

# 컴퓨터 네트워크

## -5주차-

학번: 201404377

이름: 진승언

## 과제 목표(도출해야 할 결과)

지금까지의 과제는 도커 컨테이너와 파이썬 기반 서버를 두고 네트워크 통신을 했다면 이번에는 안드로이드 폰을 이용해서 자바 기반으로 소켓 통신을 하는 것이 과제 목표였다. 따라서 폰 하나는 서버 다른 폰 하나는 클라이언트를 맡고 소켓 통신을 하는 결과를 구현했고 프로젝트는 서버와 클라이언트 두 개로 나눠서 만들었다.

## 코드 설명과 과제 해결 방법

[서버]

```
149 //내 ip 얻기
150 private String getLocalIpAddress() throws UnknownHostException {
151     WifiManager wifiManager = (WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE);
152     assert wifiManager != null;
153     WifiInfo wifiInfo = wifiManager.getConnectionInfo();
154     int ipInt = wifiInfo.getIpAddress();
155     return InetAddress.getByAddress(ByteBuffer.allocate(4).order(ByteOrder.LITTLE_ENDIAN).putInt(ipInt).array()).getHostAddress();
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
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ipText = findViewById(R.id.ip);
    portText = findViewById(R.id.port);
    text_msg = findViewById(R.id.chatting);
    edit_msg = findViewById(R.id.msg);

    //내 아이피 확인 및 세팅
    try {
        ipText.setText(getLocalIpAddress());
    } catch (UnknownHostException e) {
        e.printStackTrace();
    }
}
```

위의 getLocalIpAddress() 메소드를 통해서 현재 내 휴대폰의 ip를 알 수 있고 TextView에 set해준다. 클라이언트에서 서버 ip를 알아야 접속이 가능하므로 필요하다.

```

60
61 public void ServerSocketOpen(View view) throws IOException {
62     final String port = portText.getText().toString();
63     if(port.isEmpty()){
64         Toast.makeText( context: this, text: "포트번호를 입력해주세요.", Toast.LENGTH_SHORT).show();
65     }else {
66         Toast.makeText( context: this, text: "Socket Open", Toast.LENGTH_SHORT).show();
67         //Android API14버전이상 부터 네트워크 작업은 무조건 별도의 Thread에서 실행 해야함.
68         new Thread((Runnable) () -> {
69             // TODO Auto-generated method stub
70             try {
71                 //서버소켓 생성.
72                 serversocket = new ServerSocket(Integer.parseInt(port));
73             } catch (IOException e) {
74                 // TODO Auto-generated catch block
75                 e.printStackTrace();
76             }
77             try {
78                 //서버에 접속하는 클라이언트 소켓 얻어오기(클라이언트가 접속하면 클라이언트 소켓 리턴)
79                 socket = serversocket.accept(); //서버는 클라이언트가 접속할 때까지 여기서 대기 접속하면 다음으로 코드로 넘어감
80                 //클라이언트와 데이터를 주고 받기 위한 통로구축
81                 is = new DataInputStream(socket.getInputStream()); //클라이언트로부터 메시지를 받기 위한 통로
82                 os = new DataOutputStream(socket.getOutputStream()); //클라이언트로 메시지를 보내기 위한 통로
83
84                 MainActivity.this.runOnUiThread(new Runnable() {
85                     public void run() {
86                         Toast.makeText( context: MainActivity.this, text: "Connected With Client Socket", Toast.LENGTH_SHORT).show();
87                     }
88                 });
89             } catch (IOException e) {
90                 // TODO Auto-generated catch block
91                 e.printStackTrace();
92             }
93             //클라이언트가 접속을 끊을 때까지 무한반복하면서 클라이언트의 메시지 수신
94             while (!isDisconnected) {
95                 try {
96                     msg = is.readUTF();
97                 } catch (IOException e) {
98                     // TODO Auto-generated catch block
99                     e.printStackTrace();
100                 }
101                 //클라이언트로부터 읽어들이는 메시지를 TextView에 출력한다. 안드로이드는 메인 스레드가 아니면 UI변경 불가하므로 다음과같이 해줌.(토스트메세지도 마찬가지)
102                 runOnUiThread(new Runnable() {
103                     @Override
104                     public void run() {
105                         // TODO Auto-generated method stub
106                         text_msg.setText(text_msg.getText() + msg);
107                     }
108                 });
109             }
110             //while..
111             //start(); //Thread 실행..
112         });
113     }
114 }
115
116
117

```

ServerSocketOpen()은 Open Server 버튼을 누르면 동작할 온클릭 메소드이다.

[62~78라인]: 소켓을 생성하려면 포트번호가 필요하다. 그러므로 포트번호를 editText에서 입력했는지 검사 후 서버 소켓을 생성한다. 안드로이드에서는 소켓 통신을(네트워크 통신) 할 때 스레드가 필수적으로 사용된다.

[81~94라인]: accept()로 서버에 접속하는 클라이언트의 소켓을 얻어온다. (클라이언트가 접속할때까지 대기한다.) 그 후 클라이언트가 접속하면 데이터를 주고받기 위한 통로를 DataInputStream과 DataOutputStream으로 구성한다. 또한 클라이언트와 연결되었다고 토스트메세지를 띄워주는데 안드로이드에서는 메인스레드 외에 스레드에서는 UI를 변경하거나 띄워줄수 없기 때문에 runOnUiThread()를 사용해줘야 한다.

[96~112라인]: 클라이언트가 접속을 끊을 때까지 while문으로 돌리면서 클라이언트의 메시지를 계속 수신할 수 있게 한다. 메시지는 앞서 선언한 DataInputStream (통로)의 readUTF()를 통해 받을 수 있다. UTF는 한글을 수신할 수 있기 위해 필요하다. 그리고 수신받은 메시지를 TextView에 띄워주도록 한다. 이것도 마찬가지로 메인스레드가 아니므로 runOnUiThread()를 사용하여 UI를 변경하도록 한다.

```
118 //메시지전송
119 public void SendMessage(View view) {
120     if(os==null) return; //클라이언트와 연결되어 있지 않다면 전송불가
121     final String msg= edit_msg.getText().toString();
122     //네트워크 작업이므로 Thread 생성
123     new Thread((Runnable) () → {
124         // TODO Auto-generated method stub
125         //클라이언트로 보낼 메시지 EditText로 부터 얻어오기
126
127         try {
128             runOnUiThread(new Runnable() {
129                 @Override
130                 public void run() {
131                     // TODO Auto-generated method stub
132                     text_msg.setText("[SEND] "+msg);
133                 }
134             });
135         } catch (IOException e) {
136             // TODO Auto-generated catch block
137             e.printStackTrace();
138         }
139     }).start(); //Thread 실행..
140
141     os.writeUTF(msg); //클라이언트로 메시지 보내기 .UTF 방식으로 한글 전송 가능하게함
142     os.flush(); //다음 메시지 전송을 위해 연결통로의 버퍼를 지워주는 메소드
143 } catch (IOException e) {
144     // TODO Auto-generated catch block
145     e.printStackTrace();
146 }
147 }
```

메시지 전송버튼을 눌렀을 때 동작하는 온클릭 메소드이다.

[120라인]: DataOutputStream (os) 가 null이면 앞선 코드에서 살펴볼 수 있듯이 클라이언트가 접속되지 않은 것을 의미하므로 메시지를 보낼 수 없다.

[123~145라인]: 전송할 메시지를 갖고온 후 소켓 통신 즉 네트워크 통신을 해야 하므로 스레드가 사용된다. DataOutputStream 객체의 writeUTF를 통해 메시지를 보내준다. 그리고 다음 메시지 전송을 위해 DataOutputStream 연결통로의 버퍼를 flush()로 지워준다. 송신한 메시지를 자신의 폰에서 볼 수 있게 setText() 해준다. 이것도 UI를 건드는 작업이므로 runOnUiThread()를 이용한다.

## [클라이언트]

```
46 //클라이언트 소켓 열고 서버 소켓에 접속
47 public void ClientSocketOpen(View view) {
48     new Thread((Runnable) () -> {
49         // TODO Auto-generated method stub
50         try {
51             ip= edit_ip.getText().toString();//IP 주소가 작성되어 있는 EditText에서 서버 IP 얻어오기
52             port = edit_port.getText().toString();
53             if(ip.isEmpty() || port.isEmpty()){
54                 MainActivity.this.runOnUiThread(new Runnable() {
55                     public void run() {
56                         Toast.makeText( context MainActivity.this, text: "[IP주소와 포트번호를 입력해주세요.]", Toast.LENGTH_SHORT).show();
57                     }
58                 });
59             }
60         } catch (IOException e) {
61             // TODO Auto-generated catch block
62             e.printStackTrace();
63         }
64     }
65     //서버와 연결하는 소켓 생성..
66     socket = new Socket(InetAddress.getByName(ip), Integer.parseInt(port));
67     //여기까지 왔다는 것을 예외가 발생하지 않았다는 것이므로 소켓 연결 성공..
68
69     //서버와 메시지를 주고받을 물로 구축
70     is = new DataInputStream(socket.getInputStream());
71     os = new DataOutputStream(socket.getOutputStream());
72
73     MainActivity.this.runOnUiThread(new Runnable() {
74         public void run() {
75             Toast.makeText( context MainActivity.this, text: "Connected With Server", Toast.LENGTH_SHORT).show();
76         }
77     });
78 } catch (IOException e) {
79     // TODO Auto-generated catch block
80     e.printStackTrace();
81 }
82
83 //서버와 접속이 끊길 때까지 무한반복하면서 서버의 메시지 수신
84 while(!isConnected){
85     try {
86         msg= is.readUTF(); //서버 부터 메시지가 전송되면 이를 UTF형식으로 읽어서 String 으로 리턴
87         //runOnUiThread()는 별도의 Thread와 main Thread에게 이 작업을 요청하는 메소드이다
88         runOnUiThread(() -> {
89             // TODO Auto-generated method stub
90             text_msg.setText("RECV" +msg);
91         });
92     } catch (IOException e) {
93         // TODO Auto-generated catch block
94         e.printStackTrace();
95     }
96 }
97 }
98 }.start();//Thread 실행..
99
100 }
101
102 }
```

접속 버튼을 누르면 클라이언트의 소켓을 생성하고 서버 소켓에 연결하는 온 클릭 메소드이다.

[48~79라인]: ip와 port를 EditText를 통해 받는다. 이때 ip는 서버의 ip와 열 어논 포트번호를 입력해야 한다. 그리고 서버에서 소켓을 생성했던 것처럼 소켓을 생성한다. (다른점은 ip를 입력해야 한다.) 통로도 서버에서 했던 것처럼 생성해준다. 서버와 연결되었다면 토스트메세지를 띄워주도록 한다.

[82~100라인]: 서버와 접속이 끊길 때 까지 무한 반복하면서 서버의 메시지를 수신하고 UI에 setText() 해준다.. 이것은 서버에서의 코드와 똑같다.

```

103 public void SendMessage(View view) {
104     if(os==null) return; //서버와 연결되어 있지 않다면 전송불가...
105
106     //네트워크 작업이므로 Thread 생성
107     new Thread((Runnable) () → {
108         // TODO Auto-generated method stub
109         //서버로 보낼 메시지 EditText로부터 얻어오기
110         String msg= edit_msg.getText().toString();
111         try {
112             runOnUiThread(new Runnable() {
113                 @Override
114                 public void run() {
115                     String msg= edit_msg.getText().toString();
116                     // TODO Auto-generated method stub
117                     text_msg.setText("[ SEND ] " +msg);
118                 }
119             });
120             os.writeUTF(msg); //서버로 메시지 보내기, UTF 방식으로(한글 전송가능...)
121             os.flush(); //다음 메시지 전송을 위해 연결통로의 버퍼를 지워주는 메소드..
122         } catch (IOException e) {
123             // TODO Auto-generated catch block
124             e.printStackTrace();
125         }
126     }).start(); //Thread 실행..
127 }

```

메시지 전송버튼을 클릭하면 서버에 메시지를 전송해주는 온 클릭 메소드이다. 서버에서의 메시지 전송과 코드가 동일하다. 스레드를 이용해서 네트워크 통신을 하고 stream 통로를 통해 메시지를 전송하고 flush()로 버퍼를 비워준다.

```

135 @Override
136 protected void onStop() {
137     super.onStop();
138     try {
139         socket.close(); //소켓을 닫는다.
140     } catch (IOException e) {
141         e.printStackTrace();
142     }
143 }

```

안드로이드 생명주기인 onStop() 시 소켓연결을 close()하도록 해주었다.

[P.S]

```

4 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
5 <uses-permission android:name="android.permission.INTERNET"/>

```

안드로이드에서 manifest에 다음과 같이 접근권한을 추가해야 해한다.

## 실행결과 youtube 링크

<https://www.youtube.com/watch?v=OvRQpf1oZ5M&feature=youtu.be>

## 과제후기

다양한 환경으로 소켓통신을 구현해 볼 수 있어서 좋았습니다.