SISTEMA DISTRIBUÍDO DE SERVICIOS DE TAQUILLA VIRTUAL

•••

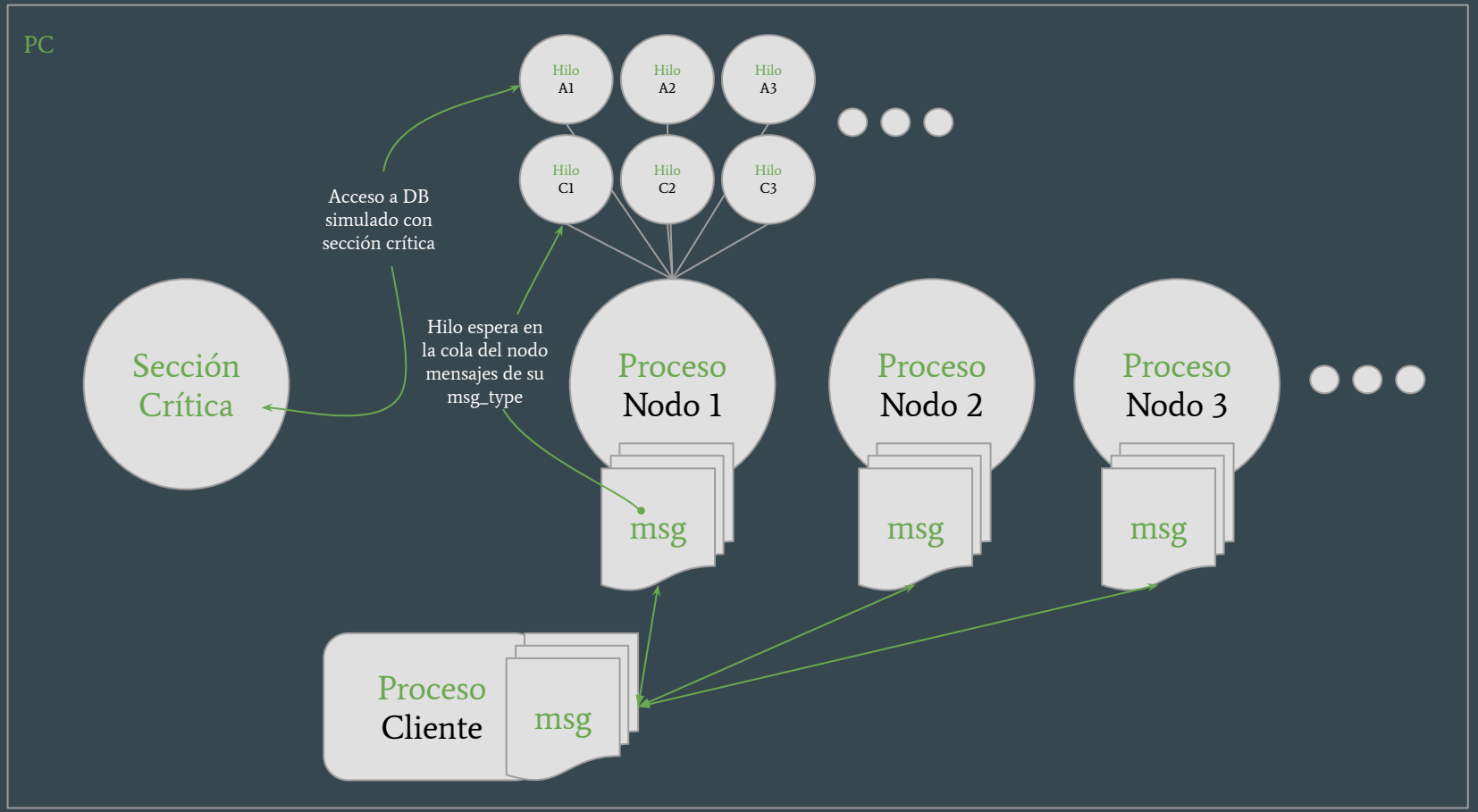
SOLUCIÓN FINAL

Aarón Riveiro Vilar
Adrian Cabaleiro Althoff
Iván Pillado Rodríguez
Manuel Martínez Vaamonde

Objetivos logrados

- 5 tipos de proceso por nodo (pagos, anulaciones, reservas, administración, consultas)
- Varios procesos del mismo tipo simultáneamente en un mismo nodo
- Sistema de prioridades
- Concurrencia de lectores
- Exclusión mutua de lectores/escritores
- Más de 10 nodos
- Más de 100 procesos por nodo

Entorno de pruebas



Estructura

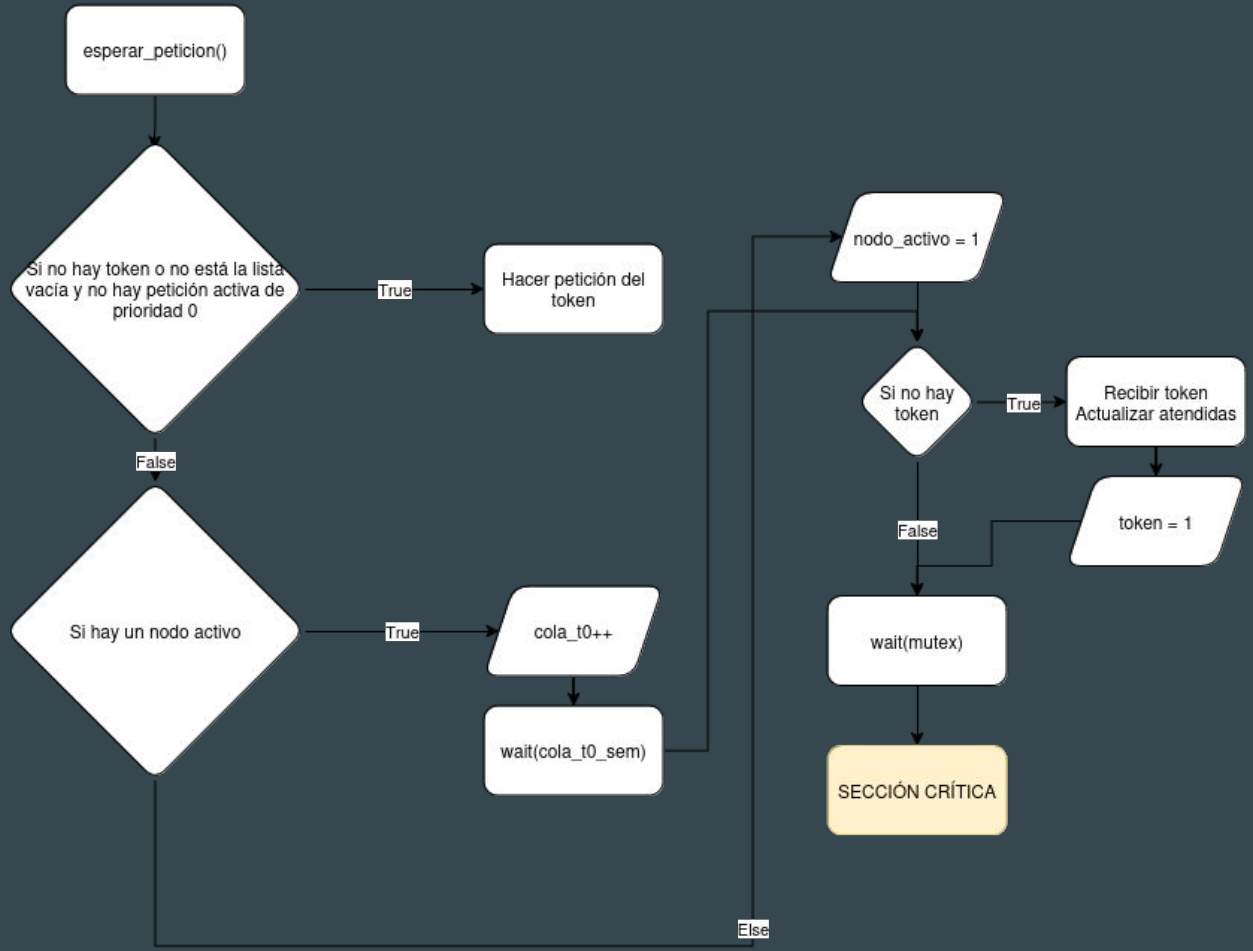
- **Nodos:**
  - **nodo.c** -> programa con toda la lógica del algoritmo. Simula uno de los múltiples nodos del sistema distribuído
  - **utils.c** -> funciones auxiliares para resolver la lógica del nodo. Gestión de una lista enlazada (añadir elemento, eliminar elemento, comprobar cantidad de elementos...), funciones relacionadas con comprobaciones sobre los vectores y prioridades, etc.
  - **ejecutar_nodo.sh** -> script para automatizar la ejecución de varios nodos
  - **kill.c** -> programa para enviar una señal de finalización a todos los nodos

- **Cliente:**
  - **cliente.c** -> programa que muestra un menú que permite seleccionar qué tipo de solicitud enviar y a qué nodo
  - **cliente_rand.c** -> programa que envía de forma automática y aleatoria una serie de solicitudes al sistema, según lo indicado en sus parámetros de ejecución

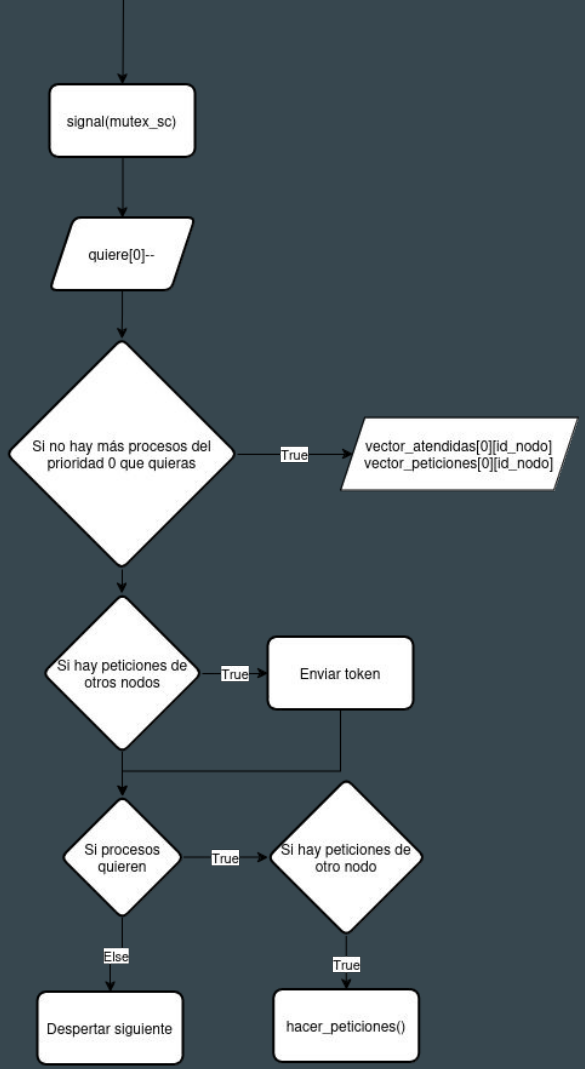
Detalles de Implementación

- Sistema de prioridades:
  - Prioridad 0: pagos y anulaciones
  - Prioridad 1: reservas y administración
  - Prioridad 2: consultas
- Se simula la Base de Datos con esperas de 1 segundo: sleep(1)
- Para hacer variar el número de nodos, es necesario modificar un #define en el código fuente
- Después de cada ejecución, asegurarse de que la cola de mensajes creada que limpia o es eliminada

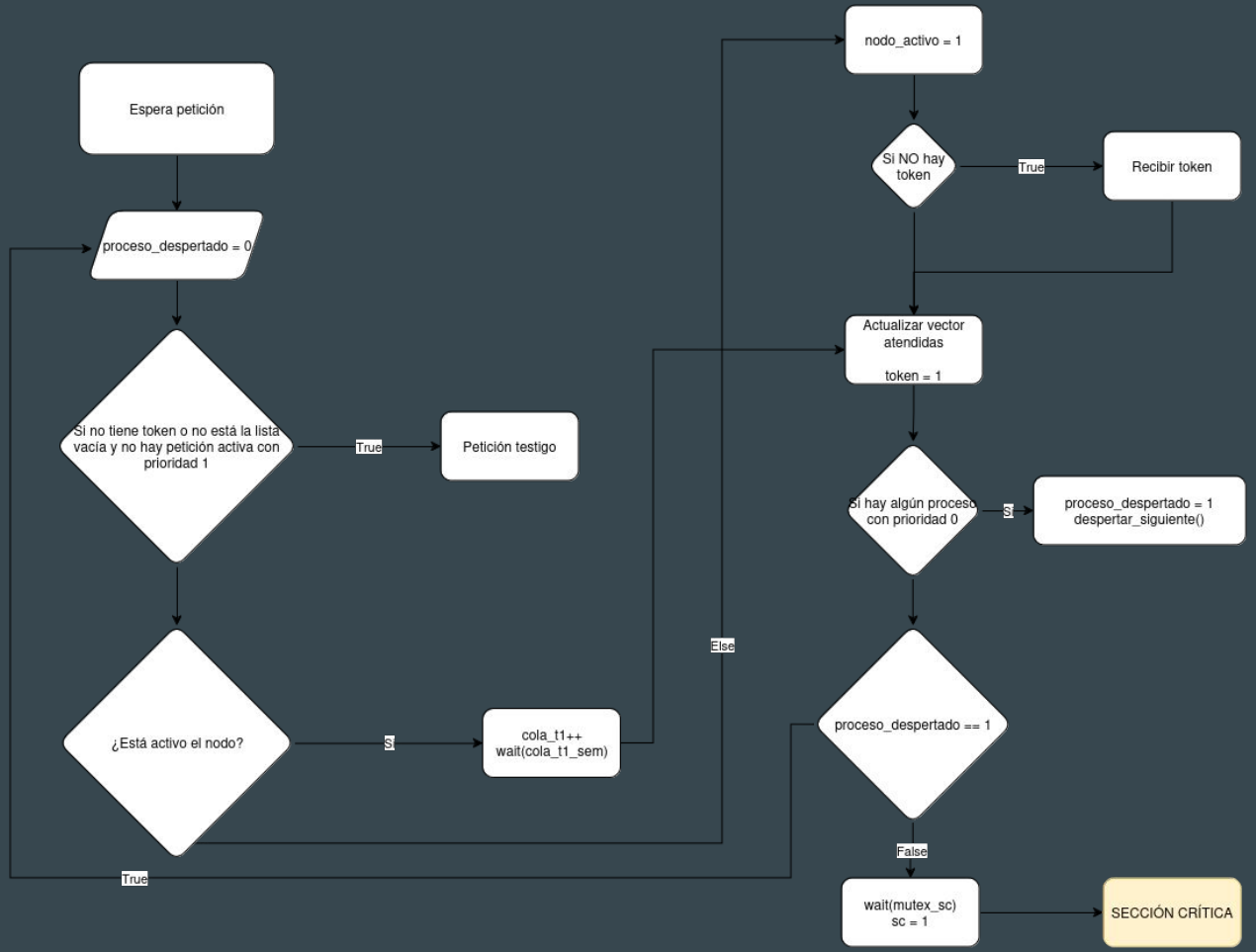
Procesos TO (1 / 2)



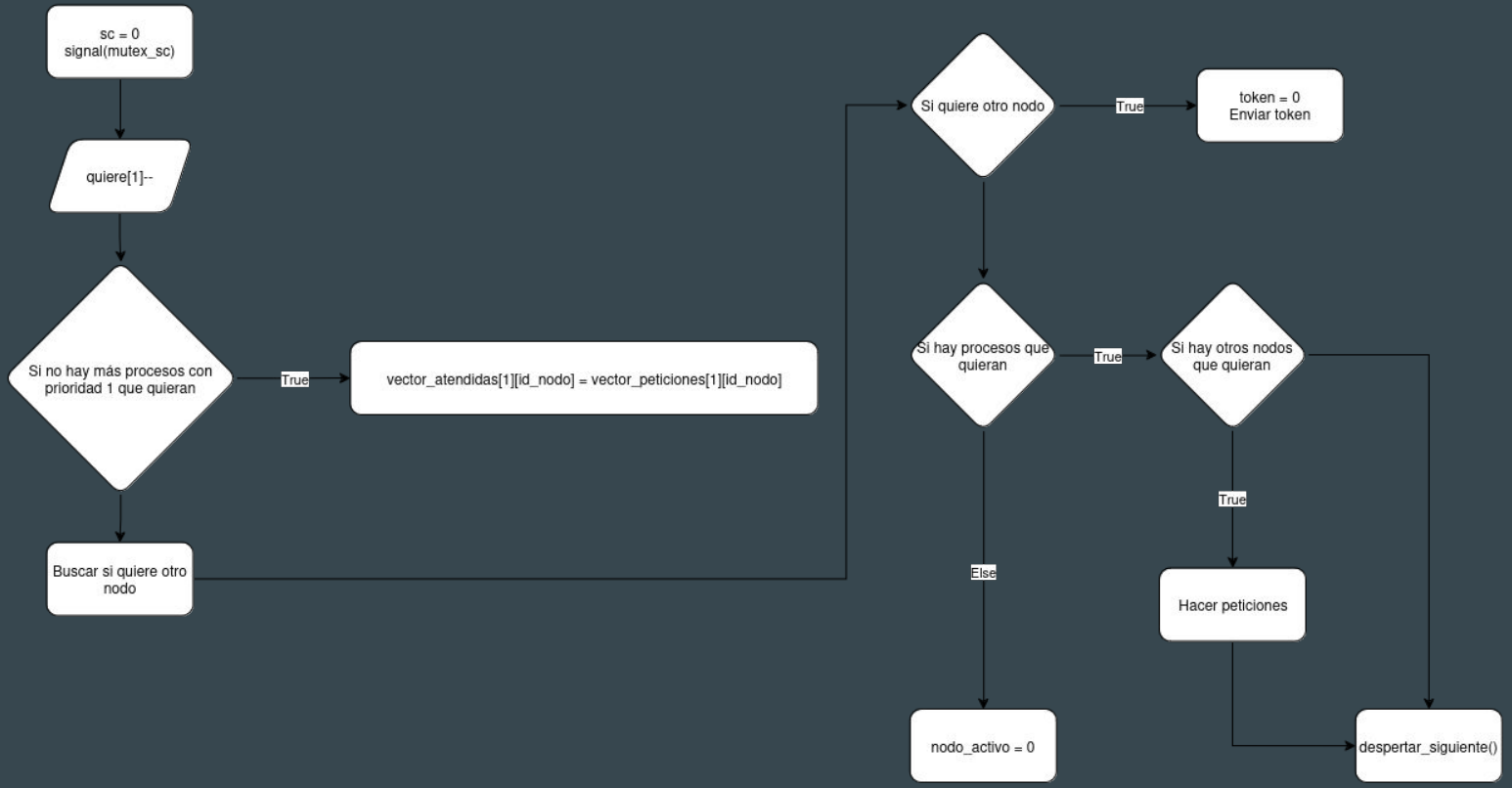
Procesos TO (2 / 2)



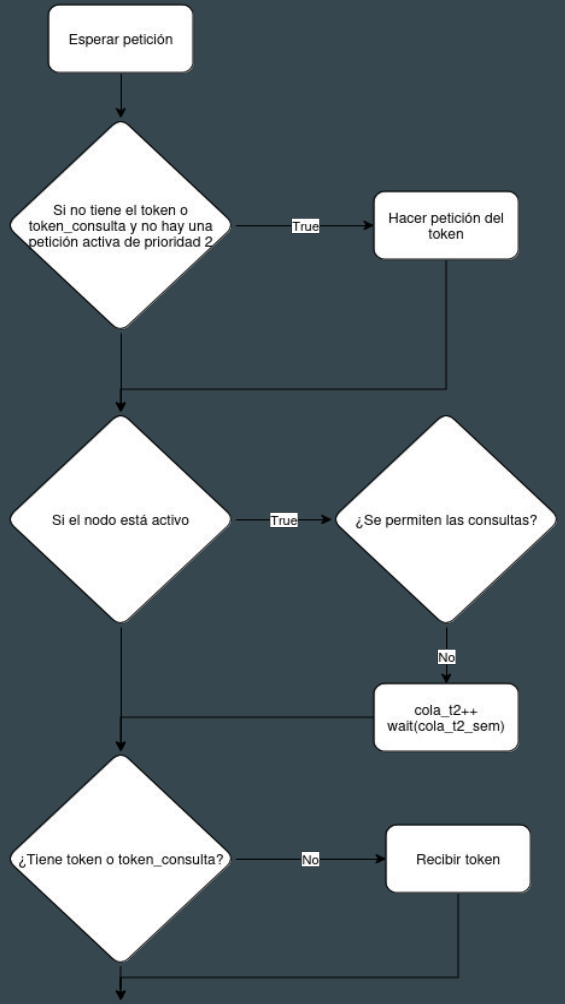
Procesos T1 (1 / 2)



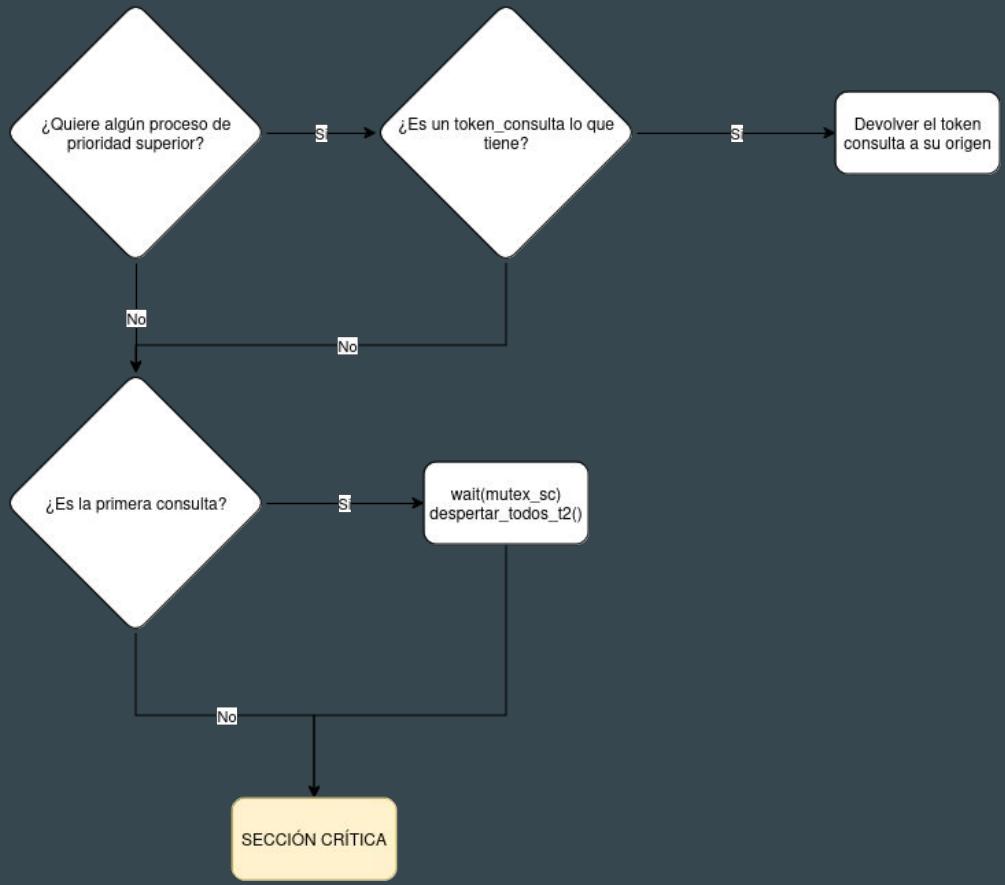
Procesos T1 (2 / 2)



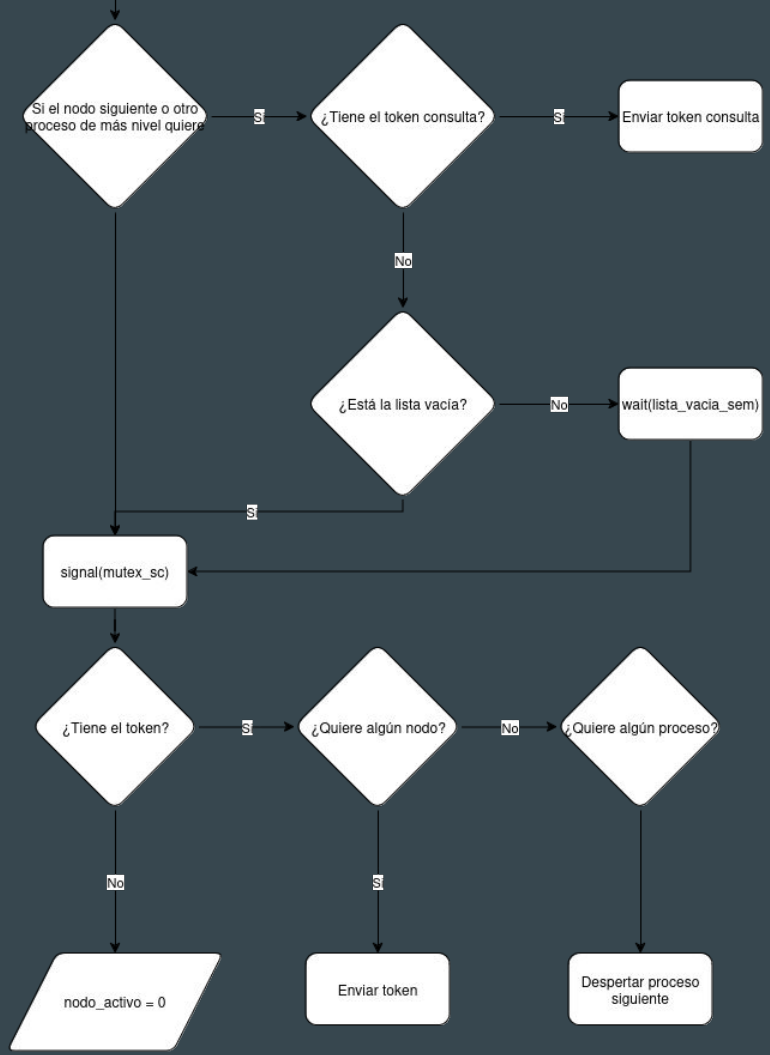
Procesos T2 (1 / 3)



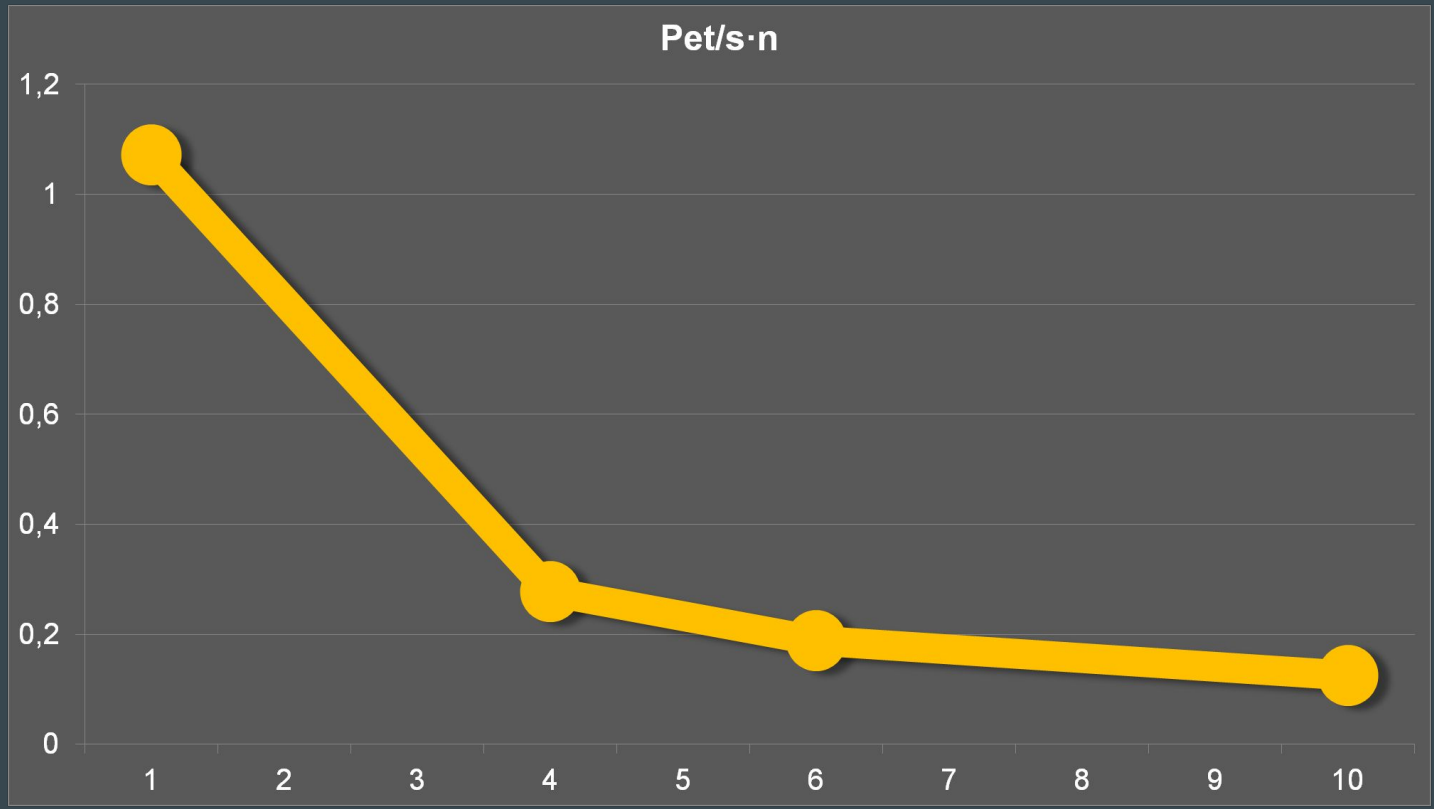
Procesos T2 (2 / 3)



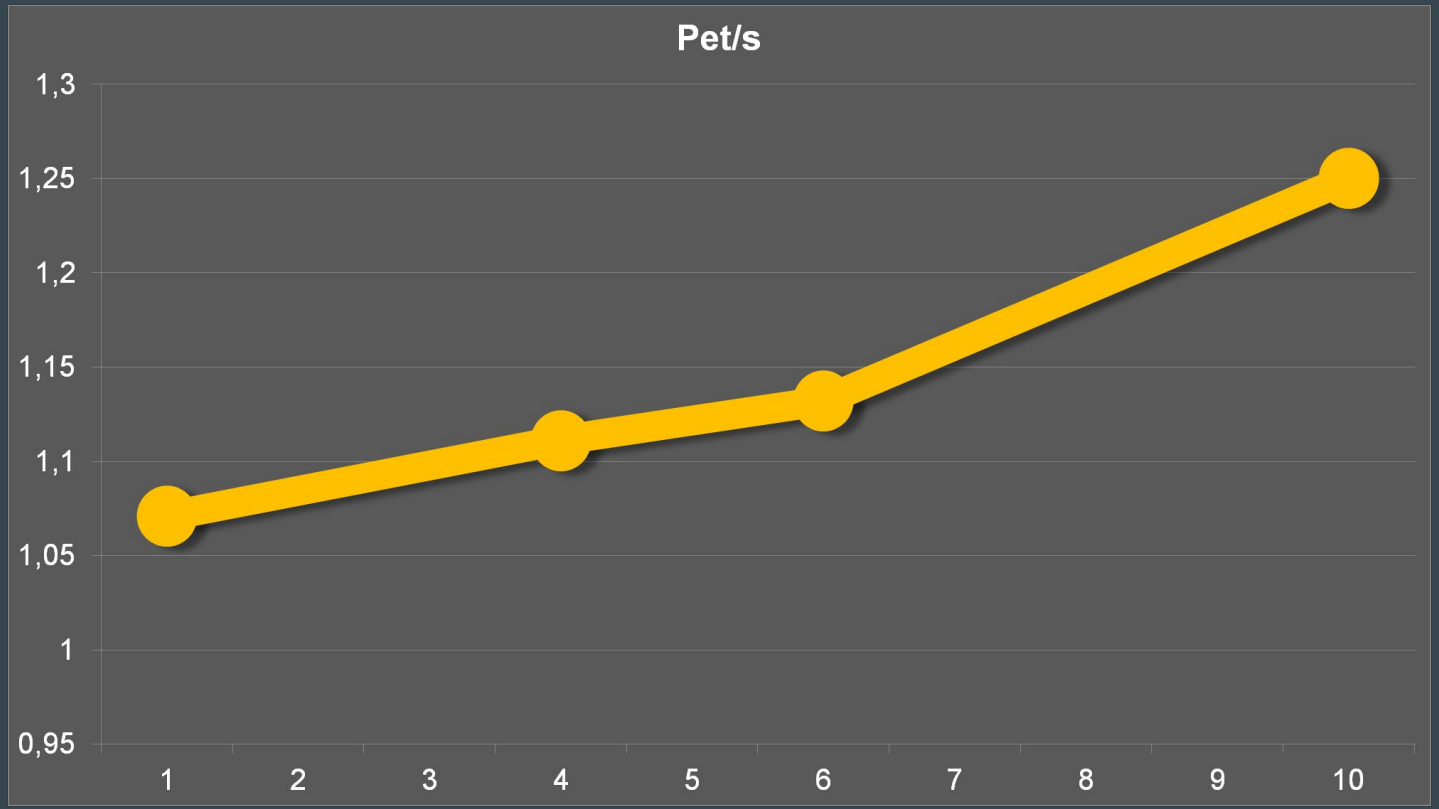
Procesos T2 (3 / 3)



Métricas: Peticiones por segundo y nodo



Métricas: Peticiones por segundo



POSIBLE CONTINUACIÓN DEL PROYECTO

- Implementar adelantamientos: evitar inanición de consultas
- Permitir que un nodo con un testigo copia pueda emitir más testigos copia: mejora de rendimiento
- Implementar la base de datos real
- Implementar un balanceador de carga para varios clientes
- Implementar nodos dinámicos: permitir altas y bajas de nodos