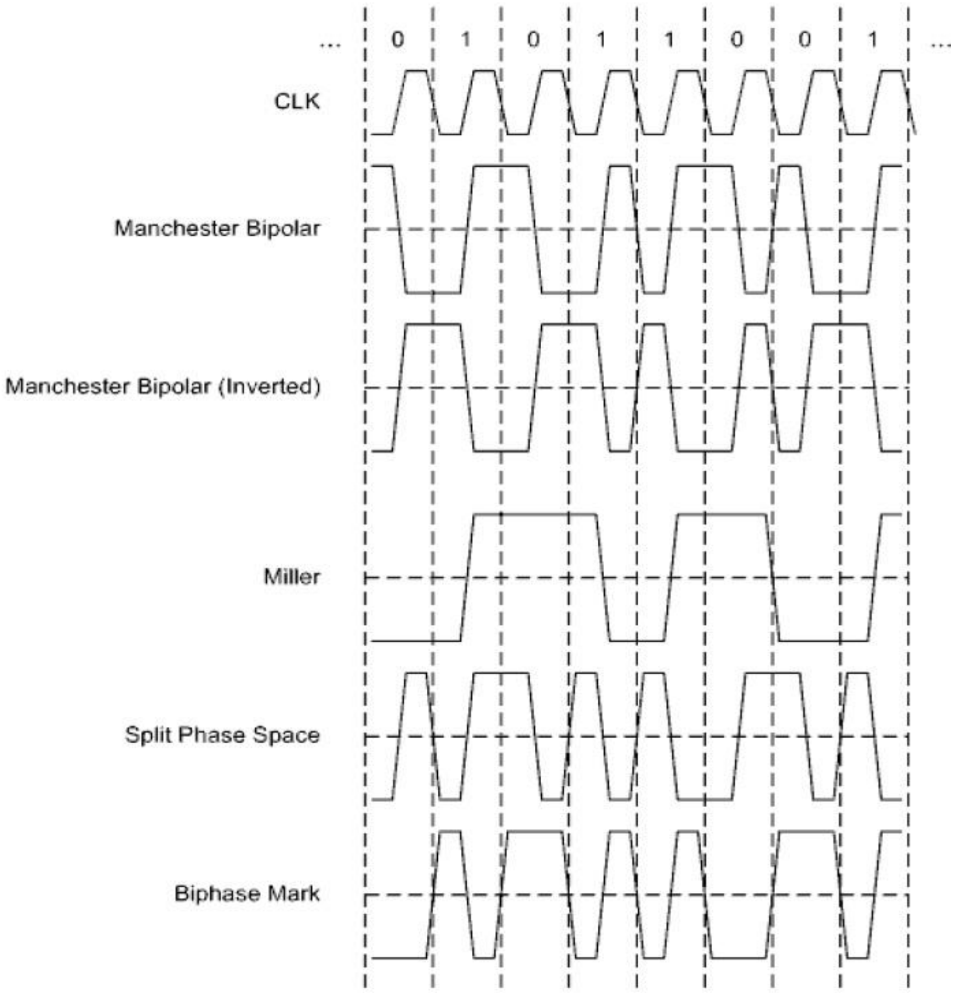
0 » сохранение, 1 » переход с сохранением направления, если граница, то в другую сторону

1.8 **Блочные коды**

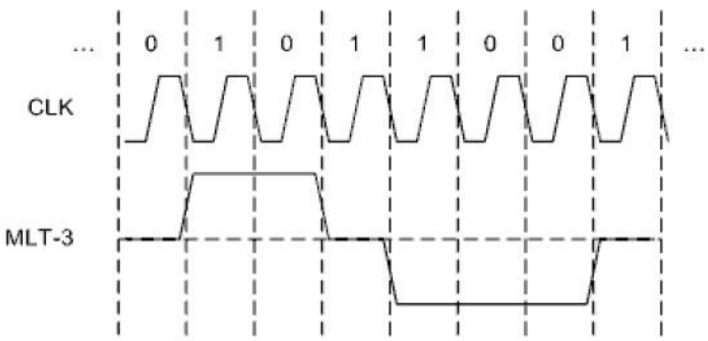
Разбиваем байт на 5 младших и 3 старших. Первоначальный RD = -1. Преобразуем 5 и 3 согласно таблице и RD. Из новой последовательности считаем количество единиц и вычитаем количество нулей. Полученное число прибавляем к RD. Повторяем то же для второго байта с учетом нового RD.



... 0 1 0 1 1 0 0 1 ...
CLK
RZ Unipolar
RZ Bipolar = Alternate Mark Inversion
RZ Inverted Unipolar



... 0 1 0 1 1 0 0 1 ...
CLK
Manchester Bipolar
Manchester Bipolar (Inverted)
Miller
Split Phase Space
Biphase Mark



... 0 1 0 1 1 0 0 1 ...
CLK
MLT-3

2. Помехоустойчивые коды (расчеты)

Кодовое расстояние рассчитывается как минимально кол-во единиц в сумме всех чисел по модулю два.

Например, для кода:

C = {0000, 1001} кодовое расстояние равно 1001 $\oplus$ 0000 = 1001, 2 единицы, т.е. кодовое расстояние = 2

C = {0001, 1011, 1000, 0111} кодовое расстояние равно $d_c = \min p\,(b_x \oplus b_y)$ = min (0001 $\oplus$ 1011 = 1010, 0001 $\oplus$ 1000 = 1001, 0001 $\oplus$ 0111 = 0110, 1011 $\oplus$ 1000 = 0011, 1011 $\oplus$ 0111 = 1100, 1000 $\oplus$ 0111 = 1111)

Минимально – 2 единицы, т.е. кодовое расстояние = 2

Код может обнаружить $t <= d_{\min}$ -1 ошибок т.е. тут 1
Код может исправить $t <= (d_{\min}$ -1)/2 ошибок т.е. тут 0

3. Код Хэмминга и циклический код (кодирование)

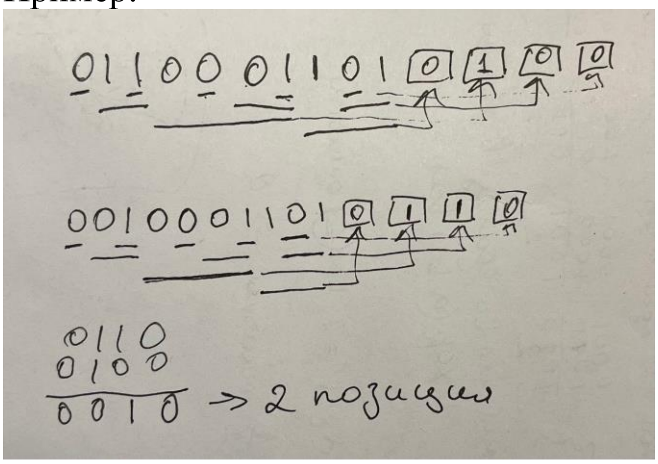
3.1 Код Хэмминга

Для определения числа контрольных символов применяется следующая формула:

$$k = \lceil \log_2(m+1) \rceil,$$

где m – число символов данных.

Первая проверка: охватывает те разряды числа, номера которых имеют в первом (младшем) разряде в двоичке единицу (1, 3, 5, 7 и т.д.) (1_2– 1, 11, 101, 111, 1001), считаем сумму единиц в этих разрядах и записываем в младший контрольный бит n_1
Вторая проверка: охватывает те разряды, которые во втором разряде имеют единицу (2, 3, 6, 7, 10, 11 и тд) ($1x_2$ – 10, 11, 110, 111, 1010, 1011 и т.д.) , считаем сумму единиц в этих разрядах и записываем в контрольный бит n_2
Третья проверка: охватывает те разряды, которые в третьем разряде имеют единицу (4, 5, 6, 7, 12, 13, 14, 15)($1xx_2$ – 100, 101, 110, 111, 1100, 1101, 1110), считаем сумму единиц в этих разрядах и записываем в контрольный бит n_3
И тд, получаем контрольное значение $n_k\, n_{k-1} \ldots n_2\, n_1$
Меняем исходное число на один бит, вычисляем контрольное значение, выполняем XOR с изначальным верным контрольным значением, получаем позицию ошибки.
Пример:



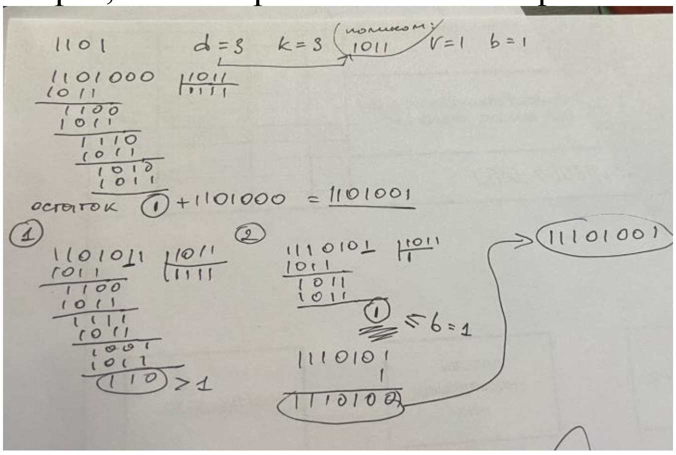
3.2 **Циклический код – CRC**

1. Разбить сообщение в полиномиальную форму: 1101 = x3 + x2 + 1 *(не знаю зачем но допустим)*
2. Определить значение кодового расстояния:
   $d_{\min}$ >= r + 1, $d_{\min}$>=2b + 1, где r – количество обнаруживаемых ошибок, b – количество исправляемых ошибок (даются по условию).
   $d_{\min}$ = 3 для обнаружения и исправления 1 ошибки.
3. Определить число контрольных разрядов $k = \lceil \log_2(m+1) \rceil$, тут $k = 3$.
4. Добавить к исходному числу справа k нулей. 1101<u>000</u>
5. Выбрать <u>простой</u> полином степени $d_{\min}$.. Например, x3 + x + 1 = 1011.

№ п/п	Степень	Многочлен	Двоичная последовательность
1	1	$x+1$	11
2	2	x^2+x+1	111
3	3	x^3+x+1	1011
4		x^3+x^2+1	1101
5		x^4+x+1	10011
6	4	x^4+x^3+1	11001
7		$x^4+x^3+x^2+x+1$	11111
8		x^5+x^2+1	100101
9		x^5+x^3+1	101001
10	5	$x^5+x^3+x^2+x+1$	101111
11		$x^5+x^4+x^2+x+1$	110111
12		$x^5+x^4+x^3+x+1$	111011
13		$x^5+x^4+x^3+x^2+1$	111101
14		x^6+x+1	1000011
15		x^6+x^3+1	1001001
16		$x^6+x^4+x^2+x+1$	1010111
17		$x^6+x^4+x^3+x+1$	1011011
18	6	x^6+x^5+1	1100001
19		$x^6+x^5+x^2+x+1$	1100111
20		$x^6+x^5+x^3+x^2+1$	1101101
21		$x^6+x^5+x^4+x+1$	1110011
22		$x^6+x^5+x^4+x^2+1$	1110101
23		x^7+x+1	10000011
24		x^7+x^3+1	10001001
25		$x^7+x^3+x^2+x+1$	10001111
26		x^7+x^4+1	10010001
27		$x^7+x^4+x^3+x^2+1$	10011101
28		$x^7+x^5+x^2+x+1$	10100111
29		$x^7+x^5+x^3+x+1$	10101011
30		$x^7+x^5+x^4+x^3+1$	10111001
31	7	$x^7+x^5+x^4+x^3+x^2+x+1$	10111111
32		x^7+x^6+1	11000001
33		$x^7+x^6+x^3+x+1$	11001011
34		$x^7+x^6+x^4+x+1$	11010011
35		$x^7+x^6+x^4+x^2+1$	11010101
36		$x^7+x^6+x^5+x^2+1$	11100101
37		$x^7+x^6+x^5+x^3+x^2+x+1$	11101111

6. Делим число на простой полином, получаем остаток, складываем это число с остатком. 1101<u>000</u> / 1011 => остаток 1. Получаем число 1101001. Это число мы как бы и «отправляем».
7. Вводим в число ошибку *(лучше ближе к концу, так быстрее),* например 11010<u>1</u>1.
8. Делим на простой полином. Получаем остаток, если он > числа исправляемых ошибок(b), сдвигаем число вправо циклически, делим снова. Когда остаток <u>меньше либо равен</u> количеству исправляемых ошибок, выполняем XOR

полученного числа и остатка, а после этого циклически сдвигаем влево столько же раз, что и вправо. Mission complited.



4. Поля Галуа (математические операции)

Сложение (вычитание) происходит с помощью XOR для <u>бинарных</u> полей. Для *небинарных (хехе)* это просто деление с остатком на основание.

Операция сложения

Самой простой является операция сложения, которая является простым побитовым сложение по модулю 2 (XOR).

Пример: 5+3=110=6

$$\oplus \begin{array}{r} 101 \\ \underline{011} \\ 110 \end{array}$$

Умножение происходит по следующему алгоритму: два полинома перемножаются по правилам математики (даже в полиномиальной форме осуществляется сложение по модулю 2, поэтому x^2+x^2=0). После чего, если полученный полином выходит за пределы поля (то есть его степень больше степени порождающего полинома), то данный полином делится на порождающий полином и результатом умножения считается <u>остаток</u>.

Вернемся к примеру с умножением:

$$5 \cdot 7 = x^4 + x^3 + x + 1 = \left| \begin{array}{c} \text{Добавим некоторые слагаемые} \\ \text{но так, чтобы ничего не изменилось} \\ \text{(еще раз напомню, что под сложением} \\ \text{понимаю сложение по модулю 2)} \end{array} \right| =$$

$$(x^4 + x^2 + x) + (x^3 + x + 1) + x^2 + x = x \cdot (x^3 + x + 1) + (x^3 + x + 1) + x^2 + x =$$

$$= \left| \begin{array}{c} \text{Так как } x^3 + x + 1 = 0, \text{то} \\ \text{полученное выражение} \\ \text{можно упростить} \end{array} \right| = x^2 + x = 110 = 6$$

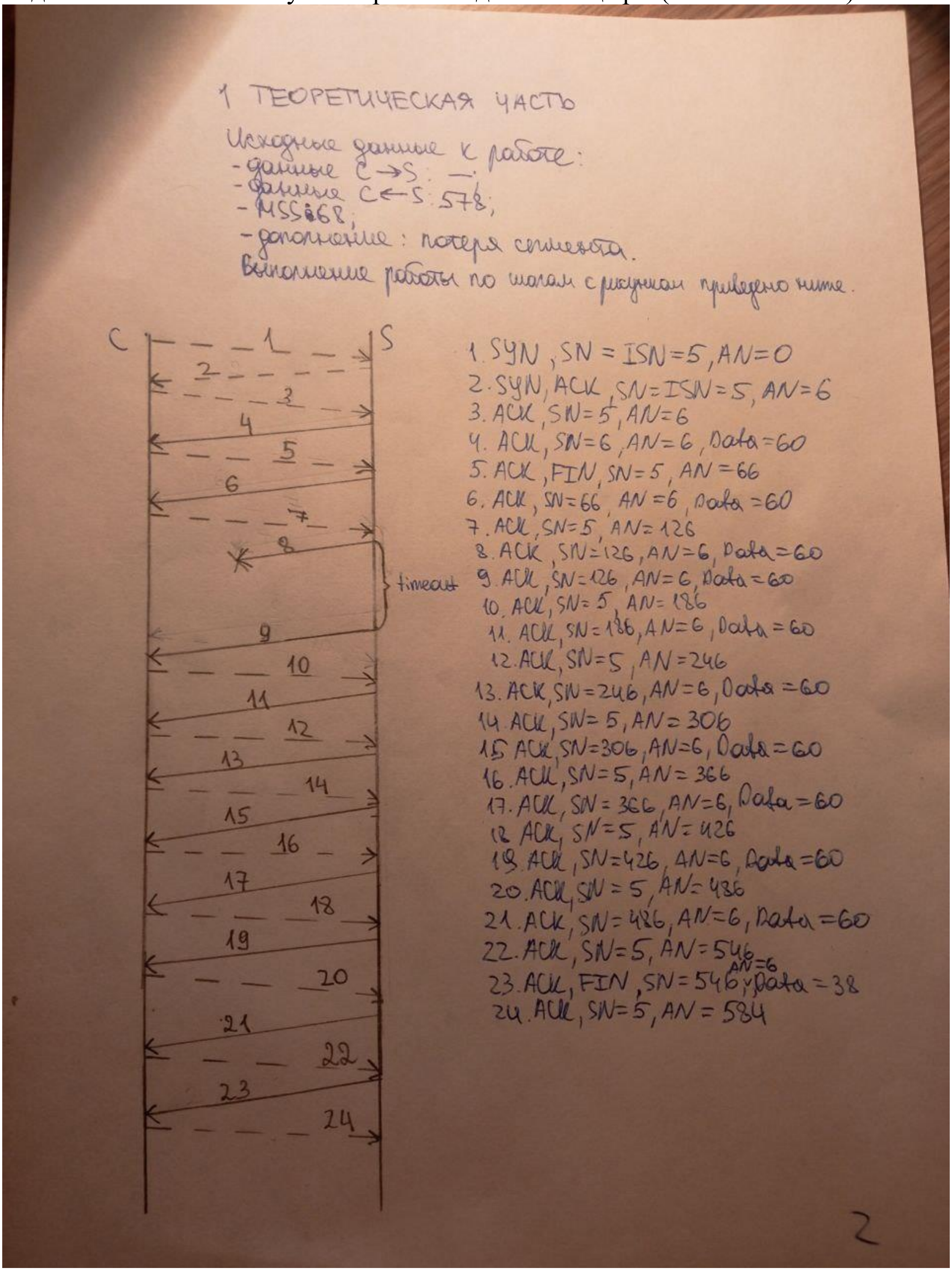
Такой же результат можно получить как остаток от деления полинома, полученного при умножении на порождающий полином:

$$\begin{array}{rl|l} + & x^4 + x^3 + x + 1 & x^3 + x + 1 \\ & \underline{x^4 + x^2 + x} & x+1 \\ + & x^3 + x^2 + 1 & \\ & \underline{x^3 + x + 1} & \\ & x^2 + x & \text{- Остаток от деления} \end{array}$$

5. TCP (диаграммы взаимодействия с детализацией до SYN, ACK, FIN, SN, AN, W и Data)

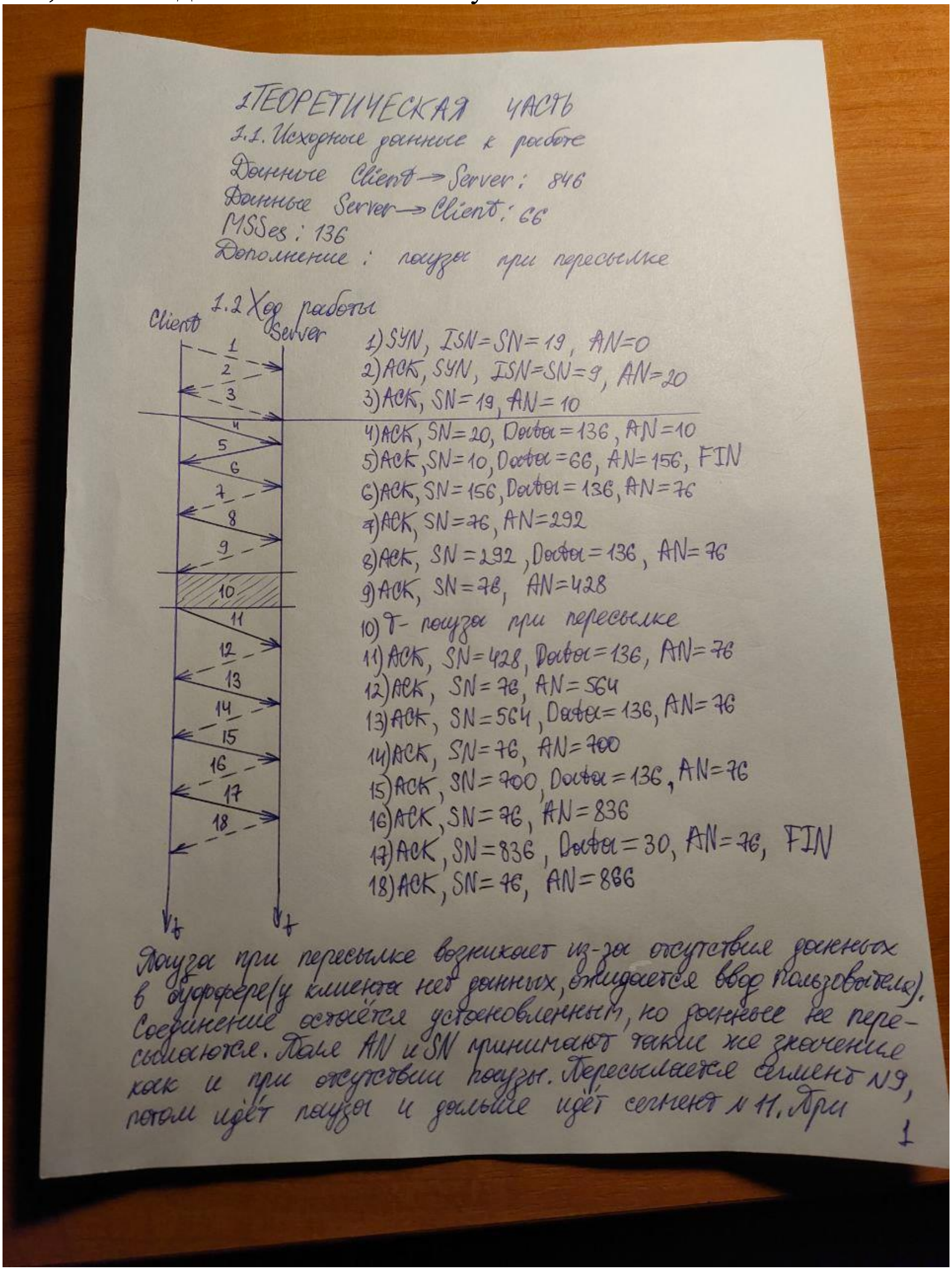
5.1 Потеря Сегмента

Суть: в какой то момент теряется отправляемый сегмент(крестик на отправлении). Выжидаем таймаут(обозначить на схеме фигурной скобкой и подписать. После таймаута отправляем данные еще раз(тот же сегмент)



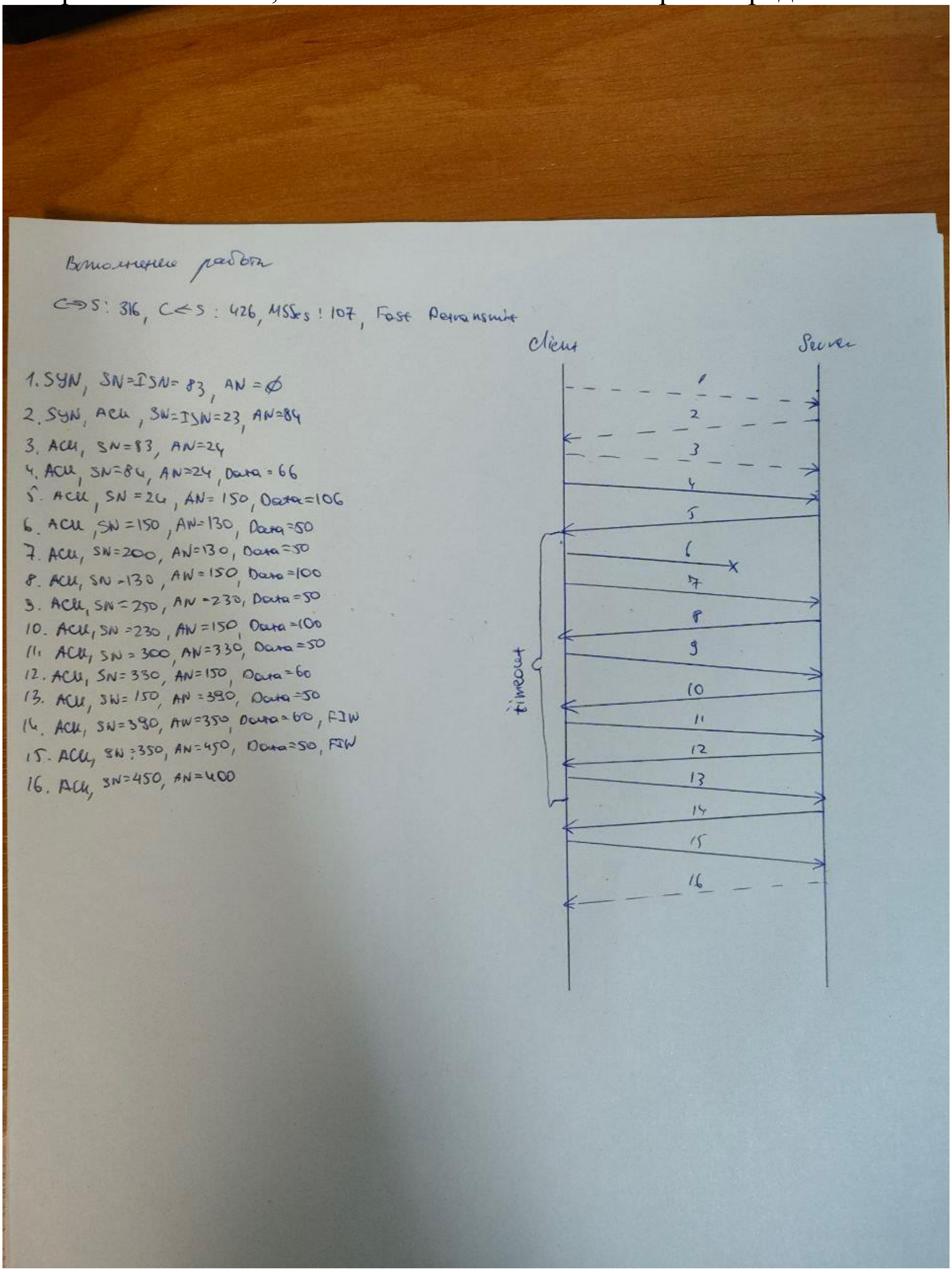
5.2 Пауза при пересылке

Суть : в какой то момент времени возникает пауза при пересылке. На схеме это будет выглядеть как увеличенное расстояние между сигналами. В этом промежутке рисуем закрашенный прямоугольник и даём ему номер. То есть, был 8 сигнал, потом пауза, прямоугольник – номер 9, следующий сигнал – 10. AN, SN – везде как были бы без паузы.



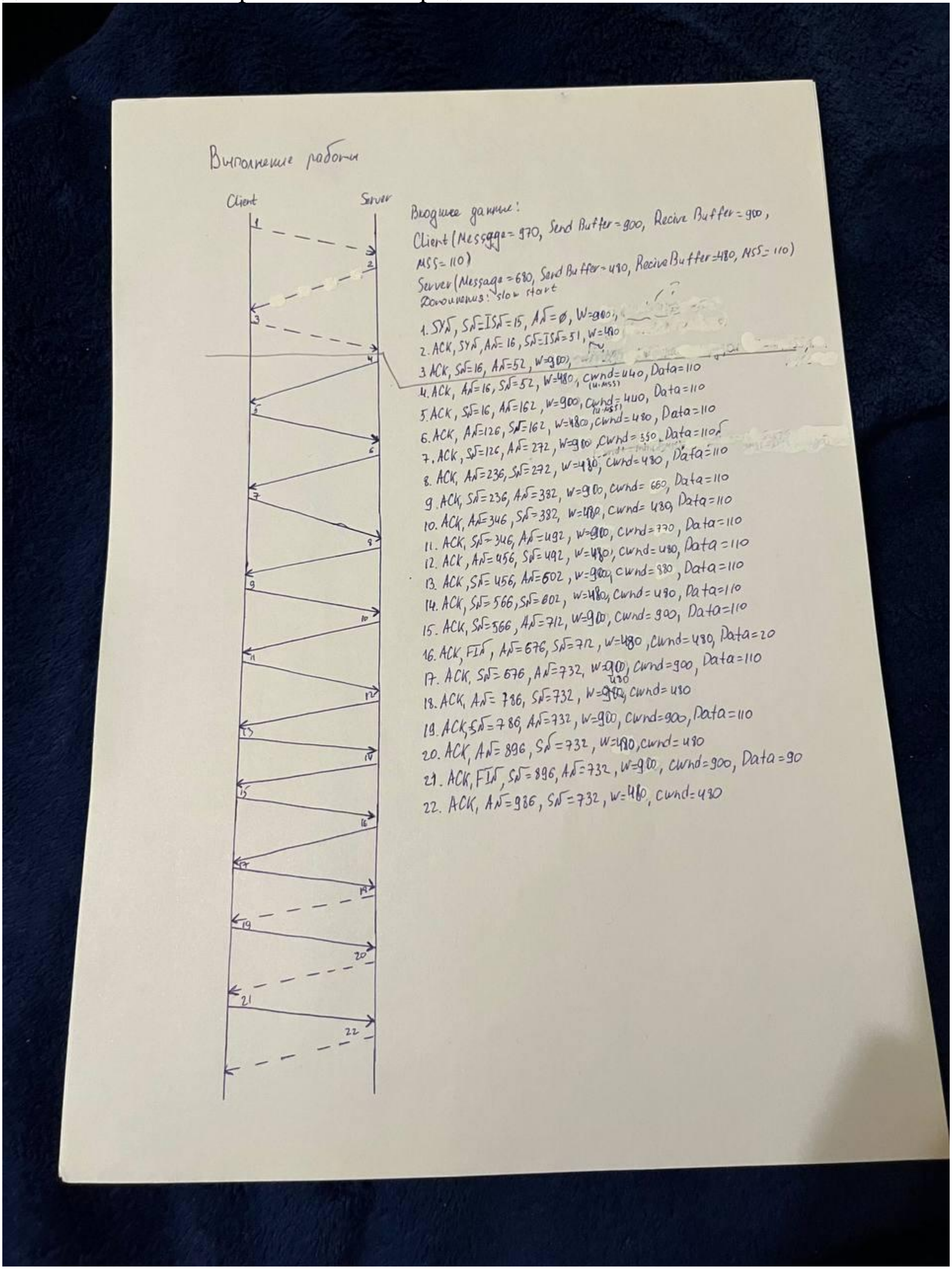
5.3 Fast retransmit

СУТЬ : в какой то момент теряется сигнал, обозначается крестиком на линии отправки. Принимающая сторона будет слать 3 сигнала с AN = номер потерянного сегмента в ответ на принимаемые сигналы. Отправляющая сторона будет слать дальше следующие сегменты. В ответ на 3 сигнал о потерянном сегменте, отправляющая сторона отправляет запрашиваемый потерянный сегмент, после этого всё шлется в старом порядке.



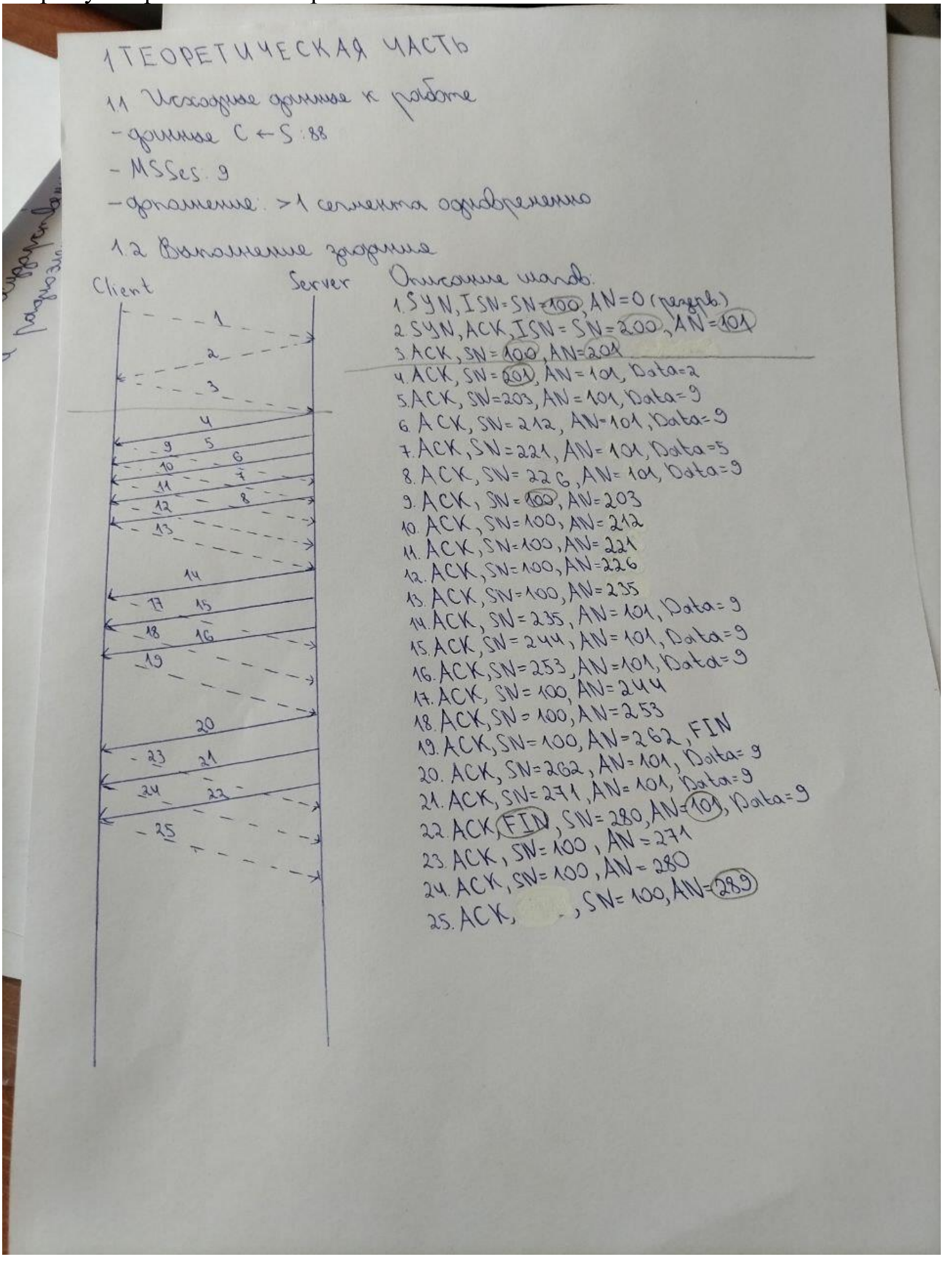
5.4 Slow start

СУТЬ: изначально нужно смотреть на mss – максимальный размер сегмента. Если меньше или равно 1095 - то размер начальный окна равен 4 * MSS. Если больше 1095 и меньше или равен 2190, то 3*MSS. Если больше, то 2 * MSS. УВЕЛИЧИВАЕТСЯ ТОЛЬКО РАЗМЕР ОКНА. Данные могут быть максимум размером с MSS. Просто каждый раз увеличиваем размер окна на mss, но данные шлем как при обычной передаче.



5.5 >1 СЕГМЕНТА

СУТЬ : может отправляться более одного сегмента за раз, однако подтверждение должно идти на каждый принятый сегмент. К примеру, отправили 3 сегмента, должно с приёмника прийти 3 сигнала подтверждения. На картинке 4 5 6 линии и 7 8 9. У первой ответной линии будет AN равный второму отправленного за раз сегмента.


1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исходные данные к работе

- данные C ← S: 88
- MSSes: 9
- дополнение: >1 сегмента одновременно

1.2 Выполнение задания

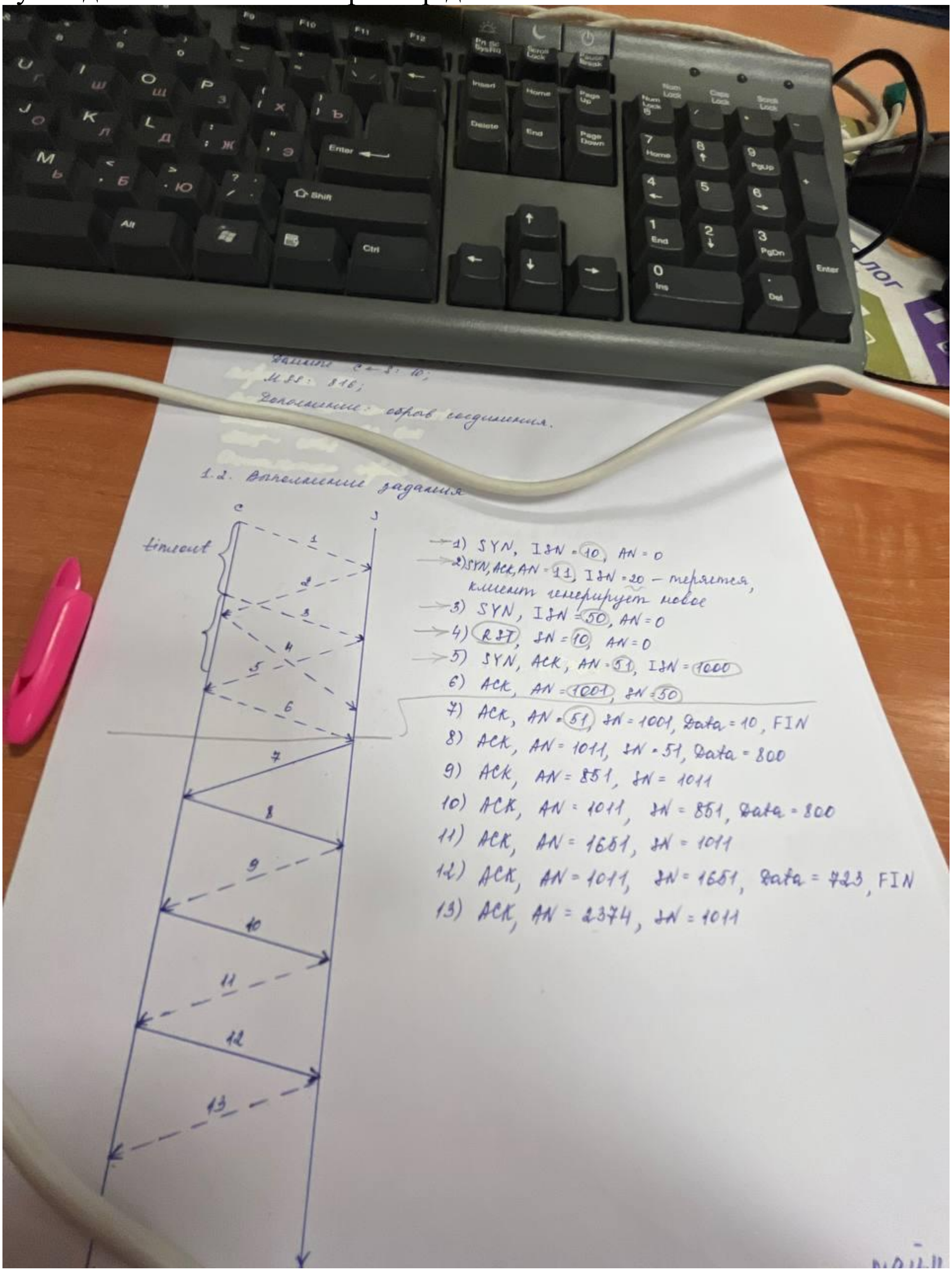
Client Server

Описание шагов:

1. SYN, ISN=SN=100, AN=0 (резерв.)
2. SYN, ACK, ISN=SN=200, AN=101
3. ACK, SN=100, AN=201
4. ACK, SN=201, AN=101, Data=2
5. ACK, SN=203, AN=101, Data=9
6. ACK, SN=212, AN=101, Data=9
7. ACK, SN=221, AN=101, Data=5
8. ACK, SN=226, AN=101, Data=9
9. ACK, SN=100, AN=203
10. ACK, SN=100, AN=212
11. ACK, SN=100, AN=221
12. ACK, SN=100, AN=226
13. ACK, SN=100, AN=235
14. ACK, SN=235, AN=101, Data=9
15. ACK, SN=244, AN=101, Data=9
16. ACK, SN=253, AN=101, Data=9
17. ACK, SN=100, AN=244
18. ACK, SN=100, AN=253
19. ACK, SN=100, AN=262, FIN
20. ACK, SN=262, AN=101, Data=9
21. ACK, SN=271, AN=101, Data=9
22. ACK, FIN, SN=280, AN=101, Data=9
23. ACK, SN=100, AN=271
24. ACK, SN=100, AN=280
25. ACK, , SN=100, AN=289

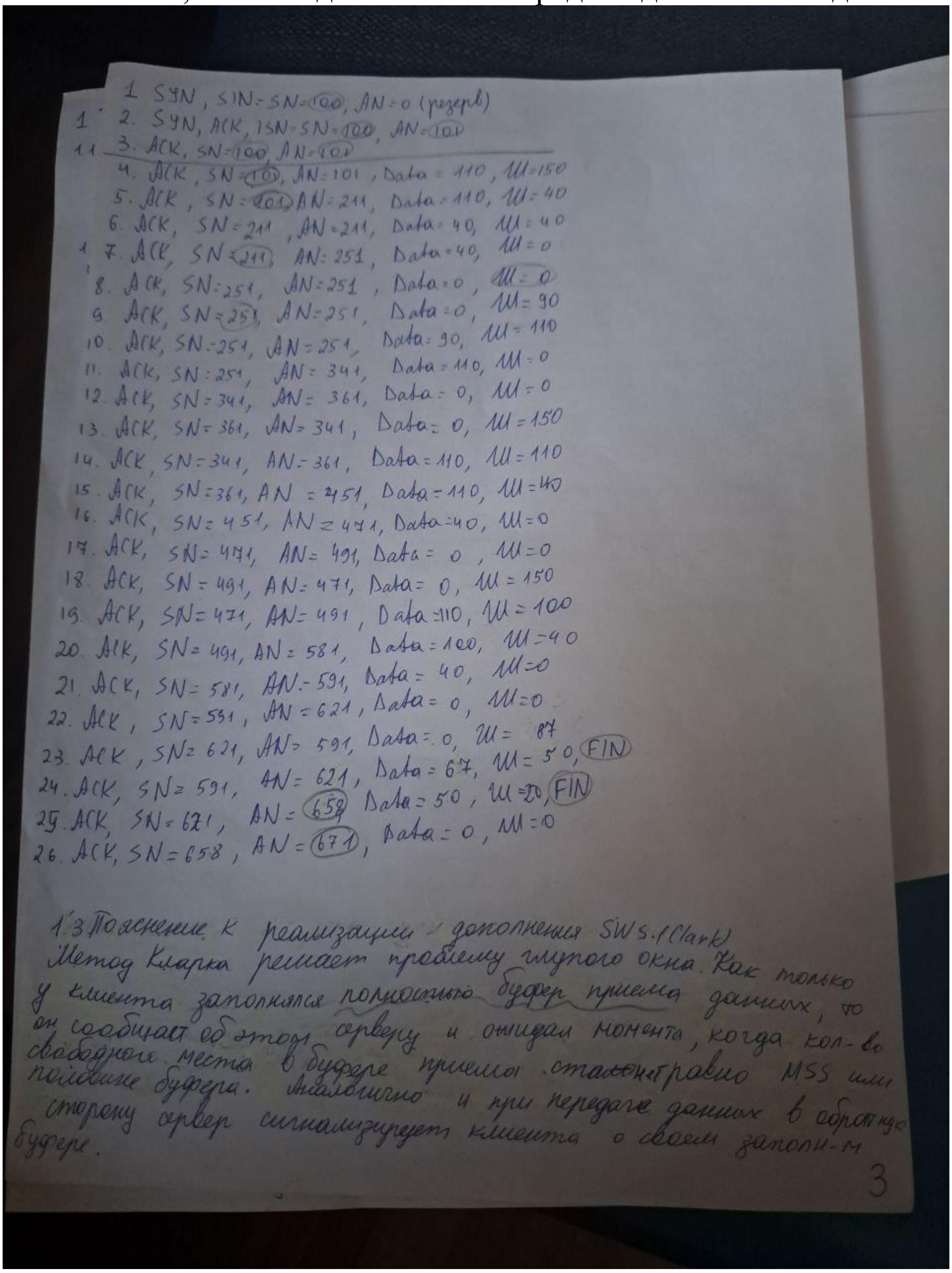
5.6 ОБРЫВ(РАЗРЫВ) СОЕДИНЕНИЯ

тут надо поменять только размер данных



5.7 КЛАРК

СУТЬ: метод кларка решает проблему глупого окна. Нам нужно на принимающей стороне нарисовать буфер при каждом приёме. Каждый раз размер буфера будет уменьшаться на количество полученных байт. Буфер подбираем такого размера, чтобы после какого то количества приёмов он стал равен 0. После этого делаем расстояние между сигналами, подписываем как «очистка буфера». Перед очисткой последним сигналом от принимающей стороны подаем также W = 0. Первым сигналом сразу после очистки посылаем еще один сигнал от принимающей стороны, сигнал с W равному размеру mss и AN такой же, как был до очистки и передаем данные как и до этого.


1. SYN, SIN=SN=100, AN=0 (резерв)
2. SYN, ACK, ISN=SN=100, AN=101
3. ACK, SN=100, AN=101
4. ACK, SN=101, AN=101, Data=110, W=150
5. ACK, SN=101, AN=211, Data=110, W=40
6. ACK, SN=211, AN=211, Data=40, W=40
7. ACK, SN=211, AN=251, Data=40, W=0
8. ACK, SN=251, AN=251, Data=0, W=0
9. ACK, SN=251, AN=251, Data=0, W=90
10. ACK, SN=251, AN=251, Data=90, W=110
11. ACK, SN=251, AN=341, Data=110, W=0
12. ACK, SN=341, AN=361, Data=0, W=0
13. ACK, SN=361, AN=341, Data=0, W=150
14. ACK, SN=341, AN=361, Data=110, W=110
15. ACK, SN=361, AN=451, Data=110, W=40
16. ACK, SN=451, AN=471, Data=40, W=0
17. ACK, SN=471, AN=491, Data=0, W=0
18. ACK, SN=491, AN=471, Data=0, W=150
19. ACK, SN=471, AN=491, Data=110, W=100
20. ACK, SN=491, AN=581, Data=100, W=40
21. ACK, SN=581, AN=591, Data=40, W=0
22. ACK, SN=591, AN=621, Data=0, W=0
23. ACK, SN=621, AN=591, Data=0, W=87
24. ACK, SN=591, AN=621, Data=67, W=50, FIN
25. ACK, SN=621, AN=658, Data=50, W=20, FIN
26. ACK, SN=658, AN=671, Data=0, W=0

1.3 Пояснение к реализации дополнения SWS.1 (Clark)
Метод Кларка решает проблему глупого окна. Как только у клиента заполнялся полностью буфер приема данных, то он сообщает об этом серверу и ожидал момента, когда кол-во свободного места в буфере приема станет равно MSS или половине буфера. Аналогично и при передаче данных в обратную сторону сервер сигнализирует клиента о своем заполн-м буфере.

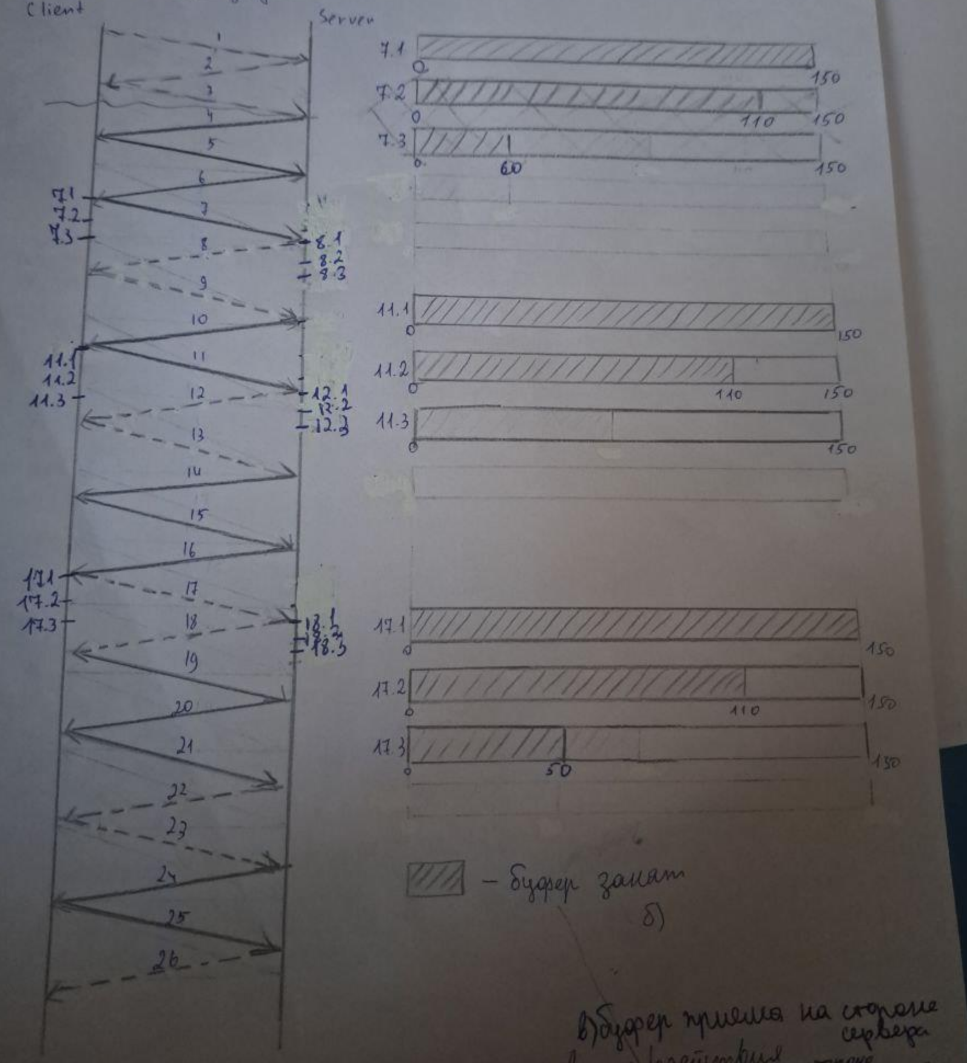
3

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Исходные данные к работе

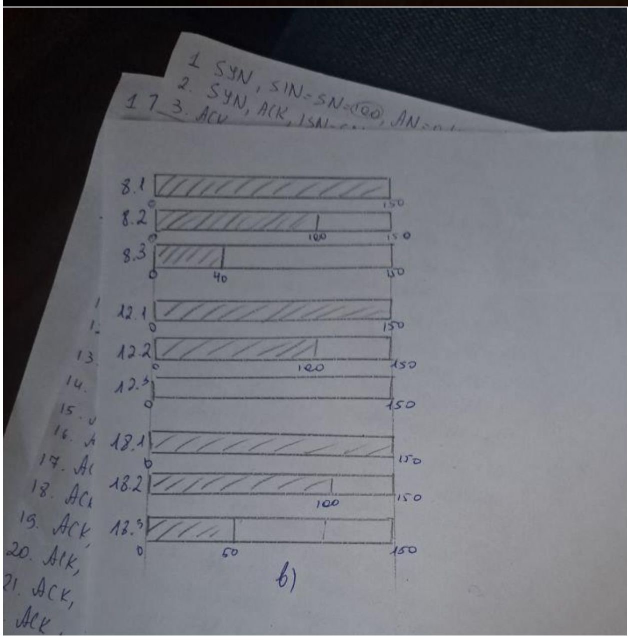
- данные: C → S : 570, S → C : 557
- MSS : 111
- дополнение : SWS (Clark)

1.2. Выполнение задания

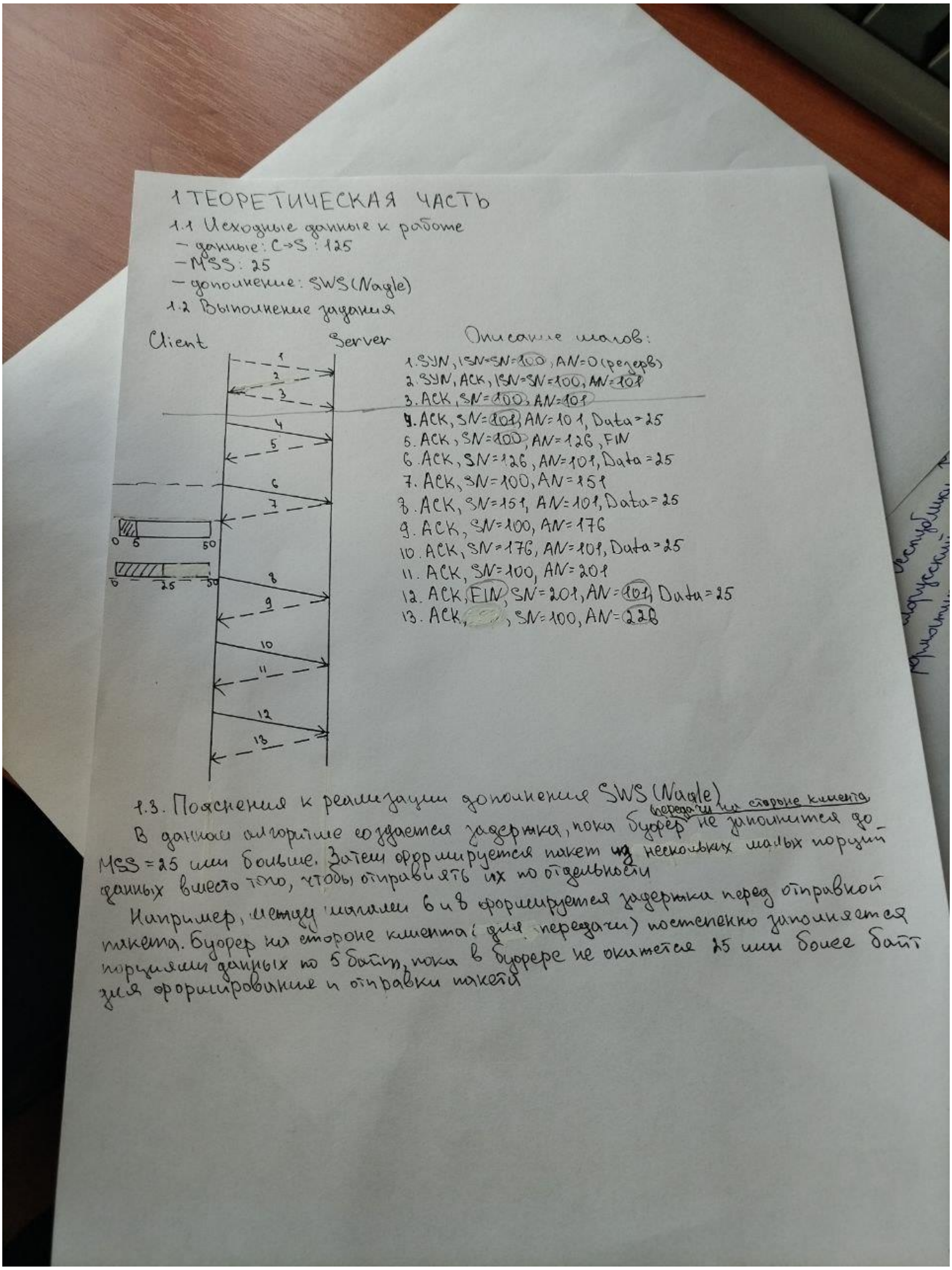


в) буфер приема на стороне сервера

а) Рисунок 1 – а) диаграмма взаимодействия б) буфер приема на стороне клиента



5.8 NAGLE



1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исходные данные к работе
- данные: C→S : 125
- MSS : 25
- дополнение: SWS (Nagle)

1.2 Выполнение задания

Описание шагов:

1. SYN, ISN=SN=100, AN=0 (резерв)
2. SYN, ACK, ISN=SN=100, AN=101
3. ACK, SN=100, AN=101
4. ACK, SN=101, AN=101, Data=25
5. ACK, SN=100, AN=126, FIN
6. ACK, SN=126, AN=101, Data=25
7. ACK, SN=100, AN=151
8. ACK, SN=151, AN=101, Data=25
9. ACK, SN=100, AN=176
10. ACK, SN=176, AN=101, Data=25
11. ACK, SN=100, AN=201
12. ACK, FIN, SN=201, AN=101, Data=25
13. ACK, , SN=100, AN=226

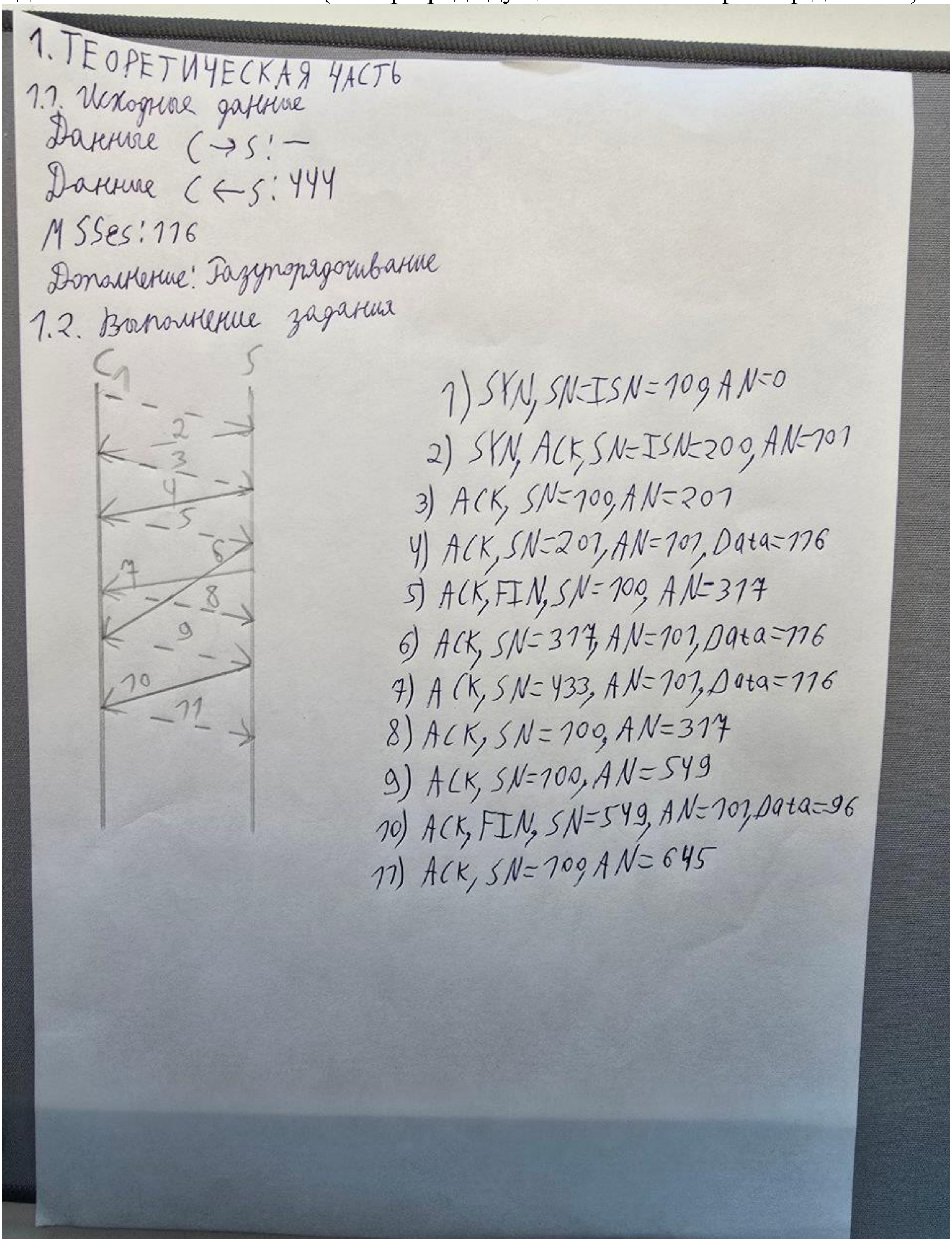
1.3. Пояснения к реализации дополнения SWS (Nagle)

В данном алгоритме создается задержка, пока буфер передачи на стороне клиента не заполнится до MSS = 25 или больше. Затем формируется пакет из нескольких малых порций данных вместо того, чтобы отправлять их по отдельности.

Например, между шагами 6 и 8 формируется задержка перед отправкой пакета. Буфер на стороне клиента (для передачи) постепенно заполняется порциями данных по 5 байт, пока в буфере не окажется 25 или более байт для формирования и отправки пакета.

5.9 РАЗУПОРЯДОЧИВАНИЕ

СУТЬ : сегменты отправляются не в том порядке, то есть, допустим, после тройного рукопожатия шлется 4 сигнал и сразу за ним 5ый, не дожидаясь подтверждения 4ого. И 5ый приходит быстрее, чем 4ый. В таком случае, ответный сигнал и на 4 и на 5 будут с AN следующего сегмента, то есть будут запрашивать следующий сегмент. Пример: 4 линия была с SN = 145, 5 линия с SN = 260. Размер данных – 115. 6 и 7 линия в таком случае будут запрашивать один и тот же сегмент 375(номер предыдущего сегмента + размер данных.)



1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Исходные данные

Данные C→S: —

Данные C←S: 444

MSSes: 116

Дополнение: Разупорядочивание

1.2. Выполнение задания

1) SYN, SN=ISN=100 AN=0
2) SYN, ACK, SN=ISN=200, AN=101
3) ACK, SN=100, AN=201
4) ACK, SN=201, AN=101, Data=116
5) ACK, FIN, SN=100, AN=317
6) ACK, SN=317, AN=101, Data=116
7) ACK, SN=433, AN=101, Data=116
8) ACK, SN=100, AN=317
9) ACK, SN=100, AN=549
10) ACK, FIN, SN=549, AN=101, Data=96
11) ACK, SN=100 AN=645