

## **Práctica 1**

Mi capa de TCP/IP

Jorge Alejandro Flores Triana

ID: 714510



**ITESO, Universidad  
Jesuita de Guadalajara**

Instituto Tecnológico y de Estudios Superiores de Occidente

Profesor – Sergio Nicolás Santana

Tópico: Sistemas de Comunicación

## Mi capa de TCP/IP

Link Video:

<https://drive.google.com/drive/folders/18rpOBIPAgmBrwcaXkDslcyxDzO0WqaTx?usp=sharing>

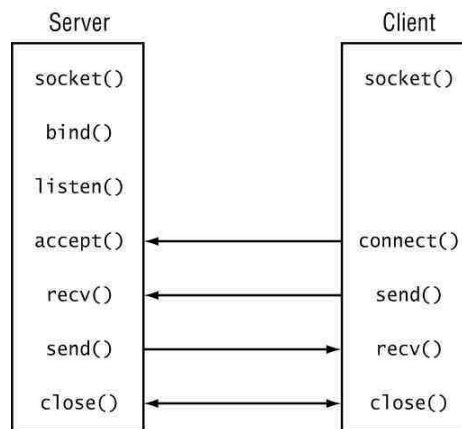
Link Código: <https://github.com/a01633675/SistemasComunicacion>

### Objetivo:

El objetivo de esta práctica era crear una capa propia dentro del modelo de TCP/IP, el cual consistía en recibir/mandar una trama de TCP, encriptar/desencriptar el mensaje utilizando AES 128 y calcular el CRC del mismo.

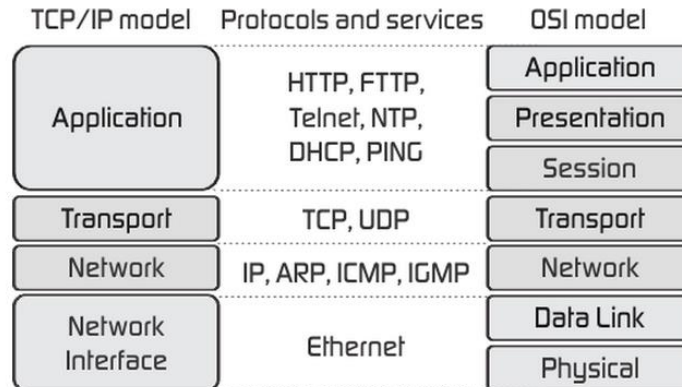
### Implementación:

Los sockets son un mecanismo que nos permite establecer un enlace entre dos programas que se ejecutan independientes el uno del otro (generalmente un programa cliente y un programa servidor). En el modelo de TCP/IP la implementación de un cliente – servidor se vería de la siguiente manera.



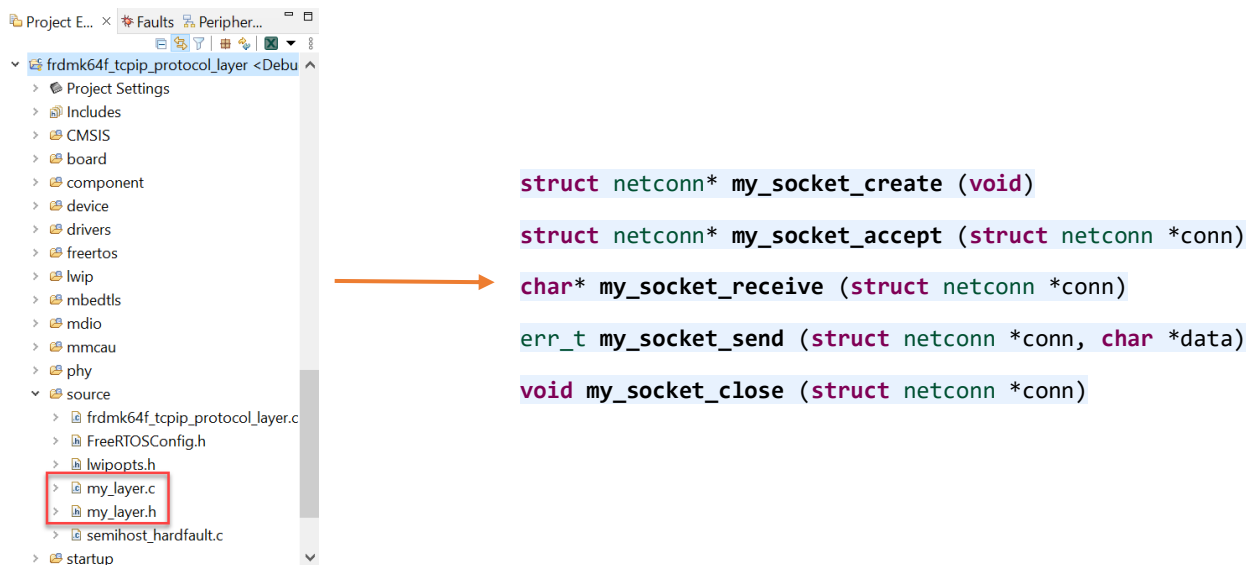
Como se puede observar, el servidor crea un socket y espera por la conexión de un cliente. Cuando el cliente hace la petición, el servidor acepta la conexión y la comunicación puede iniciar. Para el caso de nuestra práctica se decidió que la tarjeta Freedom K64 sería el servidor y la computadora el cliente.

Por otro lado, es importante tomar como referencia el modelo TCP/IP que se muestra a continuación para entender en que sección del modelo se iba a estar trabajando.



Para la práctica se tomó como base el ejemplo de TCP echo que consiste en mandar tramas de TCP desde un cliente hacia un servidor y viceversa, agregando las tareas de encriptado y cálculo de CRC.

En la práctica se implementó un driver que contiene las siguientes funciones que pueden ser utilizadas en por cualquier usuario.



Cada función puede ser utilizada independientemente y su implementación específica puede ser encontrada en el código fuente. Sin embargo, a continuación se da una pequeña descripción del código:

- `my_socket_create ()`: Esta función crea un socket de TCP, donde por medio de unos `#defines` se puede configurar la dirección IP y el puerto que se quiere utilizar.
- `my_socket_accept ()`: Esta función es implementada por el servidor para aceptar conexiones del cliente. Regresa el socket para entablar la comunicación.

- `my_socket_receive ()`: Esta función recibe una trama de TCP y la separa en el cuerpo del mensaje y el CRC (últimos 4 bytes de la trama). Después, se calcula el CRC del cuerpo del mensaje y se compara con el valor recibido. Si son iguales, se procede a desencriptar el cuerpo del mensaje utilizando AES 128. Finalmente, se regresa un puntero con el mensaje original.
- `my_socket_send ()`: Esta función recibe como parámetro un puntero con el dato que se quiere enviar. Posteriormente, se le hace un padding de ceros a 16 bytes y se encripta el mensaje con AES 128. Finalmente, se calcula el CRC del mensaje encriptado y se agrega el resultado a los últimos 4 bytes de la trama antes de ser enviada.
- `my_socket_close ()`: El servidor Cierra la conexión y elimina el socket utilizado.

Una vez, con las funciones definidas, se implementó una tarea de FreeRTOS en donde se levanta el socket del servidor y se espera por una conexión. Después, cuando un cliente busca conectarse, se acepta la solicitud y se comienza a recibir/enviar mensajes.

El cliente (script de python en la computadora) envía un mensaje desde la consola y el servidor (Freedom K64) recibe la trama. Si el cálculo del CRC y el proceso de desencriptado son correctos, el servidor envía una respuesta al cliente y éste lo recibe para checar la integridad del mensaje. Este proceso de recibir y luego enviar se repite 8 veces. Finalmente, el servidor cierra la conexión.

### **Resultados:**

Para poder ver los resultados de la aplicación, se imprime el contenido de distintas variables en la terminal incluyendo el CRC, el mensaje encriptado y el mensaje original.

Como se observa en la primer imagen, el usuario escribe un mensaje en la consola para que el cliente lo envíe al servidor. El mensaje es: “Quién eres?”. El cliente encripta el mensaje y le agrega el CRC para posteriormente enviarlo. El servidor recibe el mensaje, lo valida y obtiene la información original como se muestra en la terminal del IDE.

Después, el servidor manda una un mensaje: “Soy Alejandro Triana”. Dicha respuesta se envía de manera encriptada y con su respectivo CRC. Finalmente, el cliente recibe la información, la valida y la decodifica para obtener el mensaje final, como se muestra en la consola.

Este proceso se repite 8 veces donde el servidor recibe un mensaje por parte del cliente y una vez validado, envía una respuesta previamente definida por la aplicación.

The screenshot shows a development environment with a code editor on the left and a terminal window on the right. The code editor displays a C file named `frdmk64f_tcpip_protocol_layer.c` with various network-related functions and comments. The terminal window shows the output of a Python script `myssn_client.py` running in a command prompt. The logs indicate a successful connection and data exchange between a client and a server.

```

94  SYSMPU_Type *base = SYSMPU
95
96  /* Initialize board hardware
97  BOARD_InitBootPins ( );
98  BOARD_InitBootClocks (
99  BOARD_InitBootPeripherals (
100
101  #ifndef BOARD_INIT_DEBUG_CONSOLE
102  /* Initialize FSL debug console
103  BOARD_InitDebugConsole (
104  #endif
105
106  /* Print a note to terminal
107  PRINTF("Practica 1: TCP/IP
108
109  /* Disable SYSMPU.
110  base -> CESR &= ~SYSMPU
111
112  /* Initialize lwIP from
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

## Conclusión:

Durante la práctica aprendí a identificar de mejor manera las distintas capas del modelo TCP/IP y cómo se relacionan dentro de una aplicación. Esto dejó en claro uno de los objetivos que es mantener independientes cada nivel de la aplicación, facilitando su mantenimiento debido a la modularidad y a la escalabilidad. Por otro lado, a nivel de la implementación fue interesante crear una capa propia para entender cómo se construyen las aplicaciones en casos de uso de la vida real.

Por otro lado, hablando de la práctica en sí, creo que ayudó en gran parte haber corrido los ejemplos de TCP en clase ya que nos ayudaron a entender de mejor manera cómo funciona el stack de lwip. En cuanto a las dificultades, creo que el ejercicio de encriptar y calcular el CRC fue algo nuevo para mí ya que nunca había visto temas relacionados con la seguridad. Y aunque no es el objetivo principal de la clase el saber encriptar, creo que sirvió en gran medida para entender cómo se construyen las capas.