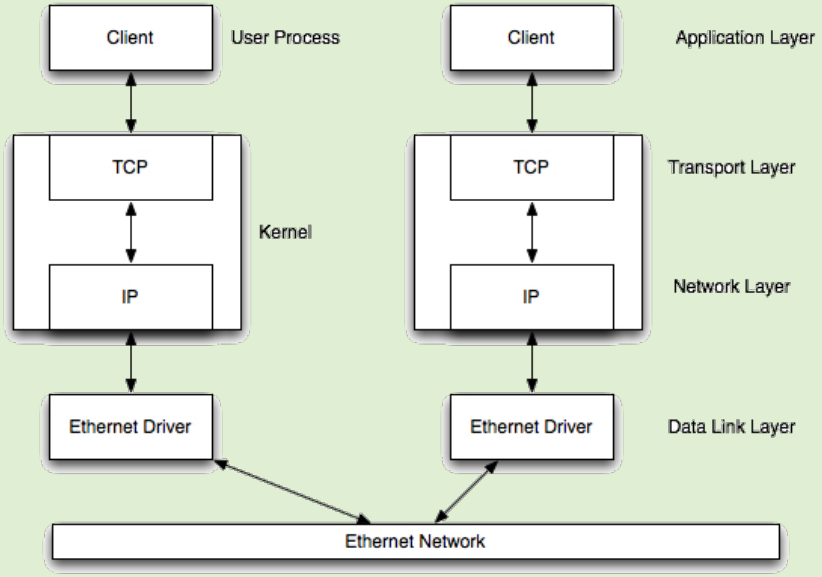
Taller de Programación en Redes
Stack TCP/IP - Sockets

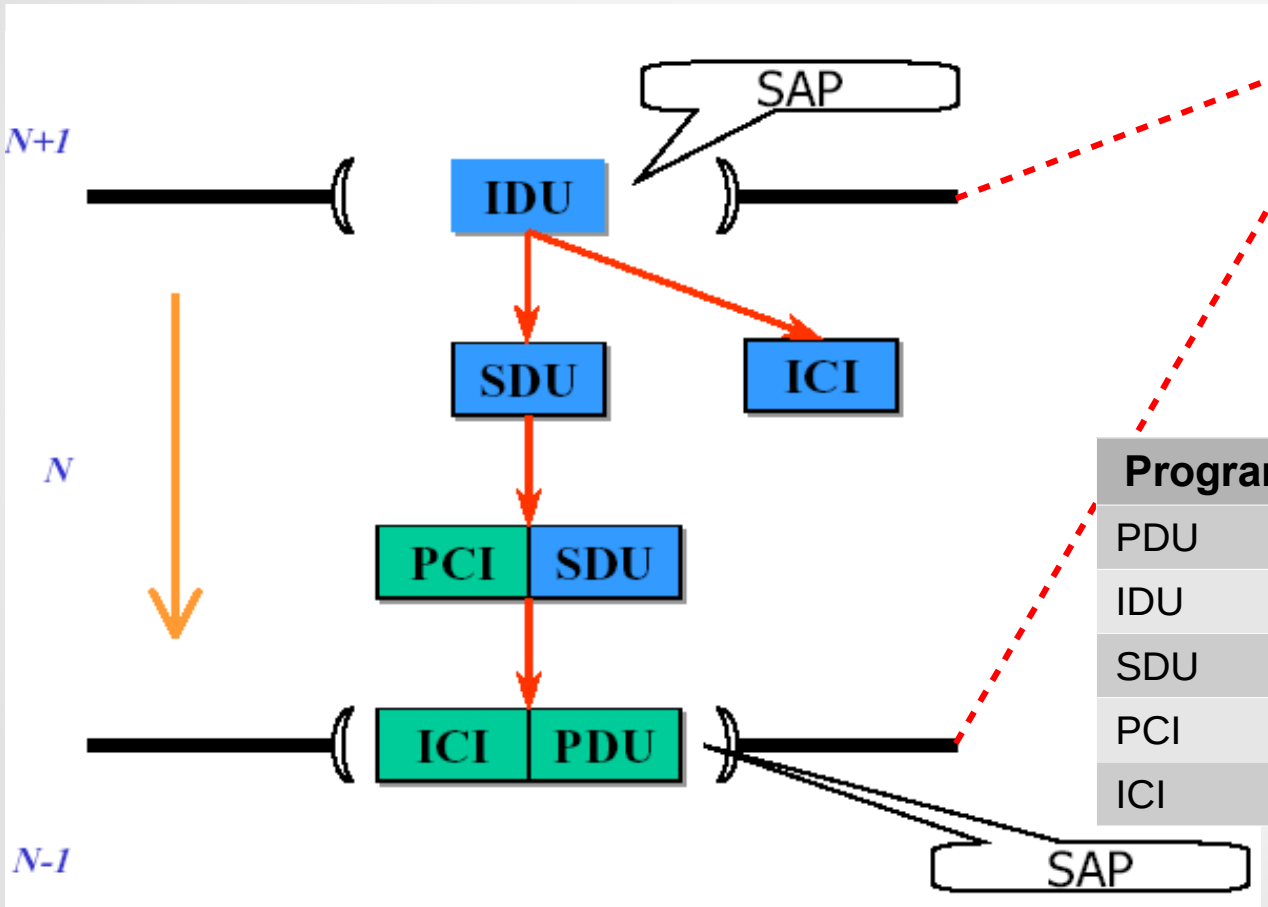


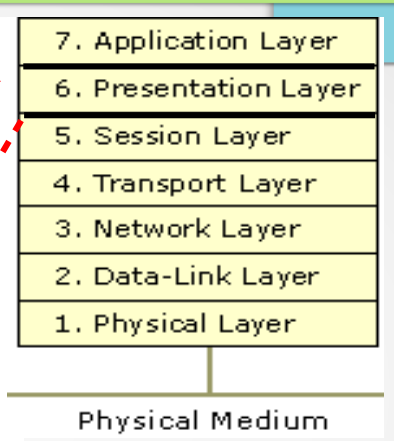
Lic. en Sistemas de Información - Universidad Nacional de Luján
Dr. Gabriel Tolosa – tolosoft@unlu.edu.ar
Lic. Marcelo Fernández – fernandezm@unlu.edu.ar
Clase 1 - Febrero 2018

Introducción - TCP/IP Sockets API

- Introducción
- Objetivos de diseño de la interfaz de sockets

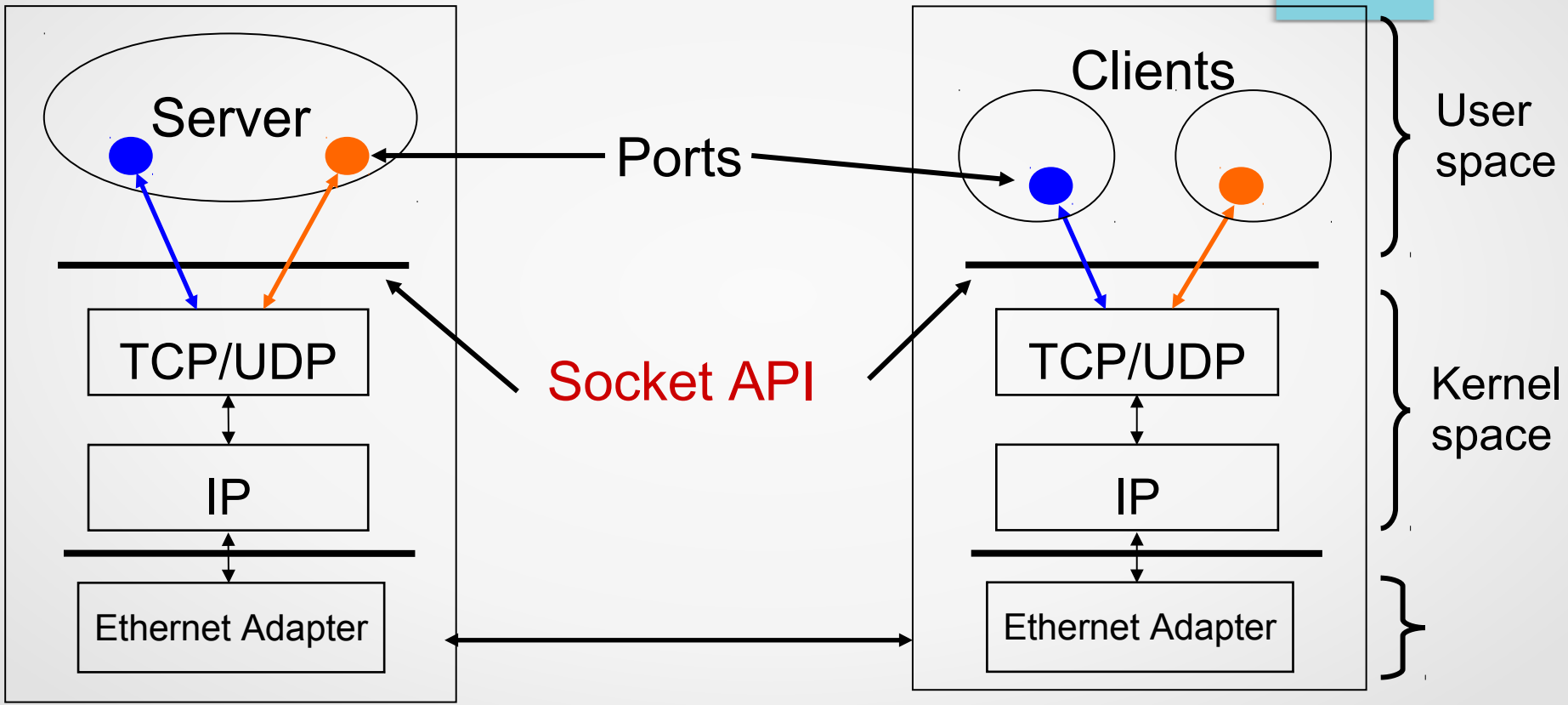
Modelo OSI – Interfaz Vertical



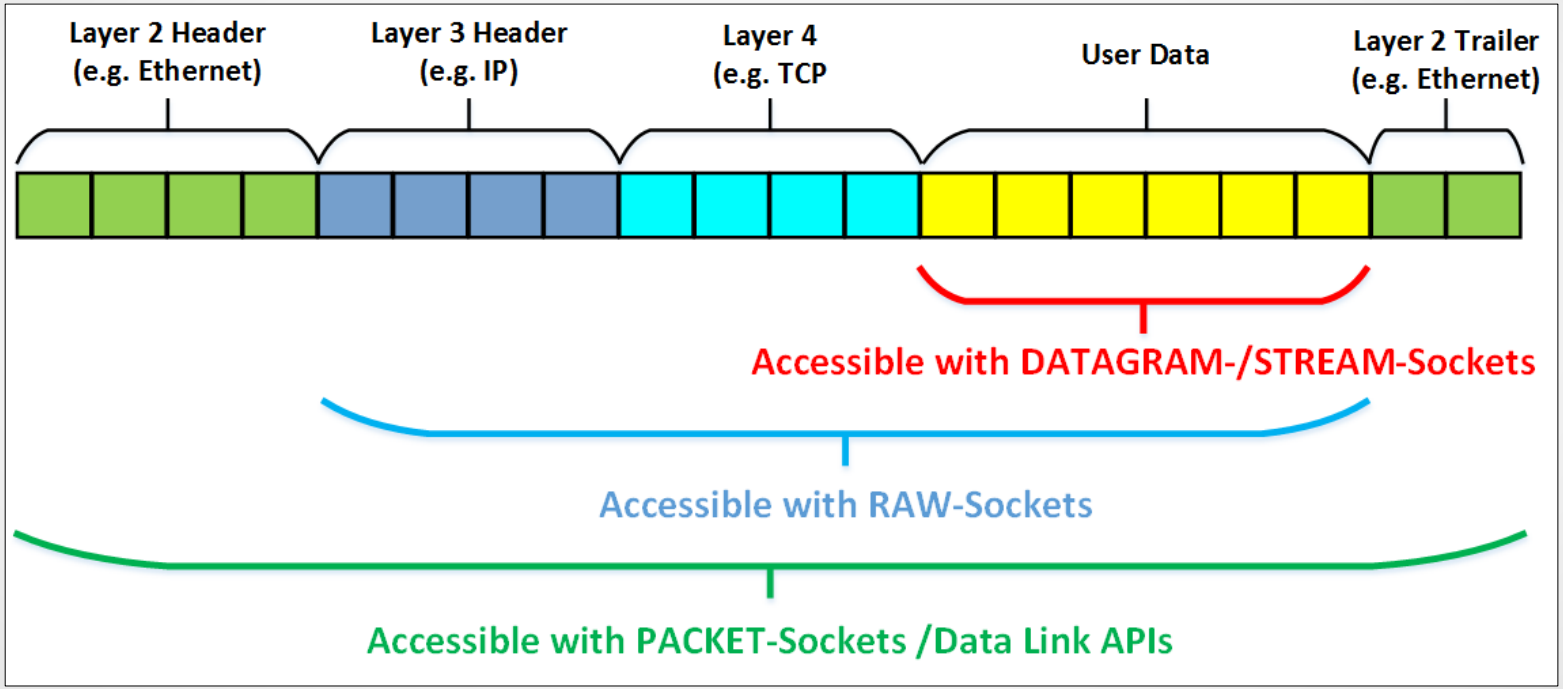


Programación de Red en el Stack TCP/IP	
PDU	Protocol Data Unit
IDU	Interface Data Unit
SDU	Service Data Unit
PCI	Protocol Control Information
ICI	Interface Control Information

Entornos de ejecución – Kernelspace/Userspace



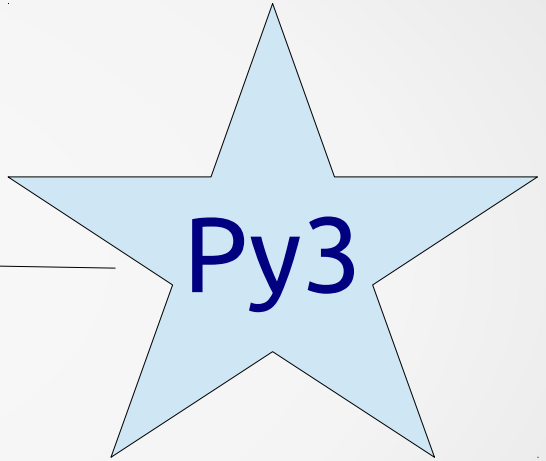
Entornos de ejecución – APIs de programación



* Tomado de http://tuprints.ulb.tu-darmstadt.de/6243/

Python – Algunas características

- Gratis Y Libre
  - Y Open Source, todo por el mismo precio: cero
- Maduro (+25 años)
  - Diseño elegante y robusto
  - Pero evoluciona
- Fácil de aprender
  - Se lee como pseudo-código
  - Sintaxis sencilla, lenguaje muy ortogonal
- Extremadamente portable
  - Unix, Windows, Mac, Android, BeOS, Win/CE
  - DOS, OS/2, Amiga, VMS, Cray...



* Tomado de slides de "Introducción a Python" por Facundo Batista

Python – Propiedades del Lenguaje

- Compila a bytecode interpretado
  - La compilación es implícita y automática
  - Tipado dinámico, pero fuerte
- Multi-paradigma
  - Todo son objetos
  - Pero puede usarse de manera procedural
- Módulos, clases, funciones, generadores
- Manejo moderno de errores
  - Por excepciones
  - Muy útil detalle de error

* Tomado de slides de "Introducción a Python" por Facundo Batista

Python – Propiedades del Lenguaje (2)

- Tipos de datos de alto nivel
  - Enteros sin límites, strings, flotantes, complejos
  - Listas, diccionarios, conjuntos
- Intérprete interactivo
  - Clave en el bajo conteo de bugs
  - Acelera sorprendentemente el tiempo de desarrollo
  - Permite explorar, probar e incluso ver la doc
- Viene con las baterías incluidas
  - Extensa biblioteca estándar
  - Clave en la productividad de Python

* Tomado de slides de "Introducción a Python" por Facundo Batista

Python – Baterías incluidas

- La Biblioteca Estándar ayuda con...
  - Servicios del sistema, fecha y hora, subprocesos, sockets, internacionalización y localización, base de datos, threads, formatos zip, bzip2, gzip, tar, expresiones regulares, XML (DOM y SAX), Unicode, SGML, HTML, XHTML, XML-RPC (cliente y servidor), email, manejo asincrónico de sockets, clientes HTTP, FTP, SMTP, NNTP, POP3, IMAP4, servidores HTTP, SMTP, herramientas MIME, interfaz con el garbage collector, serializador y deserializador de objetos, debugger, profiler, random, curses, logging, compilador, decompilador, CSV, análisis lexicográfico, interfaz gráfica incorporada, matemática real y compleja, criptografía (MD5 y SHA), introspección, unit testing, doc testing, etc., etc...

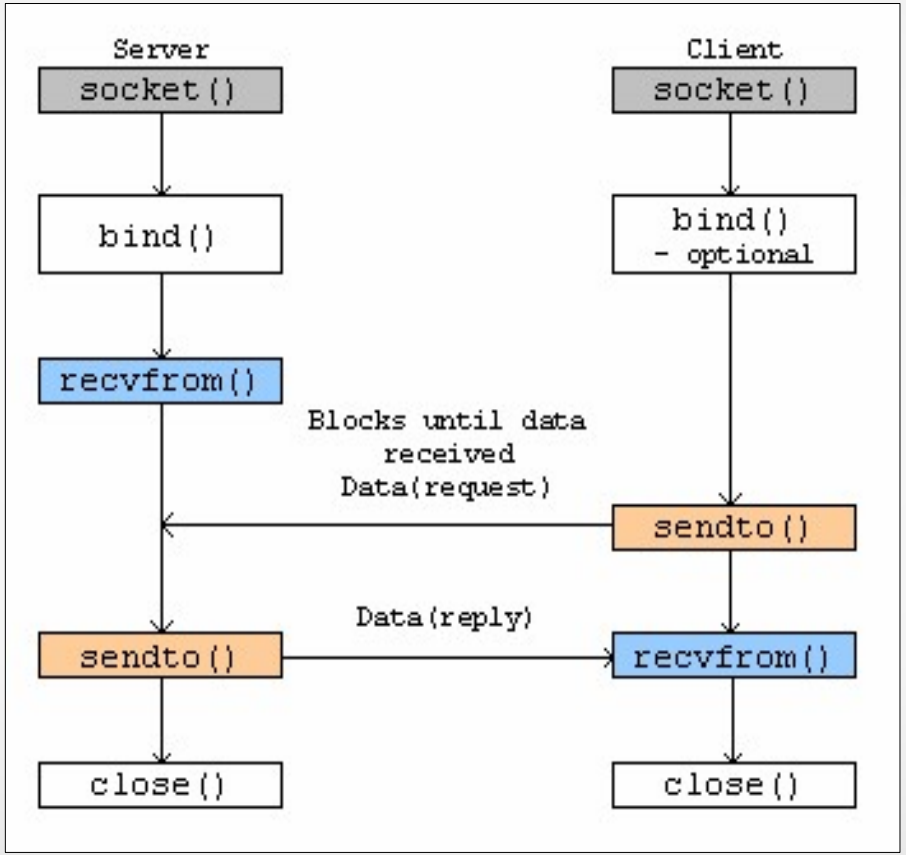
* Tomado de slides de "Introducción a Python" por Facundo Batista

Python – Baterías incluidas (2)

- Bases de datos
  - MySQL, PostgresSQL, MS SQL, MongoDB, Reddis
- Interfaces gráficas
  - Qt, GTK, win32, wxWidgets, Cairo, Kivy
- Frameworks Web
  - Django, Flask, Pyramid, Zope, Plone, webpy
- Y un montón más de temas...
  - Pillow: para trabajar con imágenes
  - PyGame: juegos, presentaciones, gráficos
  - SymPy: matemática simbólica
  - Numpy: calculos de alta performance
  - ...

* Tomado de slides de "Introducción a Python" por Facundo Batista

Socket API para UDP – Esquema de Trabajo



Taller 1 - Interacción Cliente/Servidor en UDP

Actividades del Taller 1

- Descargar echo UDP server/client
  - `http://www.marcelofernandez.info/sockets/udp_echo_server.py`
  - `http://www.marcelofernandez.info/sockets/udp_echo_client.py`
- Analizar código
- Ejecutar
- Ver comandos del SO (netstat, ps)
- Ver cómo se extrae info de los headers del protocolo UDP

UDP Echo Server

```
#!/usr/bin/env python
import socket

HOST  = 'localhost'  # IP o Hostname donde escucha
PORT  = 3500         # Puerto de escucha
bytes = 1024         # Cantidad máxima de bytes a aceptar

# Se crea un socket de tipo Internet (AF_INET) sobre UDP (SOCK_DGRAM)
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.bind((HOST, PORT))  # Indicamos al socket la direccion IP y Port de escucha

while True:
    # Se obtienen los datos desde el sock cliente
    data,address = s.recvfrom(bytes)
    print 'Conexion desde: %s:%i' % (address[0], address[1])
    if data:
        s.sendto(data, address)  # Enviamos el echo al cliente
```

UDP Echo Client

```
#!/usr/bin/env python
import socket

HOST = 'localhost'
PORT = 3500
bytes = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.sendto('Taller I: Echo sobre UDP', (HOST, PORT))
data = s.recvfrom(bytes)
s.close()
print '-->', data[0]
```

Estructuras y constantes

Dominio: Indica el dominio de comunicación que se desea utilizar:

- **PF_INET** / **AF_INET** Protocolos de Internet versión 4.
- **PF_INET6** / **AF_INET6** Protocolos de Internet versión 6.
- **PF_IPX** Protocolos IPX (Novell).
- **PF_APPLETALK** Protocolos Appletalk.
- **PF_UNIX** o **PF_LOCAL** Comunicación local entre procesos
- ...

Estructuras y constantes

Tipo: Tipo de comunicación deseada.

- **SOCK_STREAM** Conexión bidireccional confiable con el flujo de datos ordenados ==> Protocolo TCP
- **SOCK_DGRAM** Mensajes no confiables, sin conexión, con una longitud máxima fija ==> Protocolo UDP
- **SOCK_SEQPACKET** Conexión bidireccional confiable con el flujo de datos ordenados y datagramas de longitud máxima fija ==> SPX
- **SOCK_RAW** Utilizar directamente Protocolo IP, sin protocolo de transporte

Taller 1 - Interacción Cliente/Servidor en UDP

Prácticas:

- Modificar para hacer cliente/servidor al mismo tiempo con un parámetro
- Sacar localhost y probar con las IPs del vecino
- Realizar captura y analizar tráfico
- Enunciados para que trabajen:

  1) Implementar el protocolo DayTime, cliente y servidor, con UDP: http://ietf.org/rfc/rfc867.txt

  2) Aplicación para calcular el RTT en la red: http://ietf.org/rfc/rfc2681.txt

  3) Chat punto a punto sobre UDP con compresión y control de errores mediante zlib. https://docs.python.org/2.7/library/zlib.html