# UNIVERSIDAD NACIONAL DE INGENIERÍA

## SÍLABO

### INFORMACIÓN GENERAL

| | |
|---|---|
| ASIGNATURA | :COMPUTACIÓN CENTRADA EN REDES |
| DOCENTE | :GERSON GARRIDO |
| HORAS POR SEMANA | :06 (TEORÍA: 04, LABORATORIO: 02) |

### SUMILLA

El curso es de naturaleza teórico-práctica y explica cómo funciona Internet y cómo se construye software que depende de ella. Se organiza en cinco partes: **fundamentos** (arquitectura de Internet, retardo y pérdida, capas y programación con sockets); **redes locales e IP** (enlace, Ethernet, VLANs, direccionamiento, forwarding, NAT e IPv6); **routing** (algoritmos de routing, OSPF, BGP, políticas y seguridad de BGP); **transporte y aplicaciones** (entrega confiable, TCP, control de congestión, DNS, HTTP, TLS 1.3, QUIC y CDNs); y **redes modernas** (data centers, SDN, Wi-Fi, 5G y seguridad de redes). El laboratorio y un proyecto que crece durante todo el semestre se desarrollan en C, con routers FRRouting virtualizados mediante Containerlab, en un entorno común que corre en la computadora de cada estudiante.

### COMPETENCIAS

- Explica la arquitectura de Internet, sus capas y sus principios de diseño, y estima el retardo, la pérdida y el throughput de un camino de red.
- Escribe programas cliente-servidor en C con la API de sockets y analiza su tráfico con herramientas de captura.
- Diseña el direccionamiento de una red y configura switching, VLANs, forwarding y NAT, explicando el recorrido de un paquete salto por salto.
- Explica los algoritmos de routing y configura OSPF y BGP con políticas entre sistemas autónomos (*autonomous systems*).
- Implementa en C un protocolo de transporte confiable con ventana deslizante (*sliding window*) y control de congestión, y evalúa su desempeño con mediciones.
- Explica cómo funcionan DNS, HTTP, TLS 1.3 y QUIC, y diagnostica fallas de aplicaciones usando capturas de tráfico.
- Describe las redes de data center, SDN, Wi-Fi y 5G, identifica ataques comunes a las redes y aplica defensas como firewalls, VPNs y RPKI.

## UNIDADES DE APRENDIZAJE

| Unidad | Contenido | Semanas | Kurose y Ross |
|---|---|---|---|
| **I** | Fundamentos | 1–2 | Cap. 1, 2.1, 2.6, 4.5 |
| **II** | Redes locales e IP | 3–4 | Cap. 6.1–6.4, 4.1–4.3, 5.6 |
| **III** | Routing | 5–8 | Cap. 5.1–5.4 |
| **IV** | Transporte y aplicaciones | 9–13 | Cap. 3, 2.2, 2.4, 2.5, 8.2, 8.3, 8.6 |
| **V** | Redes modernas | 14–16 | Cap. 4.4, 5.5, 6.5, 6.6, 7, 8.1, 8.7, 8.9 |

Cada semana tiene, además de las secciones del libro, lecturas gratuitas en línea que cubren los mismos temas: el libro abierto del curso CS168 de la Universidad de California en Berkeley, el libro de Peterson y Davie y los RFC de cada protocolo.

## PROGRAMACIÓN SEMANAL

Cada semana de clases comprende dos sesiones de teoría de 2 horas (A y B) y una sesión de laboratorio de 2 horas. Cada capítulo de los apuntes indica una lectura previa obligatoria de 15 a 20 minutos, lo que se explica en clase y lo que queda para repasar después. Las secciones indicadas del libro de Kurose y Ross acompañan cada sesión. Las sesiones marcadas con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

| Sem. | Sesión A | Sesión B | Laboratorio |
|---|---|---|---|
| 1 | Presentación del curso. Qué es Internet: bordes, núcleo e ISPs (sec. 1.1–1.3) | Retardo, pérdida y throughput. Capas y encapsulamiento (sec. 1.4–1.5) | Entorno: Docker, Wireshark y `traceroute`. *P0 publicado* |
| 2 | Principios de diseño. Historia: de ARPANET a ATM y Frame Relay (*) (sec. 1.7, 4.5) | Programación con sockets en C: TCP y UDP (sec. 2.1, 2.6) | Un cliente y un servidor en C |
| 3 | Capa física en lo esencial. Enlace: framing y detección de errores (sec. 1.2.2, 6.1–6.2) | Acceso múltiple. Ethernet, ARP, switches y VLANs (sec. 6.3–6.4) | Capturar ARP y configurar VLANs. *Entrega P0, publicado P1* |
| 4 | IP: datagramas, CIDR y forwarding. Dentro de un router (sec. 4.1–4.3.2) | ICMP, DHCP, NAT e IPv6 (sec. 4.3.2–4.3.4, 5.6) | Subredes y NAT con Containerlab |
| 5 | Routing: estado de enlace y vector de distancias (sec. 5.1–5.2) | OSPF y routing jerárquico (sec. 5.3) | OSPF con FRRouting: una falla de enlace en vivo |
| 6 | BGP I: sistemas autónomos, eBGP e iBGP, atributos (sec. 5.4.1–5.4.3) | BGP II: clientes, proveedores, peers y políticas (sec. 5.4.4–5.4.6) | Conectar sistemas autónomos con políticas. *Entrega P1, publicado P2* |
| 7 | Los problemas de BGP: hijacks, route leaks y RPKI (sec. 5.4.5, RFC 6480) | IXPs y estructura real de Internet (sec. 1.3.3). Repaso | Provocar un route hijack y defenderse |
| 8 | **Examen parcial** | | |
| 9 | Transporte: multiplexación, UDP y entrega confiable (sec. 3.1–3.4.1) | Ventana deslizante: Go-Back-N y Selective Repeat (sec. 3.4.2–3.4.4) | Hackathon: conectar el Internet de la clase. *Entrega P2, publicado P3* |
| 10 | TCP: segmentos, conexiones y retransmisión (sec. 3.5.1–3.5.4) | TCP: control de flujo y recuperación rápida (sec. 3.5.4–3.5.6) | Leer TCP en capturas, sobre un enlace con pérdidas |

| Sem. | Sesión A | Sesión B | Laboratorio |
|---|---|---|---|
| 11 | Control de congestión: principios, AIMD y Reno (sec. 3.6–3.7.1) | CUBIC, BBR, bufferbloat y equidad (sec. 3.7.2–3.7.4) | Reno y CUBIC bajo distintos enlaces y buffers. *Entrega P3, publicado P4* |
| 12 | DNS: jerarquía, resolución y seguridad (sec. 2.4) | HTTP/1.1, HTTP/2 y caché (sec. 2.2.1–2.2.6) | DNS paso a paso y HTTP por dentro |
| 13 | Criptografía esencial y TLS 1.3 (sec. 8.2, 8.3, 8.6) | QUIC y HTTP/3. CDNs y streaming (sec. 2.2.7, 2.5, 3.8) | Un handshake TLS y HTTP/2 frente a HTTP/3. *Entrega P4, publicado P5* |
| 14 | Data centers: topologías Clos, ECMP y overlays. SDN (sec. 4.4, 5.5, 6.5.2, 6.6) | Redes para IA (*). Wi-Fi y 5G (sec. 7.1–7.5) | Una red leaf-spine con balanceo de carga |
| 15 | Seguridad de redes: ataques, firewalls, IPsec y VPNs (sec. 1.6, 8.1, 8.7, 8.9) | Presentación de proyectos | WireGuard y nftables. *Entrega P5* |
| 16 | **Examen final** | | |

## METODOLOGÍA

- **Teoría:** clases expositivas con discusión de casos, ejercicios resueltos en clase y demostraciones en vivo sobre redes virtualizadas. El estudiante lee las secciones indicadas antes de cada sesión.
- **Laboratorio:** prácticas guiadas en un entorno común para todos los estudiantes, donde configuran routers, capturan tráfico, provocan fallas y las diagnostican. No requiere equipos de red físicos.
- **Proyecto:** un solo proyecto que crece durante el semestre en seis entregas, cada una con pruebas automáticas.
- Los términos técnicos se usan en inglés cuando así se dicen y así aparecen en la bibliografía y en la documentación (router, socket, handshake). Los que de verdad se usan en español (red, enlace, capa, paquete) se presentan en español, con su equivalente en inglés la primera vez.

**Proyecto: Qhapaq Ñan, un pequeño Internet**

| N.º | Entrega | Lo que agrega | Semanas |
|---|---|---|---|
| P0 | Calentamiento | Cliente y servidor con sockets en C, y una versión propia de `traceroute` | 1–3 |
| P1 | Tu red | Direccionamiento, VLANs, NAT y OSPF de un sistema autónomo que sobrevive a la caída de un enlace | 3–6 |
| P2 | Qhapaq Ñan | BGP con clientes, proveedores y peers, políticas correctas y defensa ante un hijack | 6–9 |
| P3 | `rimay`: entrega confiable | Protocolo con ventana deslizante sobre UDP, que funciona con pérdida, duplicación y reordenamiento | 9–11 |
| P4 | `rimay`: congestión | Control de congestión, recuperación rápida y un informe con mediciones de equidad | 11–13 |
| P5 | `wasi` | Servidor HTTP sobre `rimay`, publicado en la red con un nombre en DNS, donde una descarga termina bien aunque se corte un enlace con otro camino disponible | 13–15 |

## POLÍTICAS

- **Integridad académica:** se puede discutir ideas, pero el código y las configuraciones entregados deben ser propios. Copiar soluciones publicadas en internet cuenta como plagio.
- **Defensa oral:** cualquier integrante de un equipo puede tener que explicar, en el laboratorio, cualquier parte de lo entregado.

### Uso de herramientas

| Herramienta | ¿Se permite? |
|---|---|
| Mensajes del compilador, sanitizers, Valgrind y depuradores como `gdb` | Sí, en todo el curso. |
| Documentación oficial (páginas `man`, RFC, FRRouting, Containerlab, Wireshark) y búsquedas en la web | Sí. Lo que se adapte de una fuente se cita. |
| Autocompletado del editor que sugiere nombres, firmas o el resto de una línea, sin generar funciones ni soluciones enteras | Sí. |
| Correctores de ortografía, traductores y herramientas de accesibilidad | Sí. |
| Formateadores y linters | Sí. |
| IA generativa para pedir explicaciones de conceptos, mensajes de error o salidas de comandos, sin que escriba código ni configuraciones | Sí, fuera de los exámenes, y declarando su uso. |
| IA generativa que escribe código, configuraciones de routers o partes del informe | No, en ningún trabajo del curso: proyecto, laboratorios ni exámenes. |

El uso permitido de IA generativa se declara en el `README.md` de la entrega: qué herramienta, para qué y qué se hizo con lo que dio. Las demás herramientas no se declaran, salvo las fuentes adaptadas, que se citan. En cualquier entrega se puede pedir al estudiante que explique su trabajo en forma oral.

## BIBLIOGRAFÍA

### Texto principal

1. Kurose, J. F. y Ross, K. W. (2025). *Computer Networking: A Top-Down Approach* (9.ª ed.). Pearson. Sitio del libro: `https://gaia.cs.umass.edu/kurose_ross/`

### Textos complementarios

2. Peterson, L. L. y Davie, B. S. *Computer Networks: A Systems Approach*. Disponible gratuitamente en línea: `https://book.systemsapproach.org/`
3. Hall, B. *Beej's Guide to Network Programming*. Disponible gratuitamente en línea: `https://beej.us/guide/bgnet/`

### Recursos en línea

- Libro abierto del curso CS168, Universidad de California en Berkeley: `https://textbook.cs168.io/`
- Especificaciones de los protocolos (RFC): `https://www.rfc-editor.org/`
- Documentación de FRRouting: `https://docs.frrouting.org/`

- Documentación de Containerlab: `https://containerlab.dev/`
- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.