스마트네트워크서비스 AD 과제 안내

발표 일정

- **일시:** 2025년 **12월 11일(목) 15:00 ~ 16:30**
- **장소:** 강의실 내 오프라인 발표 (프로젝터 연결 가능)
- **발표 시간:** 조별 **약 10분 이내**
- **형식:** 직접 시연 또는 발표 영상(4분 이내) 재생 가능

조별 구성

- **1인 1조, 2인 1조, 3인 1조 모두 가능**
- 조편성은 아래 **스프레드시트 링크**에 작성

  https://docs.google.com/spreadsheets/d/1qhZdyVvanCqgxFEOXXMWriaWefBB9x_EjTzHGwhAOOg/edit?usp=sharing
  - 모든 조원 이름, 학번 반드시 기입
  - 조별 프로젝트 내용 동일해야 하며, 각자 개별 제출 (ZIP 업로드)

과제 내용

- 제공된 **스켈레톤 코드**(GUI 기반 네트워크 툴 + SFC 구조) 활용
- 스켈레톤에 각자 기능 구현 / 확장
- 주요 주제 예시
  - TCP 네트워크 툴, 서버/클라이언트, DNS, 바이트 변환 등
  - Ryu OpenFlow 기반 SFC(Service Function Chain) 구성
  - GUI 기반 실시간 네트워크 모니터링, 그림판, REST 연동

제출

- **기한:** 2025년 **12월 11일(목)** 수업 시작 전
- **제출물**
  1. 구현 코드 ZIP (`조원명_학번.zip`)
  2. 발표 영상 (YouTube 링크)
  3. 발표 자료 (PPT 또는 PDF, 조원 정보 포함)
- **제출 경로:** 가상 강의실(이클래스/캔버스)

평가 기준 요약

항목	배점	주요 내용
기능 구현	30	스켈레톤 기반 기능 완성도
GUI 완성도	15	동작 안정성, 레이아웃
네트워크 기능	10	진단/전송/변환 정확성

항목	배점	주요 내용
TCP 서버·클라	20	멀티스레드·프로토콜 구현
SFC (Ryu)	15	REST 연동, 플로우 구성
코드 품질	5	주석, 구조, 예외처리
발표·자료	5	시간, 명확성
합계	**100점**	90↑ A / 80↑ B / 70↑ C

제공 자료

- `smart_net_suite_skeleton.py` : 전체 기능 프레임 GUI (Tkinter 기반)

- **16개 항목**을 한 번에 돌려볼 수 있는 **단일 파이썬(Tkinter) GUI 스켈레톤** 코드 기반
  - 실행해서 실험/데모할 수 있고, 각 항목별 버튼/입력/상태표시/멀티스레드/이벤트/임계영역까지 모두 포함하는 완성된 코드 작성
  - Windows/맥/리눅스 동작
  - 관리자 권한은 불필요하지만, 일부 명령은 OS에 따라 분기

- 요구 16개 기능 매핑
  1. **ifconfig로 IP 구성 확인**
     a. “네트워크 진단” 탭의 [IP 구성 확인]
     b. Windows는 `ipconfig`, Unix계는 `ifconfig` 자동 분기
  2. **바이트 정렬 함수**
     a. “도구/변환” 영역의 [바이트 순서 데모]
     b. (host↔network, 16/32/64비트)
  3. **IP 주소 변환 함수**
     a. [inet_pton/ntop 데모]
     b. [hton/ntoh 데모] 버튼
  4. **DNS와 이름 변환**
     a. [DNS 조회]
     b. [역방향 이름 조회]
  5. **Server 상태 확인**
     a. [포트 오픈 여부 검사] (로컬 혹은 임의 호스트:포트)
  6. **netstat -a -n -p tcp | findstr 9000**
     a. [netstat 필터] (Windows는 findstr, Unix는 grep)
  7. **GUI TCP 서버 함수 상태 표시**
     a. “TCP 서버” 탭: 시작/정지, 접속 수, 로그, 스레드/이벤트 상태
  8. **TCP 클라이언트 함수 상태 표시**
     a. “TCP 클라이언트” 탭: 접속/해제, 전송모드 선택, 로그
  9. **소켓 데이터 구조체(버퍼) 상태 표시**
     a. “버퍼/소켓” 탭: SO_SNDBUF/SO_RCVBUF 조회
  10. **네트워크 그림판**

a. “네트워크 그림판” 탭: 드래그로 선 그리기, 서버가 브로드캐스트

11. **고정길이 전송**
    a. 클라이언트 전송모드: FIXED(N바이트 정확히 전송)
12. **가변길이 전송**
    a. 전송모드: VAR(길이프리, ‘\n’ 구분)
13. **고정+가변 전송**
    a. 전송모드: MIX(4바이트 길이프리픽스 + 페이로드)
14. **데이터 전송 후 종료**
    a. [전송 후 종료] 체크 시 send→close
15. **멀티 스레드 동작**
    a. 서버: 클라이언트당 스레드 + 공유 카운터(임계영역 보호)
16. **임계영역/이벤트 연습**
    a. 서버: `threading.Lock()`로 공유카운터 보호, `threading.Event()`로 안전 종료

- AD 과제 스켈러튼 코드

```
# smart_net_suite_skeleton.py
# 통합 네트워크/SFC GUI 스켈레톤 (기능 비워둔 틀)
# - 실행은 되지만 실제 네트워크/REST 동작 없음
# - 각 TODO 지점에 구현을 채워 넣으세요.

import tkinter as tk
from tkinter import ttk, scrolledtext, messagebox

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("스마트 네트워크 서비스 — 스켈레톤")
        self.geometry("1100×720")

        self.server_running = False
        self.client_connected = False

        nb = ttk.Notebook(self); nb.pack(fill="both", expand=True)
        self.pg_diag   = ttk.Frame(nb); nb.add(self.pg_diag,   text="네트워크 진단")
        self.pg_server = ttk.Frame(nb); nb.add(self.pg_server, text="TCP 서버")
        self.pg_client = ttk.Frame(nb); nb.add(self.pg_client, text="TCP 클라이언트")
        self.pg_buf    = ttk.Frame(nb); nb.add(self.pg_buf,    text="버퍼/소켓")
        self.pg_draw   = ttk.Frame(nb); nb.add(self.pg_draw,   text="네트워크 그림판")
        self.pg_sfc    = ttk.Frame(nb); nb.add(self.pg_sfc,    text="Ryu SFC")

        self._build_diag()
        self._build_server()
        self._build_client()
        self._build_buf()
```

```
        self._build_draw()
        self._build_sfc()

    # ---------------- 네트워크 진단 ----------------
    def _build_diag(self):
        left = ttk.Frame(self.pg_diag, padding=8); left.pack(side="left", fill="y")
        right = ttk.Frame(self.pg_diag, padding=8); right.pack(side="right", fill="both", expand=True)

        ttk.Label(left, text="IP 구성 / netstat / 포트 검사").pack(anchor="w")
        ttk.Button(left, text="IP 구성 확인", command=self.do_ipconfig).pack(fill="x", pady=2)
        self.var_netstat = tk.StringVar(value="9000")
        row = ttk.Frame(left); row.pack(fill="x", pady=2)
        ttk.Entry(row, textvariable=self.var_netstat, width=10).pack(side="left")
        ttk.Button(row, text="netstat 필터", command=self.do_netstat).pack(side="left", padx=4)

        row2 = ttk.Frame(left); row2.pack(fill="x", pady=(6,2))
        self.var_host = tk.StringVar(value="127.0.0.1")
        self.var_port = tk.StringVar(value="9000")
        ttk.Entry(row2, textvariable=self.var_host, width=14).pack(side="left")
        ttk.Entry(row2, textvariable=self.var_port, width=6).pack(side="left", padx=4)
        ttk.Button(row2, text="포트 오픈 검사", command=self.do_check_port).pack(side="left", padx=4)

        ttk.Separator(left).pack(fill="x", pady=8)
        ttk.Label(left, text="바이트/주소 변환").pack(anchor="w")
        ttk.Button(left, text="hton/ntoh 데모", command=self.do_hton).pack(fill="x", pady=2)
        self.var_ipv4 = tk.StringVar(value="8.8.8.8")
        self.var_ipv6 = tk.StringVar(value="2001:4860:4860::8888")
        row3 = ttk.Frame(left); row3.pack(fill="x", pady=2)
        ttk.Entry(row3, textvariable=self.var_ipv4, width=18).pack(side="left")
        ttk.Button(row3, text="inet_pton/ntop(IPv4)", command=self.do_inet4).pack(side="left", padx=4)
        row4 = ttk.Frame(left); row4.pack(fill="x", pady=2)
        ttk.Entry(row4, textvariable=self.var_ipv6, width=26).pack(side="left")
        ttk.Button(row4, text="inet_pton/ntop(IPv6)", command=self.do_inet6).pack(side="left", padx=4)

        ttk.Separator(left).pack(fill="x", pady=8)
        ttk.Label(left, text="DNS/이름 변환").pack(anchor="w")
        self.var_dns = tk.StringVar(value="example.com")
        self.var_rev = tk.StringVar(value="8.8.8.8")
        row5 = ttk.Frame(left); row5.pack(fill="x", pady=2)
        ttk.Entry(row5, textvariable=self.var_dns, width=18).pack(side="left")
        ttk.Button(row5, text="DNS 조회", command=self.do_dns).pack(side="left", padx=4)
        row6 = ttk.Frame(left); row6.pack(fill="x", pady=2)
        ttk.Entry(row6, textvariable=self.var_rev, width=18).pack(side="left")
        ttk.Button(row6, text="역방향 조회", command=self.do_reverse).pack(side="left", padx=4)

        self.out_diag = scrolledtext.ScrolledText(right, height=30)
        self.out_diag.pack(fill="both", expand=True)

    def log_diag(self, s): self._append(self.out_diag, s)

    # ---- 진단 스켈레톤 핸들러 (구현 지점) ----
    def do_ipconfig(self):   self._todo("IP 구성 확인 실행", area="diag")
```

```
    def do_netstat(self):    self._todo(f"netstat 필터: {self.var_netstat.get()}", area="diag")
    def do_check_port(self): self._todo(f"포트 검사: {self.var_host.get()}:{self.var_port.get()}", area="diag")
    def do_hton(self):       self._todo("hton/ntoh 데모", area="diag")
    def do_inet4(self):      self._todo(f"inet_pton/ntop IPv4: {self.var_ipv4.get()}", area="diag")
    def do_inet6(self):      self._todo(f"inet_pton/ntop IPv6: {self.var_ipv6.get()}", area="diag")
    def do_dns(self):        self._todo(f"DNS 조회: {self.var_dns.get()}", area="diag")
    def do_reverse(self):    self._todo(f"역방향 조회: {self.var_rev.get()}", area="diag")

    # ---------------- TCP 서버 ----------------
    def _build_server(self):
        top = ttk.Frame(self.pg_server, padding=8); top.pack(fill="x")
        self.var_srv_port = tk.StringVar(value="9000")
        ttk.Label(top, text="포트").pack(side="left")
        ttk.Entry(top, textvariable=self.var_srv_port, width=6).pack(side="left", padx=4)
        self.var_broadcast = tk.BooleanVar(value=True)
        ttk.Checkbutton(top, text="그림판 브로드캐스트", variable=self.var_broadcast).pack(side="left", padx
=6)
        ttk.Button(top, text="서버 시작", command=self.server_start).pack(side="left", padx=4)
        ttk.Button(top, text="서버 정지", command=self.server_stop).pack(side="left", padx=4)

        stat = ttk.Frame(self.pg_server, padding=8); stat.pack(fill="x")
        self.lbl_clients = ttk.Label(stat, text="접속: 0"); self.lbl_clients.pack(side="left")
        self.lbl_counter = ttk.Label(stat, text="카운터: 0"); self.lbl_counter.pack(side="left", padx=12)
        ttk.Button(stat, text="상태 갱신", command=self.server_status).pack(side="left")

        self.out_srv = scrolledtext.ScrolledText(self.pg_server, height=28)
        self.out_srv.pack(fill="both", expand=True)

    def log_srv(self, s): self._append(self.out_srv, s)

    # ---- 서버 스켈레톤 핸들러 (구현 지점) ----
    def server_start(self):
        self.server_running = True
        self.log_srv(f"[서버] 시작 @ {self.var_srv_port.get()} (스켈레톤)")
        # TODO: 소켓 생성/리스닝/스레드 시작

    def server_stop(self):
        self.server_running = False
        self.log_srv("[서버] 정지 요청 (스켈레톤)")
        # TODO: stop event, join

    def server_status(self):
        # TODO: 실제 접속 수/카운터 반영
        self.lbl_clients.config(text="접속: ?")
        self.lbl_counter.config(text="카운터: ?")
        self.log_srv("[서버] 상태 갱신 (스켈레톤)")

    # ---------------- TCP 클라이언트 ----------------
    def _build_client(self):
        top = ttk.Frame(self.pg_client, padding=8); top.pack(fill="x")
        self.var_cli_host = tk.StringVar(value="127.0.0.1")
        self.var_cli_port = tk.StringVar(value="9000")
```

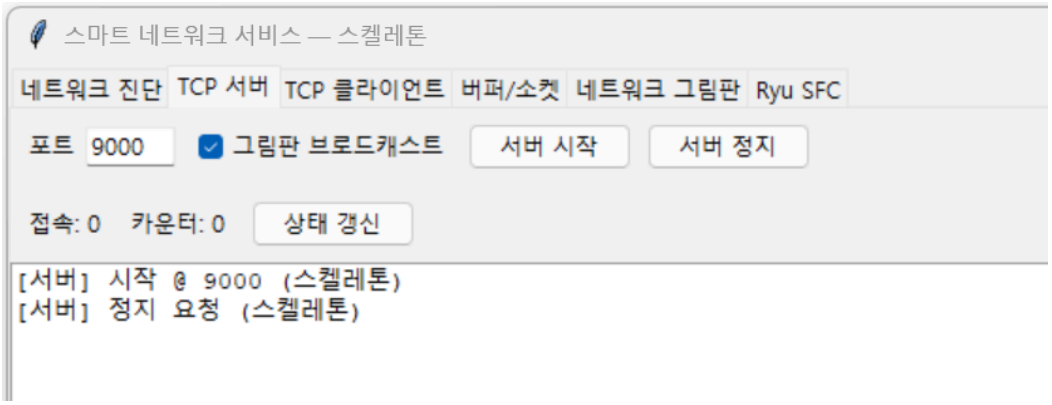
```
        ttk.Label(top, text="호스트").pack(side="left")
        ttk.Entry(top, textvariable=self.var_cli_host, width=16).pack(side="left", padx=4)
        ttk.Label(top, text="포트").pack(side="left")
        ttk.Entry(top, textvariable=self.var_cli_port, width=6).pack(side="left", padx=4)
        ttk.Button(top, text="접속", command=self.cli_connect).pack(side="left", padx=4)
        ttk.Button(top, text="해제", command=self.cli_close).pack(side="left", padx=4)

        opt = ttk.Frame(self.pg_client, padding=8); opt.pack(fill="x")
        self.var_mode = tk.StringVar(value="VAR")
        ttk.Radiobutton(opt, text="VAR(\\n)", variable=self.var_mode, value="VAR").pack(side="left")
        ttk.Radiobutton(opt, text="FIXED(32B)", variable=self.var_mode, value="FIXED").pack(side="left", p
adx=6)
        ttk.Radiobutton(opt, text="MIX(4B len+data)", variable=self.var_mode, value="MIX").pack(side="lef
t", padx=6)
        self.var_after_close = tk.BooleanVar(value=False)
        ttk.Checkbutton(opt, text="전송 후 종료", variable=self.var_after_close).pack(side="left", padx=8)

        msg = ttk.Frame(self.pg_client, padding=8); msg.pack(fill="x")
        self.var_msg = tk.StringVar(value="hello")
        ttk.Entry(msg, textvariable=self.var_msg, width=60).pack(side="left")
        ttk.Button(msg, text="전송", command=self.cli_send).pack(side="left", padx=6)

        self.out_cli = scrolledtext.ScrolledText(self.pg_client, height=28)
        self.out_cli.pack(fill="both", expand=True)

    def log_cli(self, s): self._append(self.out_cli, s)

    # ---- 클라이언트 스켈레톤 핸들러 (구현 지점) ----
    def cli_connect(self):
        self.client_connected = True
        self.log_cli(f"[클라] 연결 시도 → {self.var_cli_host.get()}:{self.var_cli_port.get()} (스켈레톤)")
        # TODO: socket connect + recv 루프

    def cli_close(self):
        self.client_connected = False
        self.log_cli("[클라] 연결 해제 (스켈레톤)")
        # TODO: close

    def cli_send(self):
        self.log_cli(f"[클라] 모드={self.var_mode.get()} 메시지='{self.var_msg.get()}' (스켈레톤)")
        # TODO: VAR/FIXED/MIX 전송 구현

    # ---------------- 버퍼/소켓 ----------------
    def _build_buf(self):
        top = ttk.Frame(self.pg_buf, padding=8); top.pack(fill="x")
        ttk.Button(top, text="클라 소켓 버퍼 조회", command=self.buf_client).pack(side="left", padx=4)
        ttk.Button(top, text="임시 소켓 버퍼 조회", command=self.buf_temp).pack(side="left", padx=4)
        self.out_buf = scrolledtext.ScrolledText(self.pg_buf, height=30)
        self.out_buf.pack(fill="both", expand=True)

    def log_buf(self, s): self._append(self.out_buf, s)
```

```
    # ---- 버퍼 스켈레톤 핸들러 ----
    def buf_client(self):
        self.log_buf("[버퍼] 클라이언트 소켓 버퍼 조회 (스켈레톤)")
        # TODO: getsockopt SO_SNDBUF/SO_RCVBUF

    def buf_temp(self):
        self.log_buf("[버퍼] 임시 소켓 생성 후 버퍼 조회 (스켈레톤)")
        # TODO: socket() 후 옵션 조회

    # ---------------- 네트워크 그림판 ----------------
    def _build_draw(self):
        info = ttk.Frame(self.pg_draw, padding=8); info.pack(fill="x")
        ttk.Label(info, text="그림판 스켈레톤 — 드래그 시 선, (옵션) 네트워크 브로드캐스트").pack(side="left")
        self.canvas = tk.Canvas(self.pg_draw, bg="white", height=520)
        self.canvas.pack(fill="both", expand=True, padx=8, pady=8)
        self.canvas.bind("<ButtonPress-1>", self._draw_start)
        self.canvas.bind("<B1-Motion>", self._draw_move)
        self._last_xy = None

    def _draw_start(self, e): self._last_xy = (e.x, e.y)
    def _draw_move(self, e):
        if not self._last_xy: return
        x1,y1 = self._last_xy; x2,y2 = e.x, e.y
        self.canvas.create_line(x1,y1,x2,y2)  # TODO: 네트워크로 동기화하려면 여기서 송신
        self._last_xy = (x2,y2)

    # ---------------- Ryu SFC (REST) ----------------
    def _build_sfc(self):
        top = ttk.Frame(self.pg_sfc, padding=8); top.pack(fill="x")
        self.var_rest_host = tk.StringVar(value="127.0.0.1")
        self.var_rest_port = tk.StringVar(value="8080")
        self.var_dpid = tk.StringVar(value="1")
        self.var_prio = tk.StringVar(value="100")
        self.var_h1 = tk.StringVar(value="1")
        self.var_fw = tk.StringVar(value="2")
        self.var_nat= tk.StringVar(value="3")
        self.var_h2 = tk.StringVar(value="4")

        ttk.Label(top, text="Ryu").grid(row=0, column=0, sticky="e")
        ttk.Entry(top, textvariable=self.var_rest_host, width=14).grid(row=0, column=1)
        ttk.Label(top, text=":").grid(row=0, column=2)
        ttk.Entry(top, textvariable=self.var_rest_port, width=6).grid(row=0, column=3, padx=4)
        ttk.Label(top, text="DPID").grid(row=0, column=4, sticky="e")
        ttk.Entry(top, textvariable=self.var_dpid, width=6).grid(row=0, column=5)
        ttk.Label(top, text="prio").grid(row=0, column=6, sticky="e")
        ttk.Entry(top, textvariable=self.var_prio, width=6).grid(row=0, column=7)

        ports = ttk.Frame(self.pg_sfc, padding=8); ports.pack(fill="x")
        for i,(lab,var) in enumerate([("h1",self.var_h1),("fw",self.var_fw),("nat",self.var_nat),("h2",self.var_h
2)]):
            ttk.Label(ports, text=lab).grid(row=0, column=i*2)
            ttk.Entry(ports, textvariable=var, width=6).grid(row=0, column=i*2+1, padx=4)
```

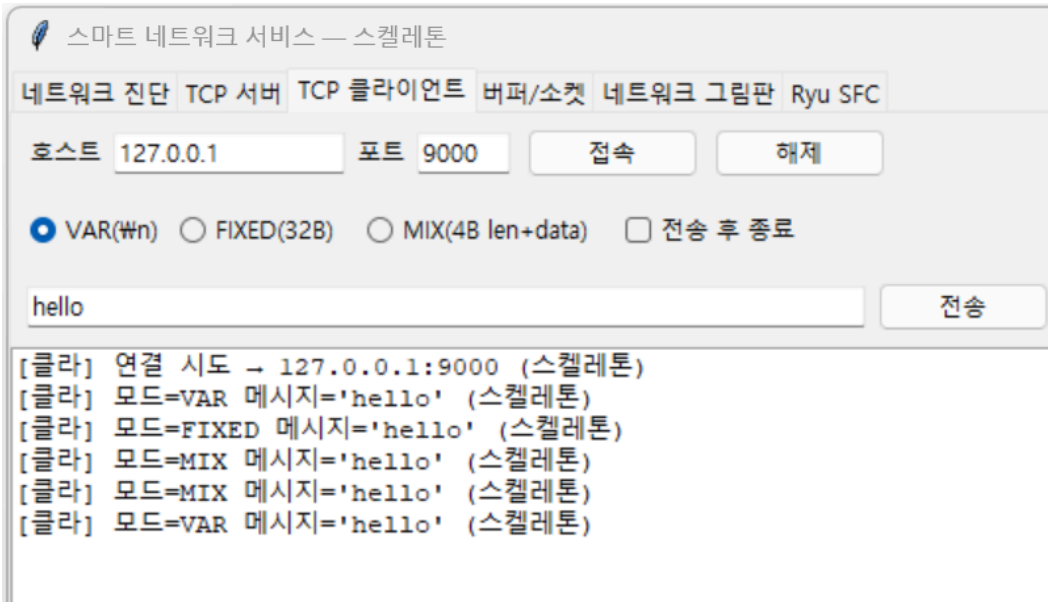
```
        btns = ttk.Frame(self.pg_sfc, padding=8); btns.pack(fill="x")
        ttk.Button(btns, text="SFC 설치", command=self.sfc_install).pack(side="left", padx=4)
        ttk.Button(btns, text="바이패스", command=self.sfc_bypass).pack(side="left", padx=4)
        ttk.Button(btns, text="플로우 조회", command=self.sfc_dump).pack(side="left", padx=4)
        ttk.Button(btns, text="플로우 삭제", command=self.sfc_clear).pack(side="left", padx=4)

        self.out_sfc = scrolledtext.ScrolledText(self.pg_sfc, height=24)
        self.out_sfc.pack(fill="both", expand=True, padx=8, pady=8)

    def log_sfc(self, s): self._append(self.out_sfc, s)

    # ---- SFC 스켈레톤 핸들러 ----
    def sfc_install(self): self._todo("SFC 설치 (REST POST /stats/flowentry/add)", area="sfc")
    def sfc_bypass(self):  self._todo("바이패스 설치", area="sfc")
    def sfc_dump(self):    self._todo("플로우 조회 (GET /stats/flow/<dpid>)", area="sfc")
    def sfc_clear(self):   self._todo("플로우 삭제 (DELETE /stats/flowentry/clear/<dpid>)", area="sfc")

    # ---------------- 공용 ----------------
    def _append(self, widget, text):
        widget.insert("end", text + "\n")
        widget.see("end")

    def _todo(self, msg, area):
        target = {"diag": self.out_diag, "sfc": self.out_sfc}.get(area, None)
        if target: self._append(target, f"[TODO] {msg}")
        else: messagebox.showinfo("TODO", msg)

if __name__ == "__main__":
    App().mainloop()
```

- 네트워크 진단 화면

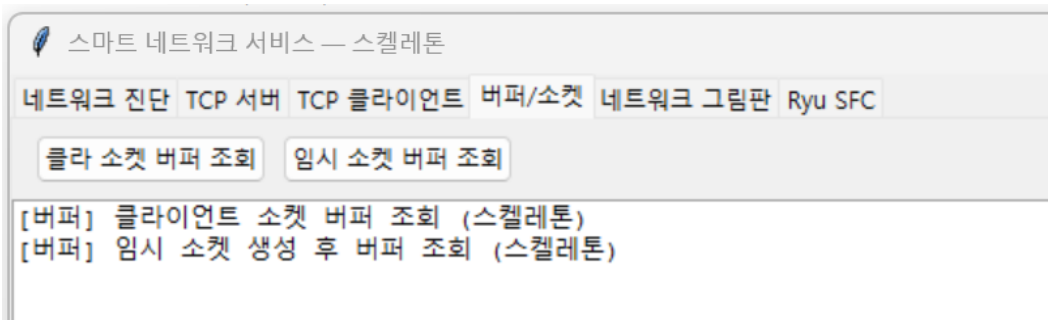
- TCP 서버 화면



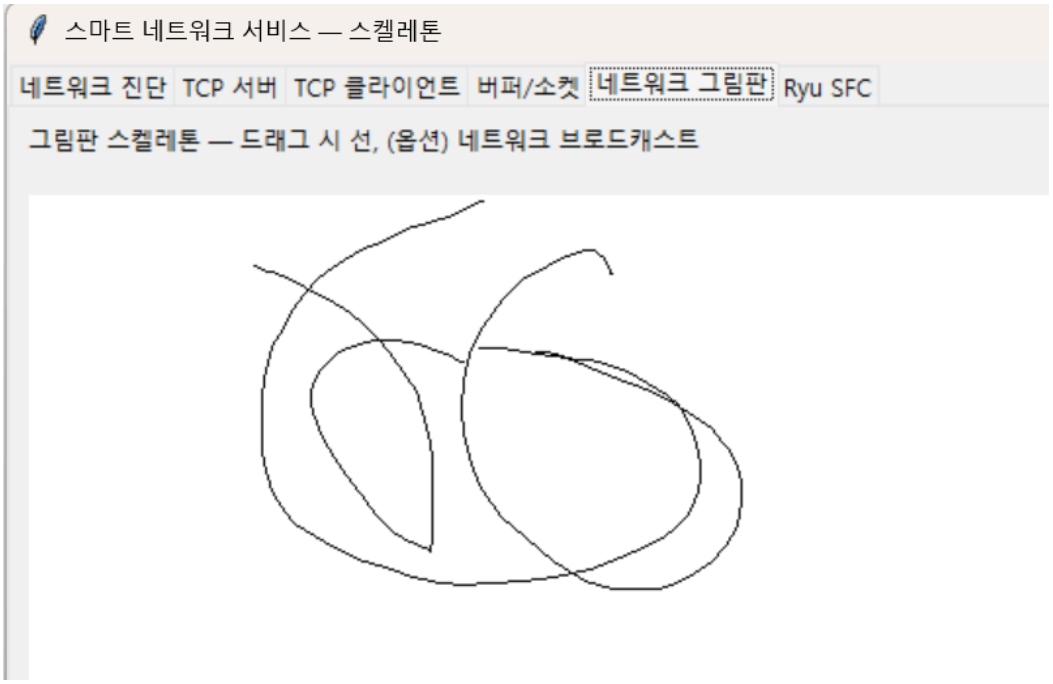
- TCP 클라이언트 화면



- 버퍼/소켓



- 네트워크 그림판



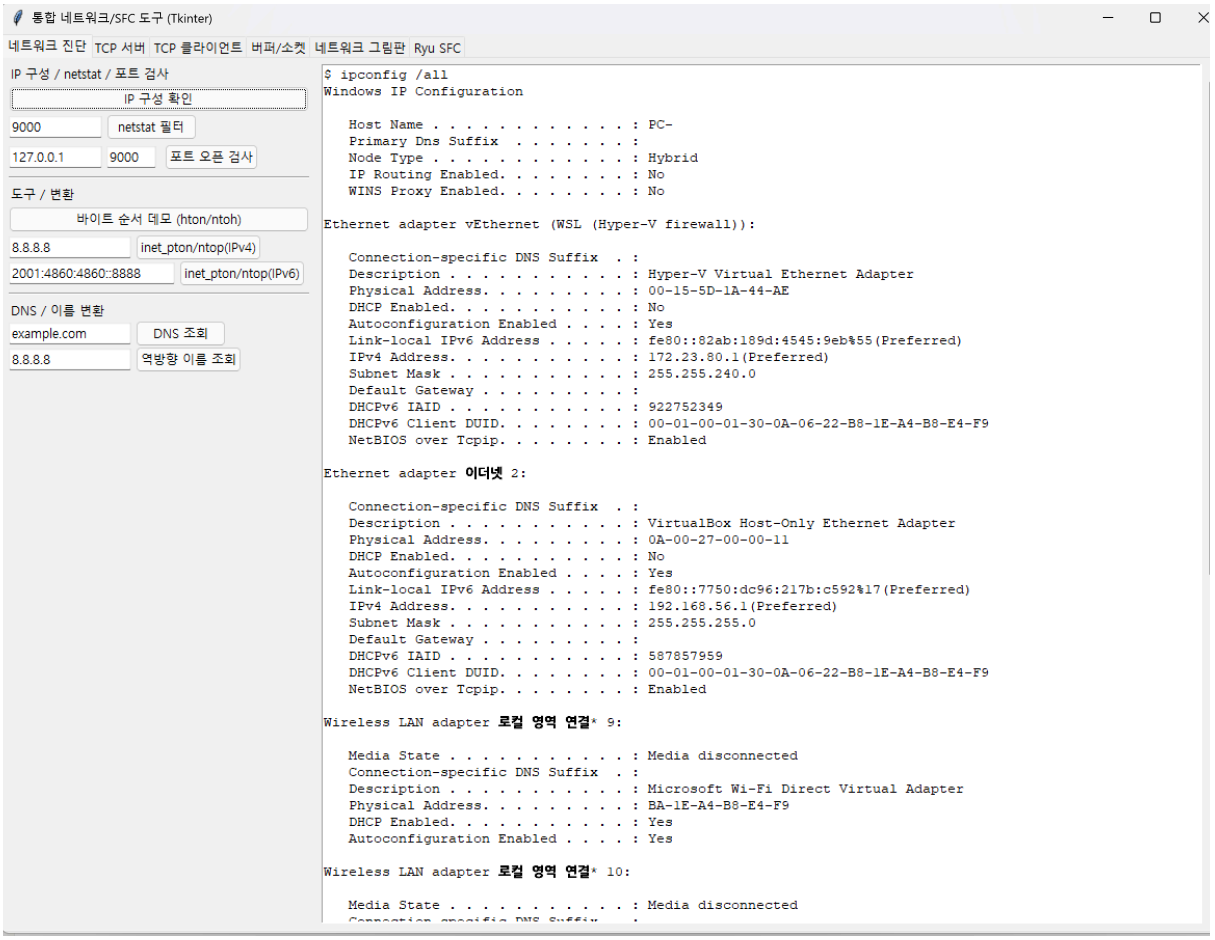
- Ryu SFC

과제 채점 포인트

- **발표 데모 루틴(4분)**
  1. 서버 시작(9000) → 클라이언트 접속 → VAR/FIXED/MIX 전송 → “전송 후 종료” 체크로 종료 시나리오
  2. 네트워크 진단 탭에서 IP 구성/넷스탯/포트 검사 → DNS/역방향/inet_pton/ntop/hton/ntoh 순으로 “재료” 시연
  3. 다른 노트북(또는 같은 PC에서 두 번째 클라)로 그림판 동시 그리기(브로드캐스트)
  4. 버퍼 탭에서 SO_SNDBUF/RCVBUF 조회로 “소켓 구조체 상태” 언급
  5. 서버 탭의 “상태 갱신”으로 접속 수/공유 카운터(임계영역) 설명, 종료 시 `Event` 로 안전 정지 설명

- **리포트/코드 정리**:
  - README에 16개 요구사항을 위 매핑처럼 **체크리스트 표**로 정리
  - 스크린샷(각 탭) + 실행 방법(OS별) + 테스트 절차(포트 검사, 전송 모드 비교)
  - 발표자료 첫 페이지에 **팀원/성명/학번** 반드시 기재

- 실행 화면
  - IP 구성 버튼 실행 후



- netstat 필터 버튼 실행

```
   DNS Servers . . . . . . . . . . . : 210.123.32.13
                                       164.124.101.2
   NetBIOS over Tcpip. . . . . . . . : Enabled

Ethernet adapter Bluetooth 네트워크 연결:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Bluetooth Device (Personal Area Network)
   Physical Address. . . . . . . . . : B8-1E-A4-B8-E4-FA
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
$ netstat -a -n -p tcp | findstr 9000
```