

# Proyecto de realidad aumentada colaborativa

Sistema de gestión de taxis

**Grupo G1.1**

Ángel Fernández Moreno  
Jose Antonio García Mecinas  
Marcos Vives Del Sol

# Resumen

## ❖ Montado en C99

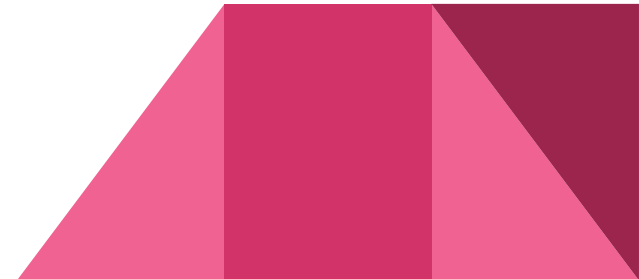
- Tanto cliente como servidor
- Sin usar librerías externas, sólo POSIX
- Multiplataforma (Windows, Linux, OSX, BSD...)

## ❖ Protocolo basado en UDP

- Paquetes de datos representados por estructuras de tamaño fijo

## ❖ Servidor monohilo

- Un mismo hilo procesa todos y cada de uno de los paquetes



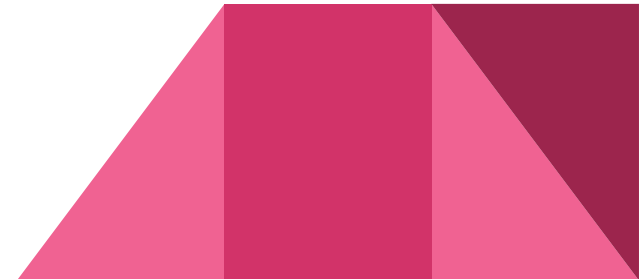
# Problemas

## Problema: se pierden paquetes críticos

Es UDP, es posible perder paquetes como los de LOGIN (petición de aceptación desde el cliente al servidor), bloqueando cliente y/o servidor esperando un paquete que nunca llegará.

## Solución: handshakes y reenvío

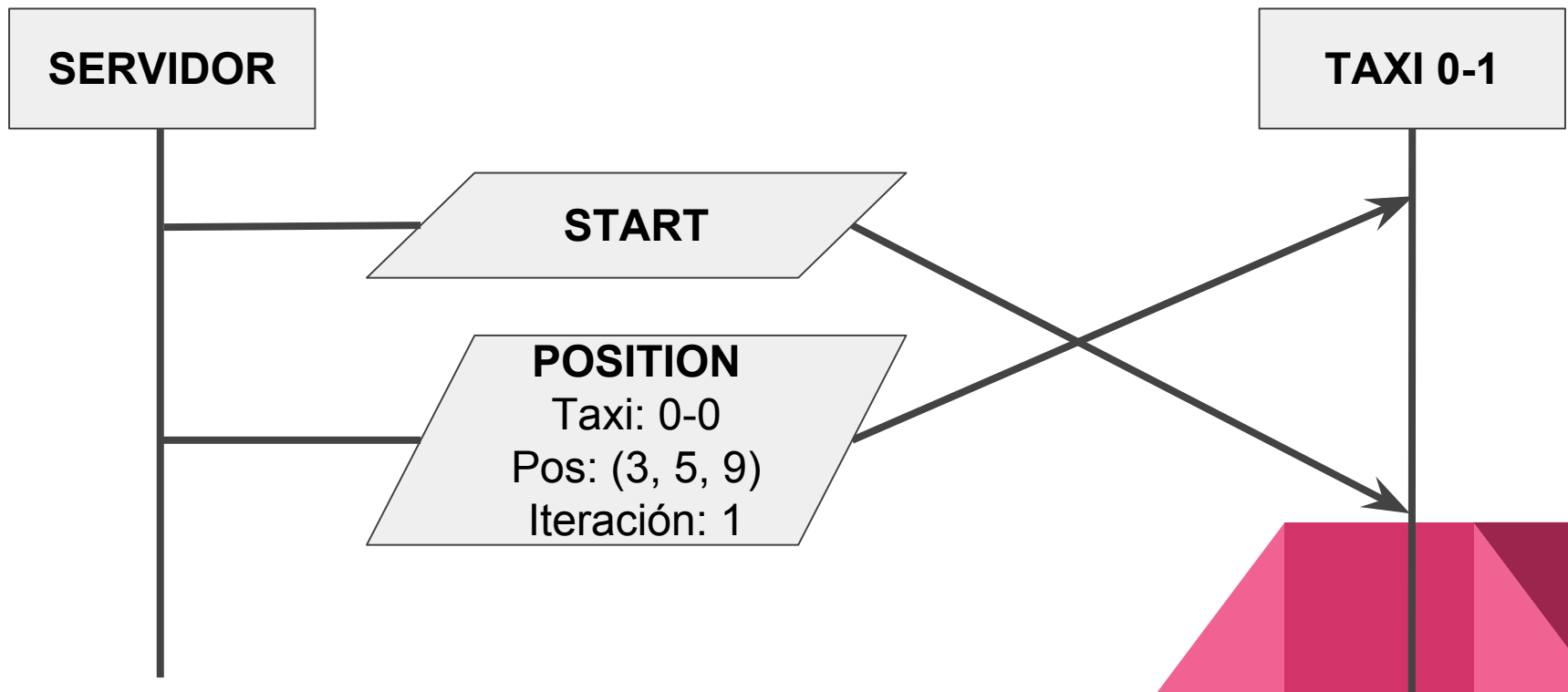
- 0x00: LOGIN
- 0x01: ACCEPT
- 0x02: ACCACK ← nuevo
- 0x03: START
- 0x04: POSITION
- 0x05: POSACK
- 0x06: STATS
- 0x07: STATAACK ← nuevo



# Problemas

## Problema: adelantamiento de paquetes

El cliente no sabe cómo responder a un paquete si supuestamente aún no hemos empezado.



# Problemas

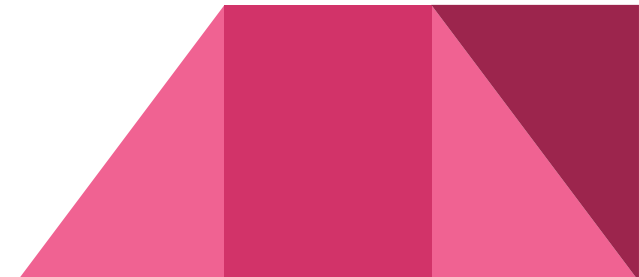
## Solución:

Se modifica la abstracción `connection` y se añade una lista enlazada de paquetes aún no procesados.

Para cada paquete, el programa informa a la abstracción `connection` si:

- El paquete se ha procesado: `connection_accept`
- No se puede procesar aún: `connection_save`
- No es válido o es irrelevante: `connection_discard`

Por cada `connection_accept` se revisa el contenido entero de la cola, si hay algo en ella.



# Problemas

## **Problema: Los clientes se desincronizan.**

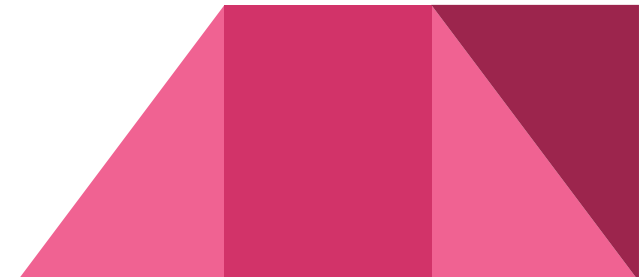
Unos van más adelantados que otros, y hacer que el servidor los sincronice rompe el modelo P2P. No podemos esperar a recibir siempre todos los paquetes para avanzar, algunos inevitablemente se perderán.

## **Solución A - secuestro de paquetes:**

Sólo devolvemos ACKs para las posiciones cuya iteración es igual o anterior a la nuestra. Las posteriores no son procesadas inmediatamente: se guardan en la cola hasta que alcancemos dicha iteración.

## **Problema de la solución A:**

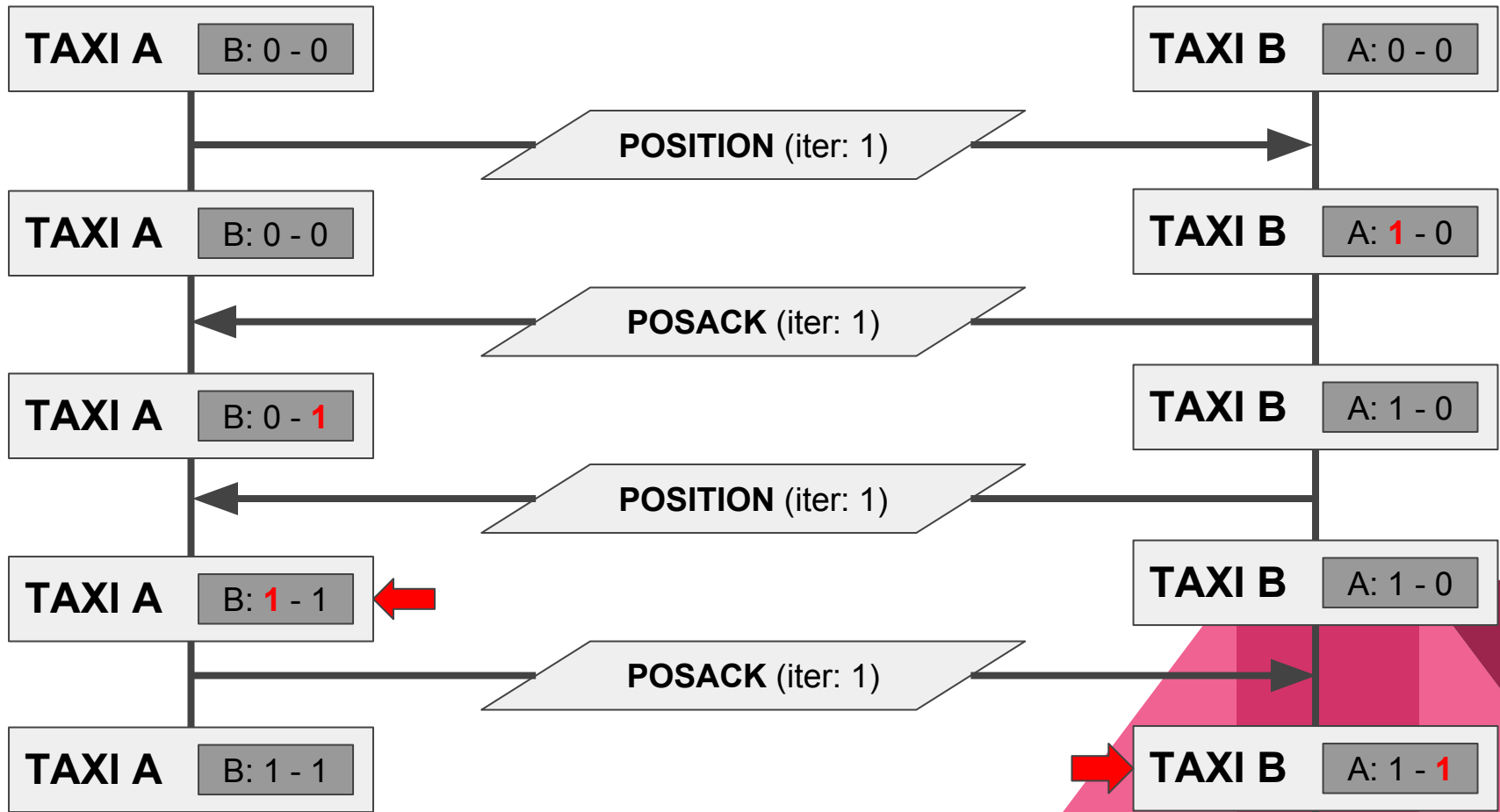
Interbloqueos. Un paquete perdido puede dar al traste al encadenarse retraso tras retraso.



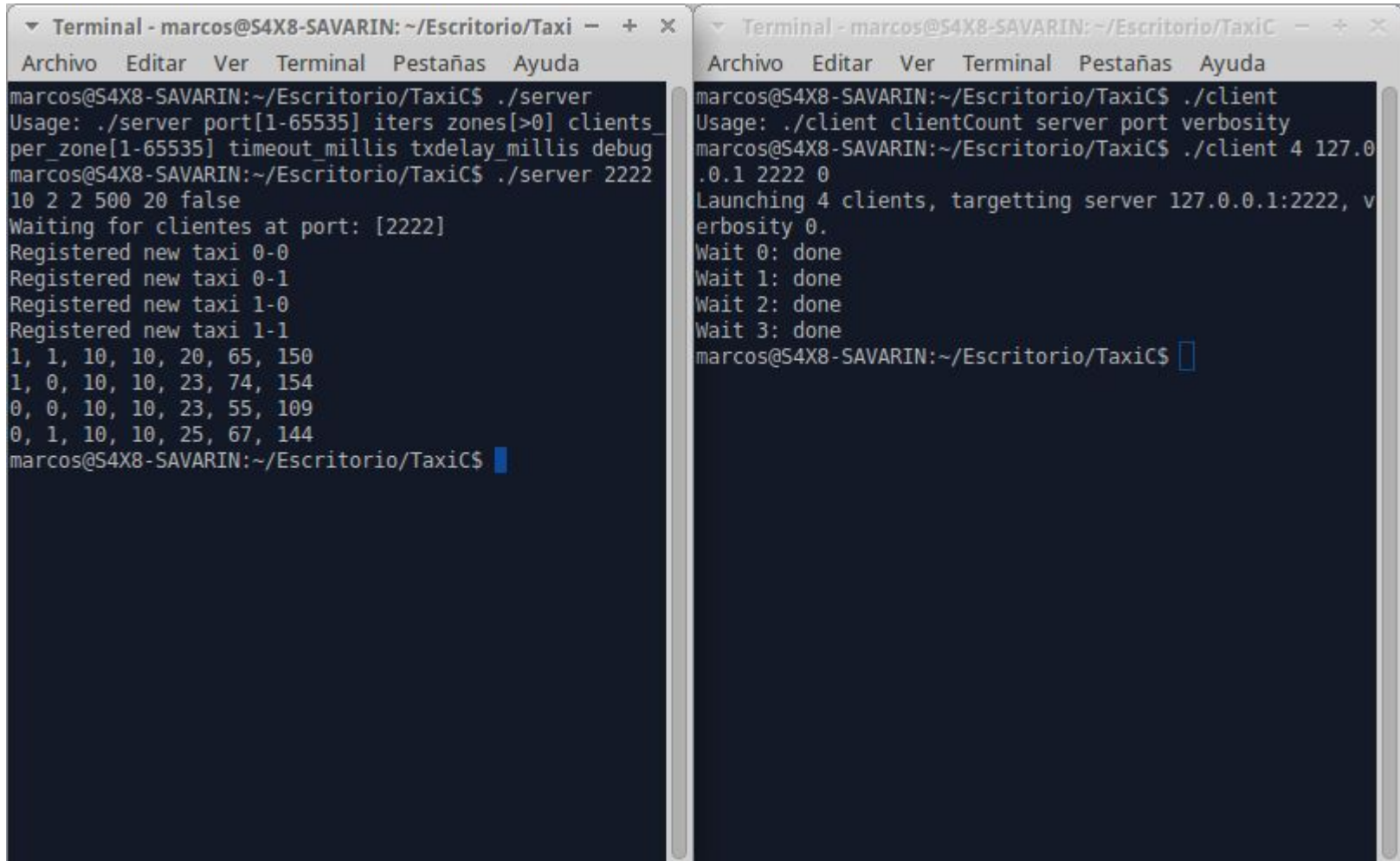
# Problemas

## Solución B - pensamiento en colmena:

Cada cliente ahora no sólo guarda su iteración. Conoce la del resto de vecinos.



# Funcionamiento - lanzamiento



```
Terminal - marcos@S4X8-SAVARIN: ~/Escritorio/Taxi
Archivo Editar Ver Terminal Pestañas Ayuda
marcos@S4X8-SAVARIN:~/Escritorio/Taxi$ ./server
Usage: ./server port[1-65535] iters zones[>0] clients_per_zone[1-65535] timeout_millis txdelay_millis debug
marcos@S4X8-SAVARIN:~/Escritorio/Taxi$ ./server 2222 10 2 2 500 20 false
Waiting for clientes at port: [2222]
Registered new taxi 0-0
Registered new taxi 0-1
Registered new taxi 1-0
Registered new taxi 1-1
1, 1, 10, 10, 20, 65, 150
1, 0, 10, 10, 23, 74, 154
0, 0, 10, 10, 23, 55, 109
0, 1, 10, 10, 25, 67, 144
marcos@S4X8-SAVARIN:~/Escritorio/Taxi$

Terminal - marcos@S4X8-SAVARIN: ~/Escritorio/TaxiC
Archivo Editar Ver Terminal Pestañas Ayuda
marcos@S4X8-SAVARIN:~/Escritorio/TaxiC$ ./client
Usage: ./client clientCount server port verbosity
marcos@S4X8-SAVARIN:~/Escritorio/TaxiC$ ./client 4 127.0.0.1 2222 0
Launching 4 clients, targetting server 127.0.0.1:2222, verbosity 0.
Wait 0: done
Wait 1: done
Wait 2: done
Wait 3: done
marcos@S4X8-SAVARIN:~/Escritorio/TaxiC$
```

**Servidor - debug deshabilitado**

**Master Client - verbosity 0**



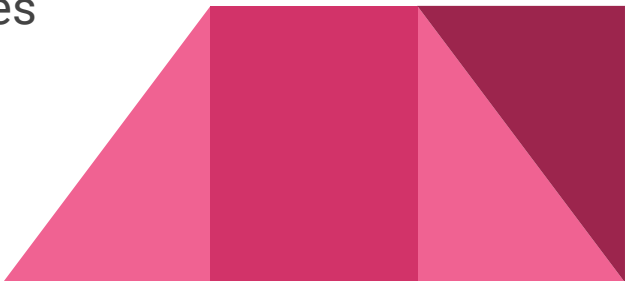
# Funcionamiento - lanzamiento

El debugging se puede **deshabilitar en tiempo de ejecución, sin necesidad de recompilar.**

Niveles del servidor:

- OFF: sólo muestra estadísticas
- ON: muestra estadísticas y flujo de paquetes

Niveles del cliente:

- 0: No muestra nada
  - 1: Muestra errores
  - 1: Muestra errores y información de estado
  - 2: Muestra errores, información, timeouts y conocimiento de los vecinos
  - 3: Igual que 2, pero además muestra flujo de paquetes
- 

# Funcionamiento - lanzamiento

[illegible]

## Servidor - debug habilitado

## Master Client - verbosity 3

# Funcionamiento - Wireshark

Capturing from eth0 [Wireshark 1.12.7 (Git Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: `udp.srcport == 32323 or udp.dstport == 32323` Expression... Clear

No.	Time	Source	Destination	Protocol	Length
4610	3.089199000	192.168.0.243	192.168.0.240	UDP	60
4611	3.089207000	192.168.0.240	192.168.0.243	UDP	46
4639	6.008386000	192.168.0.240	192.168.0.243	UDP	47
4640	6.008396000	192.168.0.240	192.168.0.243	UDP	47
4641	6.008445000	192.168.0.240	192.168.0.243	UDP	47

Frame 4610: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface  
Ethernet II, Src: a6:9d:09:bb:5d:d9 (a6:9d:09:bb:5d:d9), Dst: Micro-St 1c:d1:29 (d  
Internet Protocol Version 4, Src: 192.168.0.243 (192.168.0.243), Dst: 192.168.0.24  
User Datagram Protocol, Src Port: 32323 (32323), Dst Port: 47977 (47977)  
Data (16 bytes)  
Data: 0100cb01da010a00204e000010270000  
[Length: 16]

0000 d8 cb 8a 1c d1 29 a6 9d 09 bb 5d d9 08 00 45 00 .....). . .]...E  
0010 00 2c 00 00 40 00 40 11 b7 8d c0 a8 00 f3 c0 a8 ....@.@. ....  
0020 00 f0 7e 43 bb 69 00 18 62 64 01 00 cb 01 da 01 ...~C.i... bd.....

01 00 CB 01 DA 01 0A 00 20 4E 00 00 10 27 00 00

Data (data.data), 16 bytes Packets: 57303 · Displ... Profile: Default

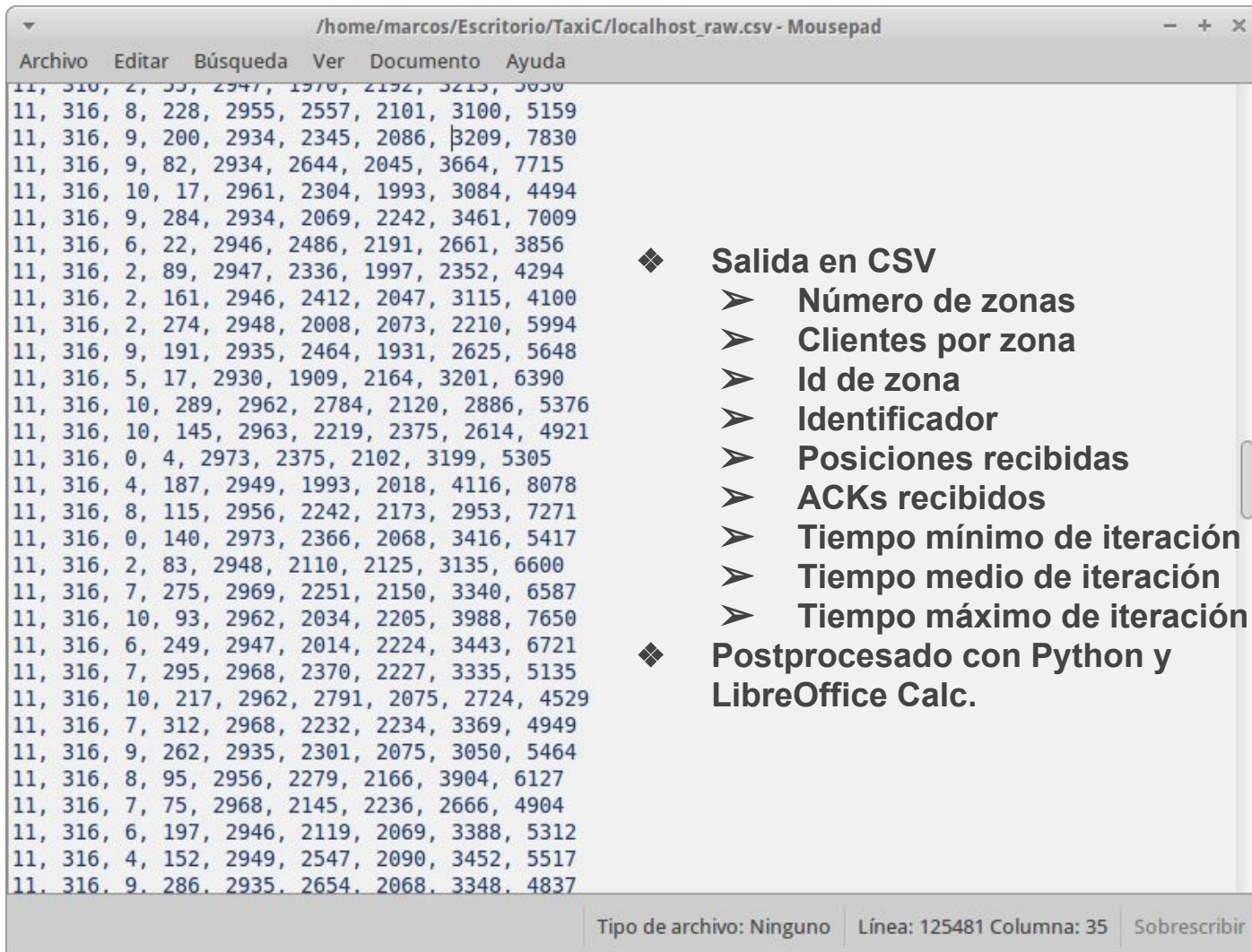
Byte 0 a 0x01:  
→ Tipo ACCEPT

A continuación  
datos del ACCEPT:

- ❖ Zona: 0x00
- ❖ ID: 0x01CB
- ❖ Vecinos: 0x01DA
- ❖ Ticks:
  - 0x000A
  - 10 iteraciones
- ❖ Timeout:
  - 0x00004E20
  - 20000 ms
- ❖ Retraso envío máx.:
  - 0x00002710
  - 0~10000 ms



# Funcionamiento - salida



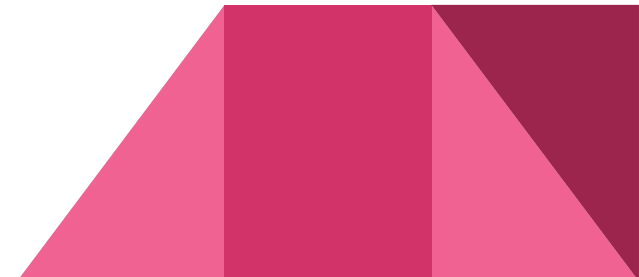
The screenshot shows a text editor window titled "/home/marcos/Escritorio/TaxiC/localhost\_raw.csv - Mousepad". The window contains a list of CSV rows, each starting with "11, 316," followed by various numerical values. The status bar at the bottom indicates "Tipo de archivo: Ninguno", "Línea: 125481", "Columna: 35", and a "Sobrescribir" button.

Archivo	Editar	Búsqueda	Ver	Documento	Ayuda
11, 316, 2, 55, 2947, 1970, 2192, 3215, 3030					
11, 316, 8, 228, 2955, 2557, 2101, 3100, 5159					
11, 316, 9, 200, 2934, 2345, 2086, 3209, 7830					
11, 316, 9, 82, 2934, 2644, 2045, 3664, 7715					
11, 316, 10, 17, 2961, 2304, 1993, 3084, 4494					
11, 316, 9, 284, 2934, 2069, 2242, 3461, 7009					
11, 316, 6, 22, 2946, 2486, 2191, 2661, 3856					
11, 316, 2, 89, 2947, 2336, 1997, 2352, 4294					
11, 316, 2, 161, 2946, 2412, 2047, 3115, 4100					
11, 316, 2, 274, 2948, 2008, 2073, 2210, 5994					
11, 316, 9, 191, 2935, 2464, 1931, 2625, 5648					
11, 316, 5, 17, 2930, 1909, 2164, 3201, 6390					
11, 316, 10, 289, 2962, 2784, 2120, 2886, 5376					
11, 316, 10, 145, 2963, 2219, 2375, 2614, 4921					
11, 316, 0, 4, 2973, 2375, 2102, 3199, 5305					
11, 316, 4, 187, 2949, 1993, 2018, 4116, 8078					
11, 316, 8, 115, 2956, 2242, 2173, 2953, 7271					
11, 316, 0, 140, 2973, 2366, 2068, 3416, 5417					
11, 316, 2, 83, 2948, 2110, 2125, 3135, 6600					
11, 316, 7, 275, 2969, 2251, 2150, 3340, 6587					
11, 316, 10, 93, 2962, 2034, 2205, 3988, 7650					
11, 316, 6, 249, 2947, 2014, 2224, 3443, 6721					
11, 316, 7, 295, 2968, 2370, 2227, 3335, 5135					
11, 316, 10, 217, 2962, 2791, 2075, 2724, 4529					
11, 316, 7, 312, 2968, 2232, 2234, 3369, 4949					
11, 316, 9, 262, 2935, 2301, 2075, 3050, 5464					
11, 316, 8, 95, 2956, 2279, 2166, 3904, 6127					
11, 316, 7, 75, 2968, 2145, 2236, 2666, 4904					
11, 316, 6, 197, 2946, 2119, 2069, 3388, 5312					
11, 316, 4, 152, 2949, 2547, 2090, 3452, 5517					
11, 316, 9, 286, 2935, 2654, 2068, 3348, 4837					

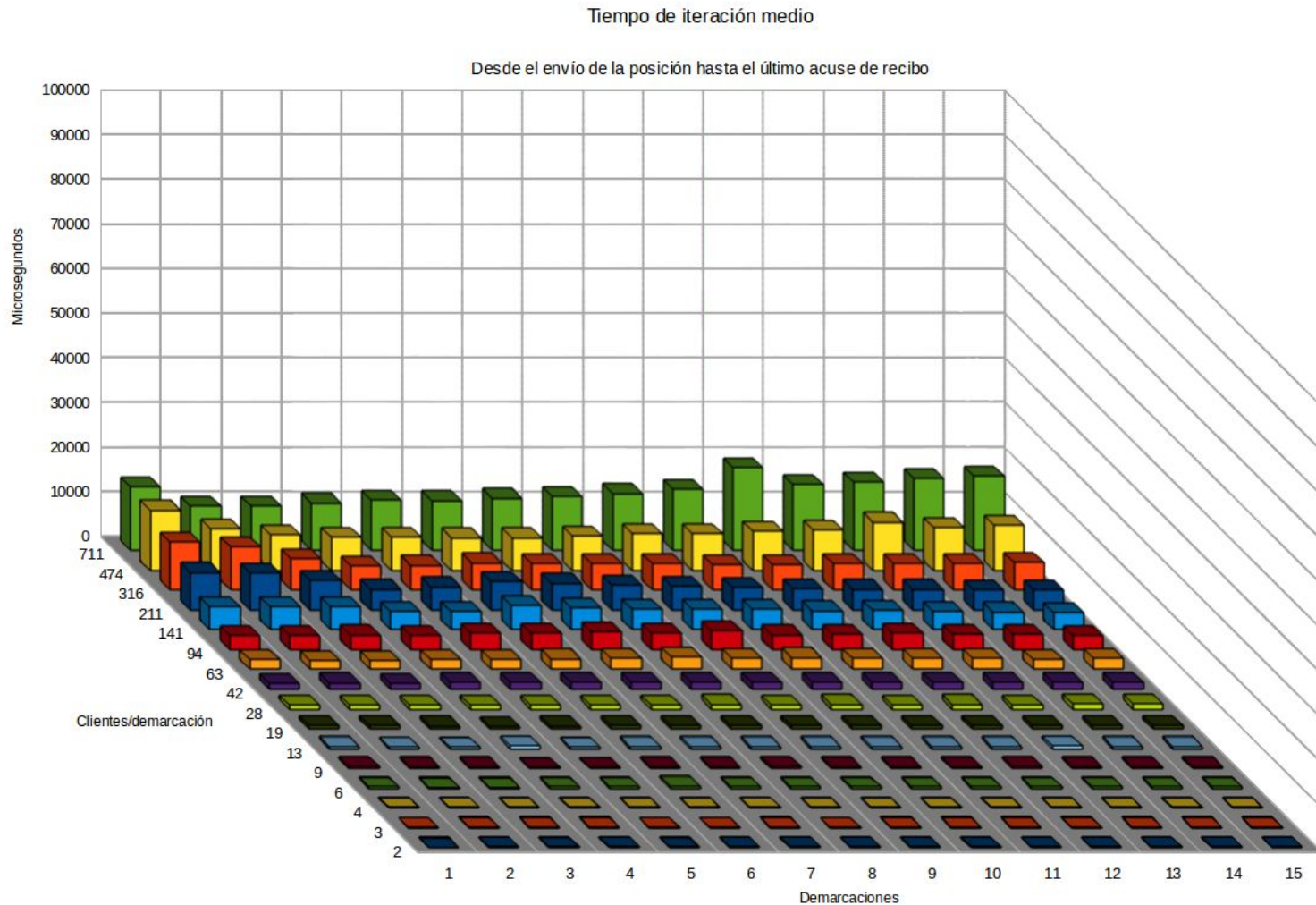
- ❖ **Salida en CSV**
  - Número de zonas
  - Clientes por zona
  - Id de zona
  - Identificador
  - Posiciones recibidas
  - ACKs recibidos
  - Tiempo mínimo de iteración
  - Tiempo medio de iteración
  - Tiempo máximo de iteración
- ❖ **Postprocesado con Python y LibreOffice Calc.**

# Pruebas - configuración

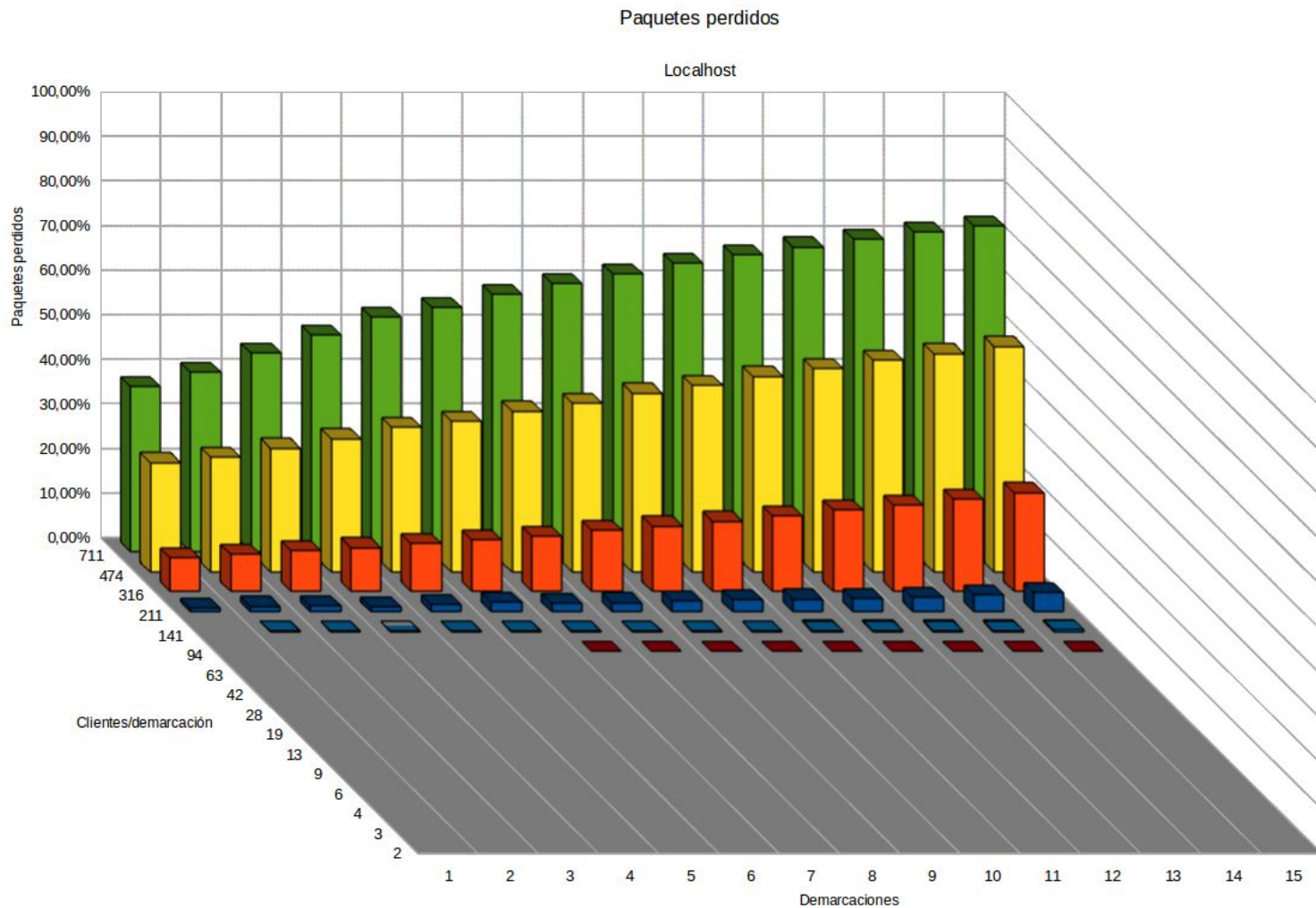
- ❖ Probado en localhost y sobre LAN cableada.
- ❖ Entre 1 y 15 demarcaciones.
- ❖ Entre 2 y 711 clientes **por demarcación (no totales)**.
- ❖ **Diez iteraciones** por configuración.
- ❖ Tiempo **máximo** por iteración de **30 segundos**:
  - Espera inicial aleatoria de entre 0 a 10 segundos.
  - Timeout establecido a 20 segundos.



# Pruebas - tiempo de ejecución localhost

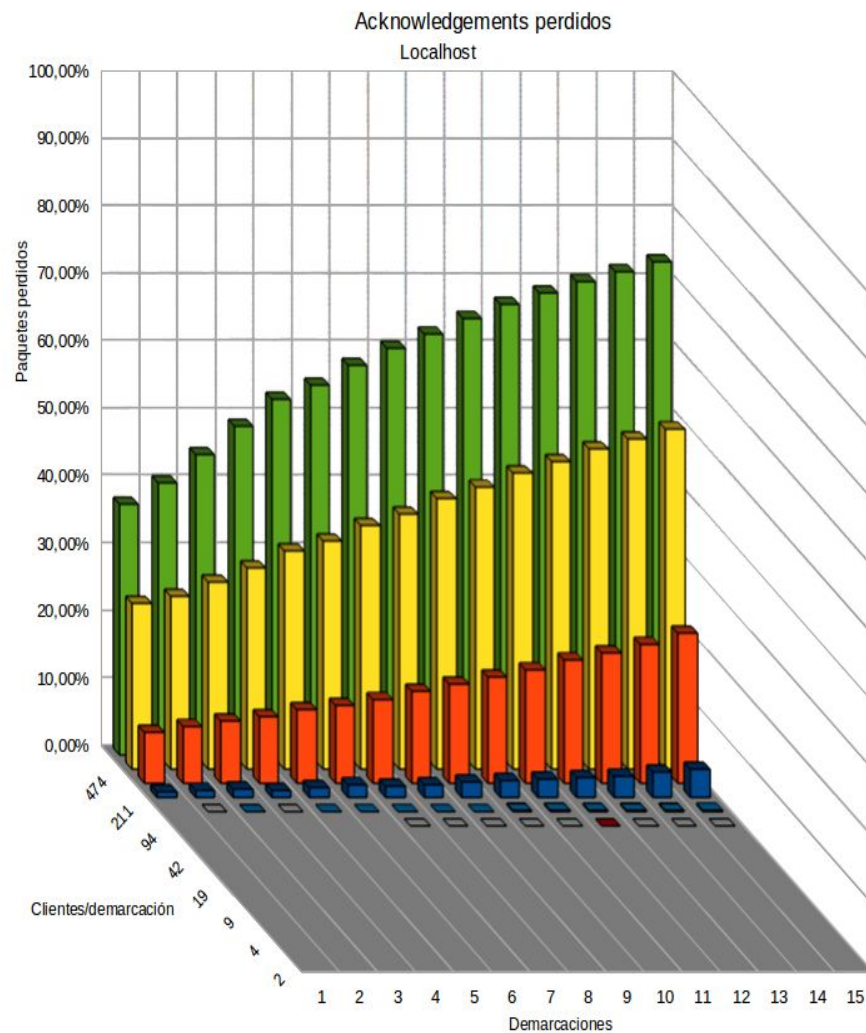
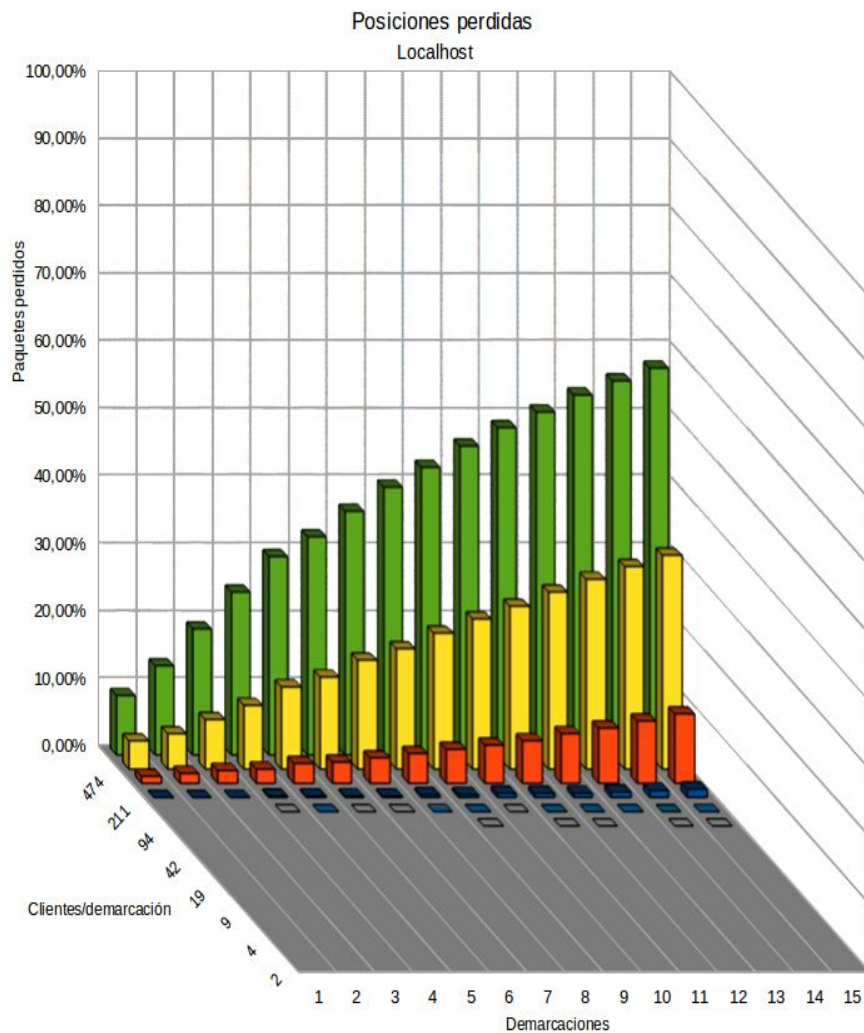


# Pruebas - pérdida total en localhost





# Pruebas - pérdida en localhost



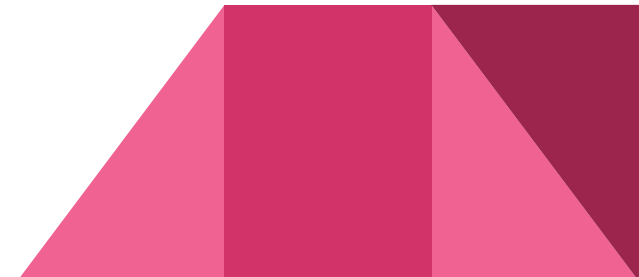


# Pruebas - LAN

- Nuestro servidor es C.
- Multiplataforma.
- Con un consumo de procesador y memoria RAM despreciable.

## Así que:

- ¿Por qué limitarnos a probarlo en un sistema Windows?
- Es más, ¿por qué limitarnos a un sobremesa?
- Y aún más, ¿por qué limitarnos a probarlo en un x86?

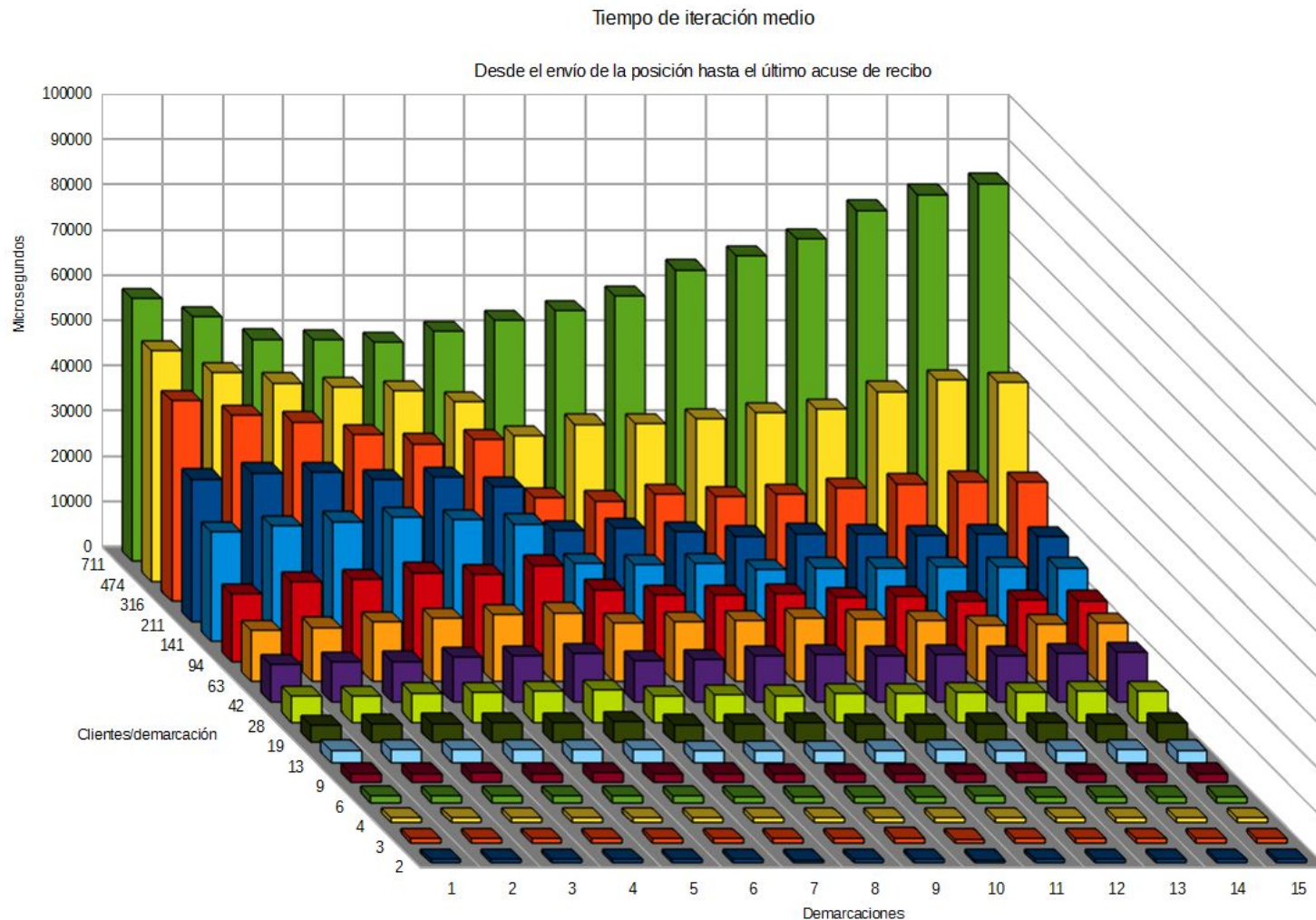


# Pruebas - LAN

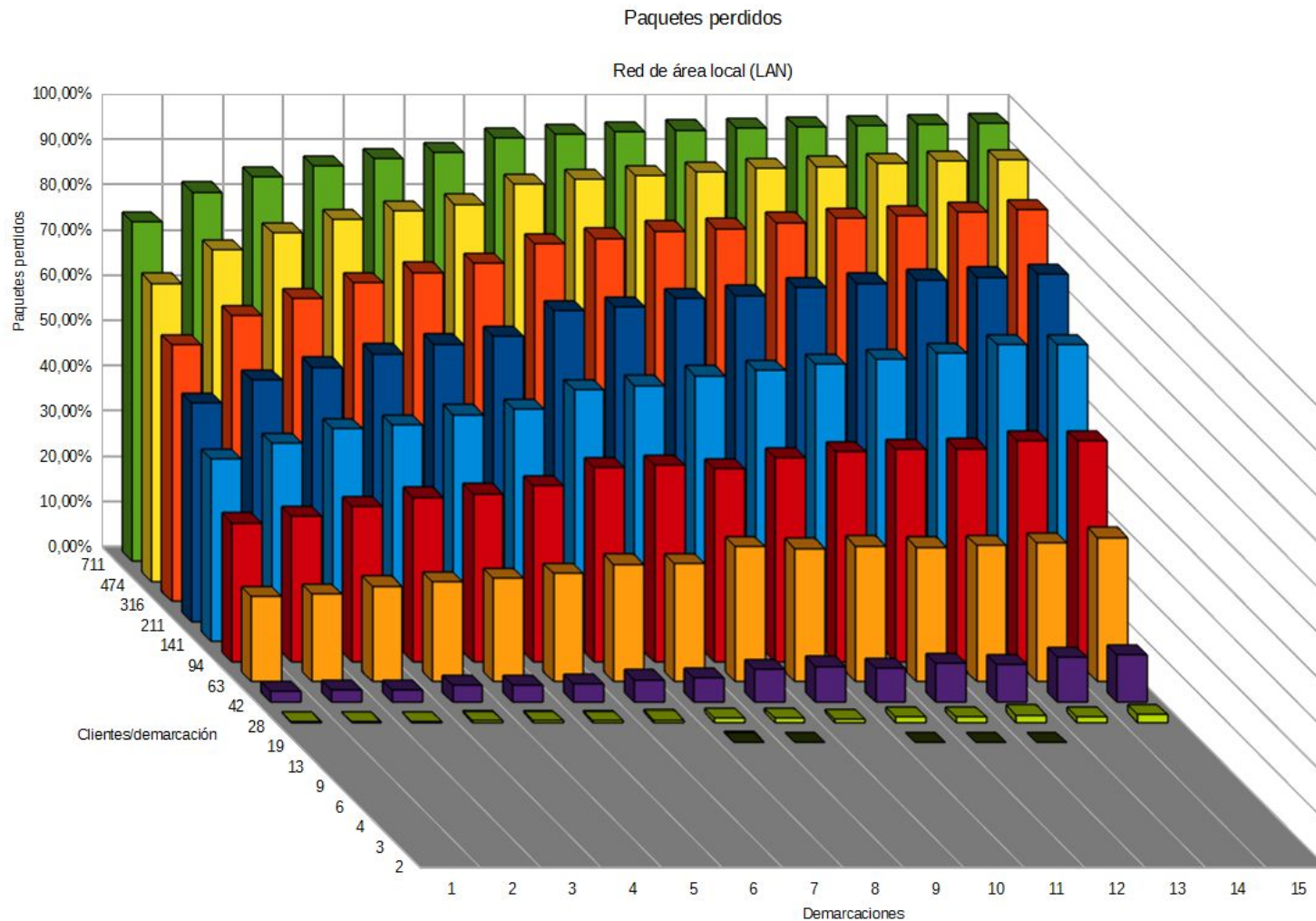


¿Por qué no probarlo en un ordenador ARM chino de 15€?

# Pruebas - tiempo de ejecución LAN



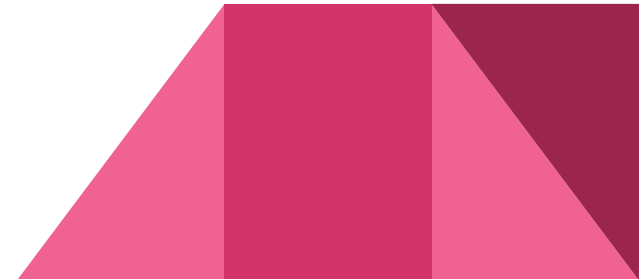
# Pruebas - pérdida total en LAN



# Pruebas - conclusiones

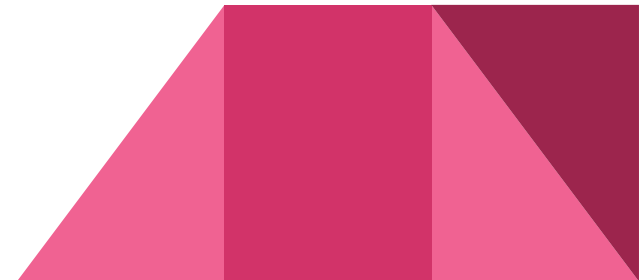
Lo que estaréis todos pensando ahora:

**“Vaya castaña de protocolo, se pierden un montón de paquetes”**



# Pruebas - conclusiones

**¿Cómo es posible que pueden perderse tantos paquetes?**

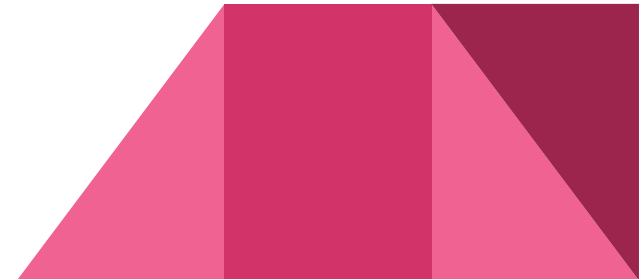




# Pruebas - conclusiones

Hemos hecho cálculos:

$$\text{paquetes} = 2 \times \text{iteraciones} \times \text{demarcaciones} \times (\text{clientes}^2 - \text{clientes})$$



# Pruebas - conclusiones

- ❖ 5 demarcaciones
- ❖ 711 clientes cada una
- ❖ 10 iteraciones

$$\text{paquetes} = 2 \times 10 \times 5 \times (711^2 - 711)$$

$$\text{paquetes} = 50.481.000$$

Para 30 segundos por iteración como en las pruebas:

168.270 paquetes/segundo

¿Ya no parece tan raro, no?





# Desarrollo: Tiempo Estimado Vs Tiempo Real

Aquí podemos observar las diferencias del tiempo que hay entre lo planificado en un principio y el tiempo real que hemos empleado a la hora de realizar el proyecto, que como podéis ver es bastante mayor.

Aquí tenemos la estimación que realizamos en la primera sesión del proyecto, que como veremos a continuación nos quedamos muy lejos.

			Horas individuales	Horas grupales
SESIÓN 1	23 Septiembre	5 Octubre	3 Horas	5 Horas
SESIÓN 2	5 Octubre	26 Octubre	4 Horas	5 Horas
SESIÓN 3	26 Octubre	23 Noviembre	5 Horas	6 Horas
SESIÓN 4	23 Noviembre	21 Diciembre	5 Horas	6 Horas

# Desarrollo: Tiempo Estimado Vs Tiempo Real

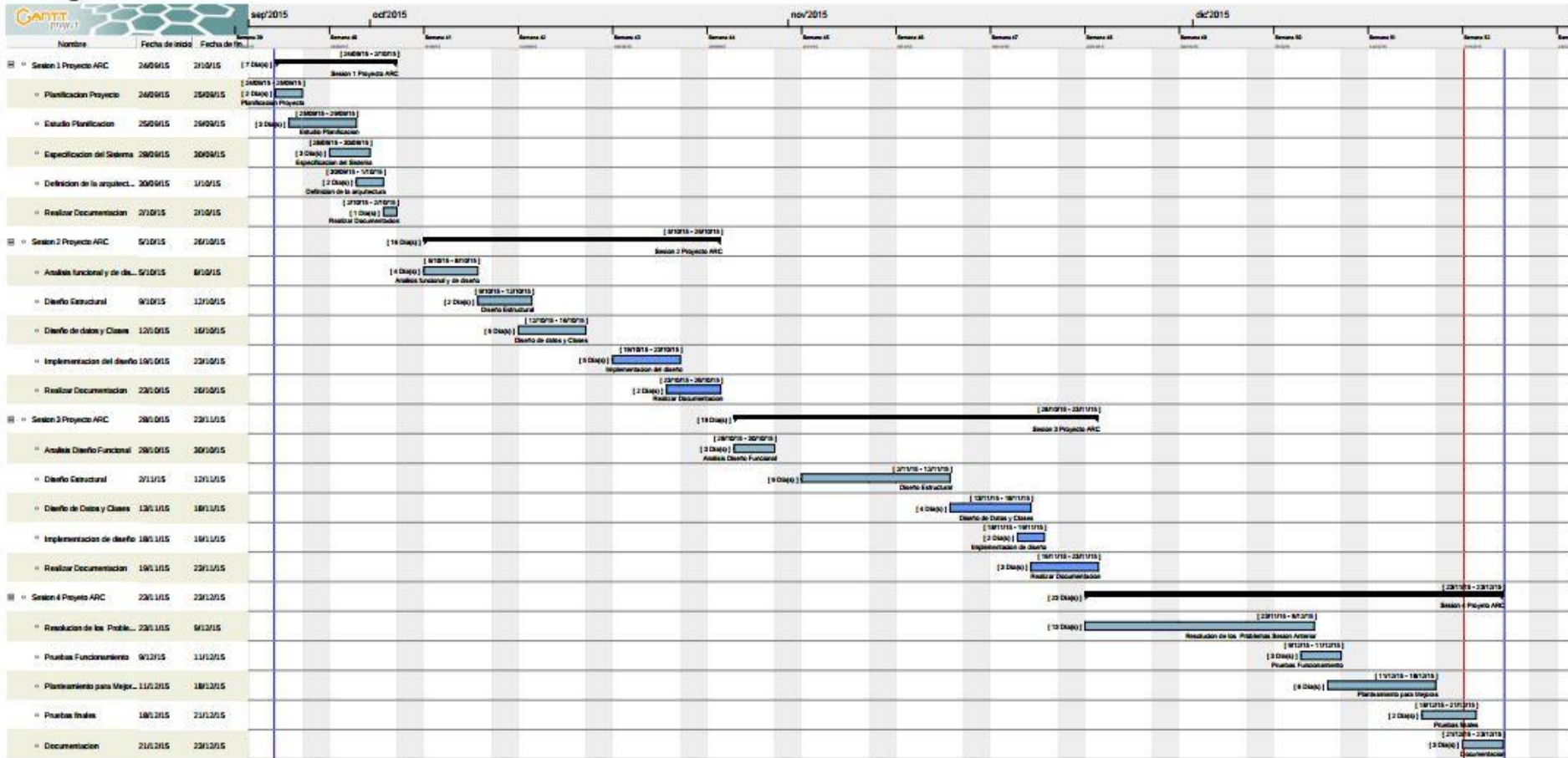
## Tarea

Nombre	Fecha de inicio	Fecha de fin
Sesion 1 Proyecto ARC	24/09/15	2/10/15
Planificacion Proyecto	24/09/15	25/09/15
Estudio Planificacion	25/09/15	29/09/15
Especificacion del Sistema	28/09/15	30/09/15
Definicion de la arquitectura	30/09/15	1/10/15
Realizar Documentacion	2/10/15	2/10/15
Sesion 2 Proyecto ARC	5/10/15	26/10/15
Analisis funcional y de diseño	5/10/15	8/10/15
Diseño Estructural	9/10/15	12/10/15
Diseño de datos y Clases	12/10/15	16/10/15
Implementacion del diseño	19/10/15	23/10/15
Realizar Documentacion	23/10/15	26/10/15
Sesion 3 Proyecto ARC	28/10/15	23/11/15
Analisis Diseño Funcional	28/10/15	30/10/15
Diseño Estructural	2/11/15	12/11/15
Diseño de Datos y Clases	13/11/15	18/11/15
Implementacion de diseño	18/11/15	19/11/15
Realizar Documentacion	19/11/15	23/11/15
Sesion 4 Proyecto ARC	23/11/15	23/12/15
Resolucion de los Problemas Sesion Anterior	23/11/15	9/12/15
Pruebas Funcionamiento	9/12/15	11/12/15
Planteamiento para Mejoras	11/12/15	18/12/15
Pruebas finales	18/12/15	21/12/15
Documentacion	21/12/15	23/12/15

# Desarrollo: Tiempo Estimado Vs Tiempo Real

3

## Diagrama de Gantt



# Código fuente

<https://github.com/UV-AXC-G1>

- ❖ Versión en C (actual, desde la sesión 3):
  - TaxiC
    - Rama “master” con sincronización por secuestro
    - Rama “adaptime” con sincronización en colmena
- ❖ Versión en Java (original, descartada pero funcional ☹):
  - TaxiComms
  - TaxiServer
  - TaxiClient