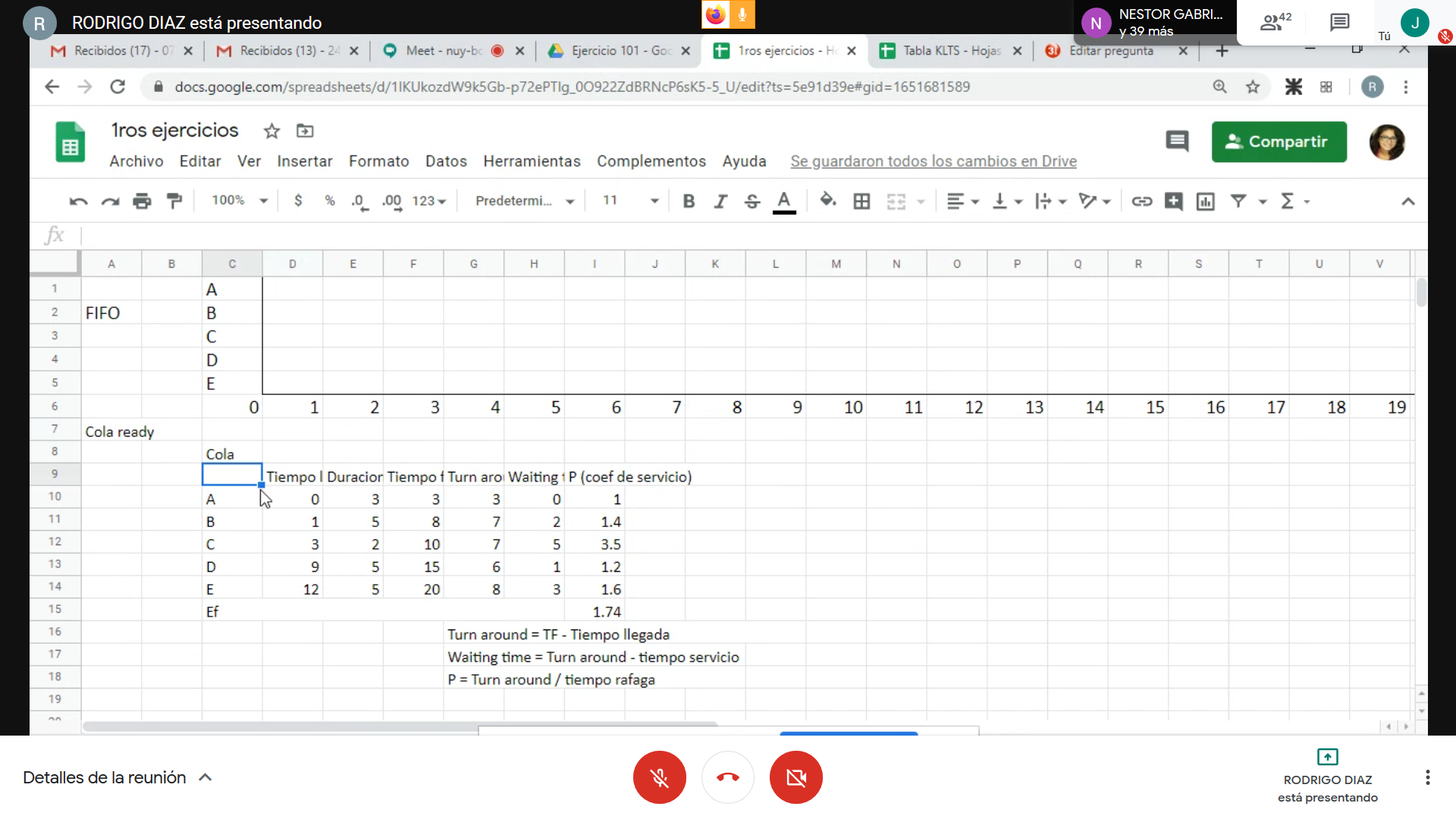
El planificado mediano plazo, lo que hace es traer el proceso desde memoria secundaria a memoria principal, no se encarga de finalizar y crear.

El response time es un criterio orientado al usuario.

El starvation o inanición es un problema que puede ocurrir en algoritmos que utilicen prioridades.

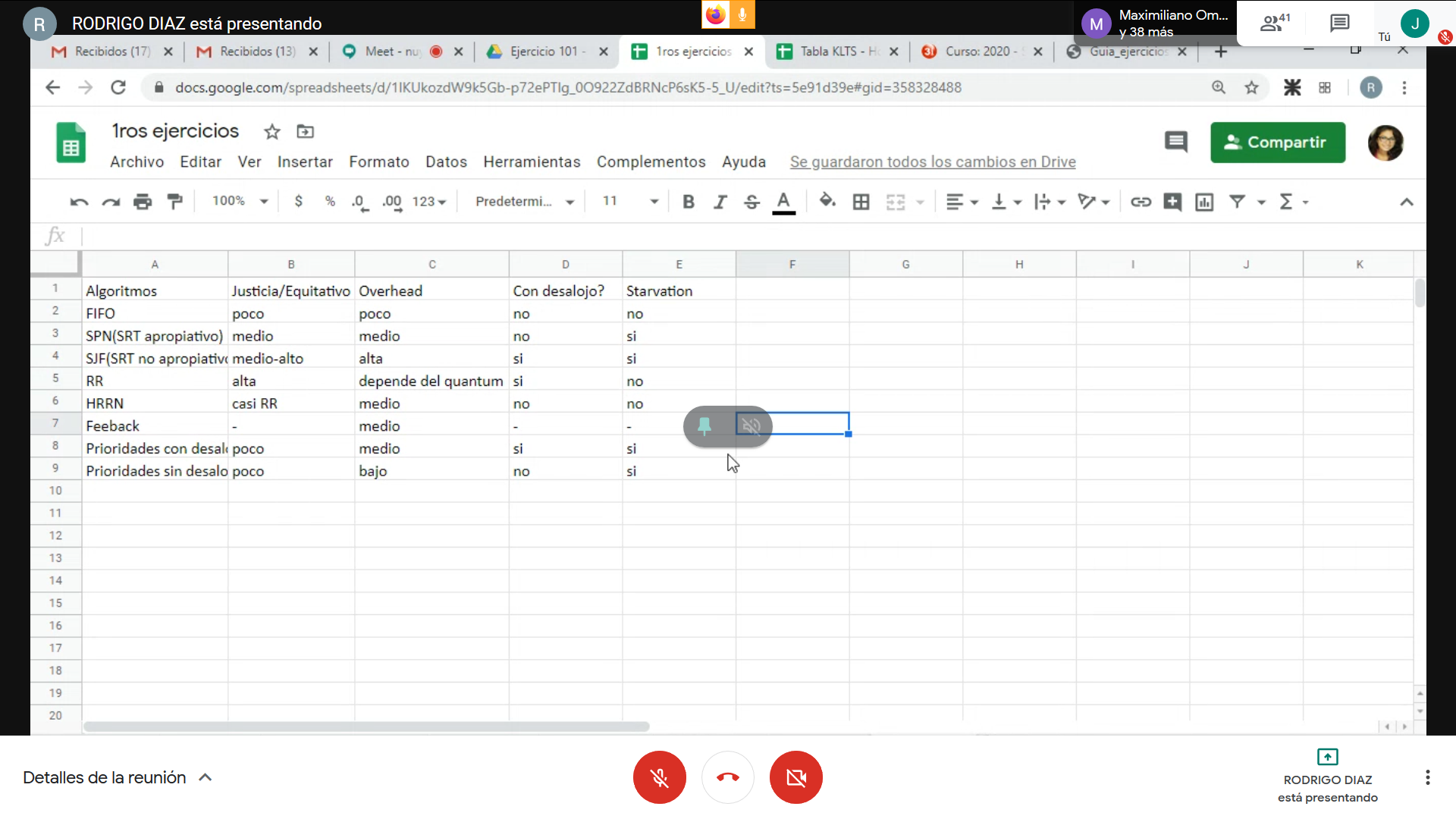
En un algoritmo multinivel retroalimentado la prioridad de los procesos es estática.

Ejemplo:

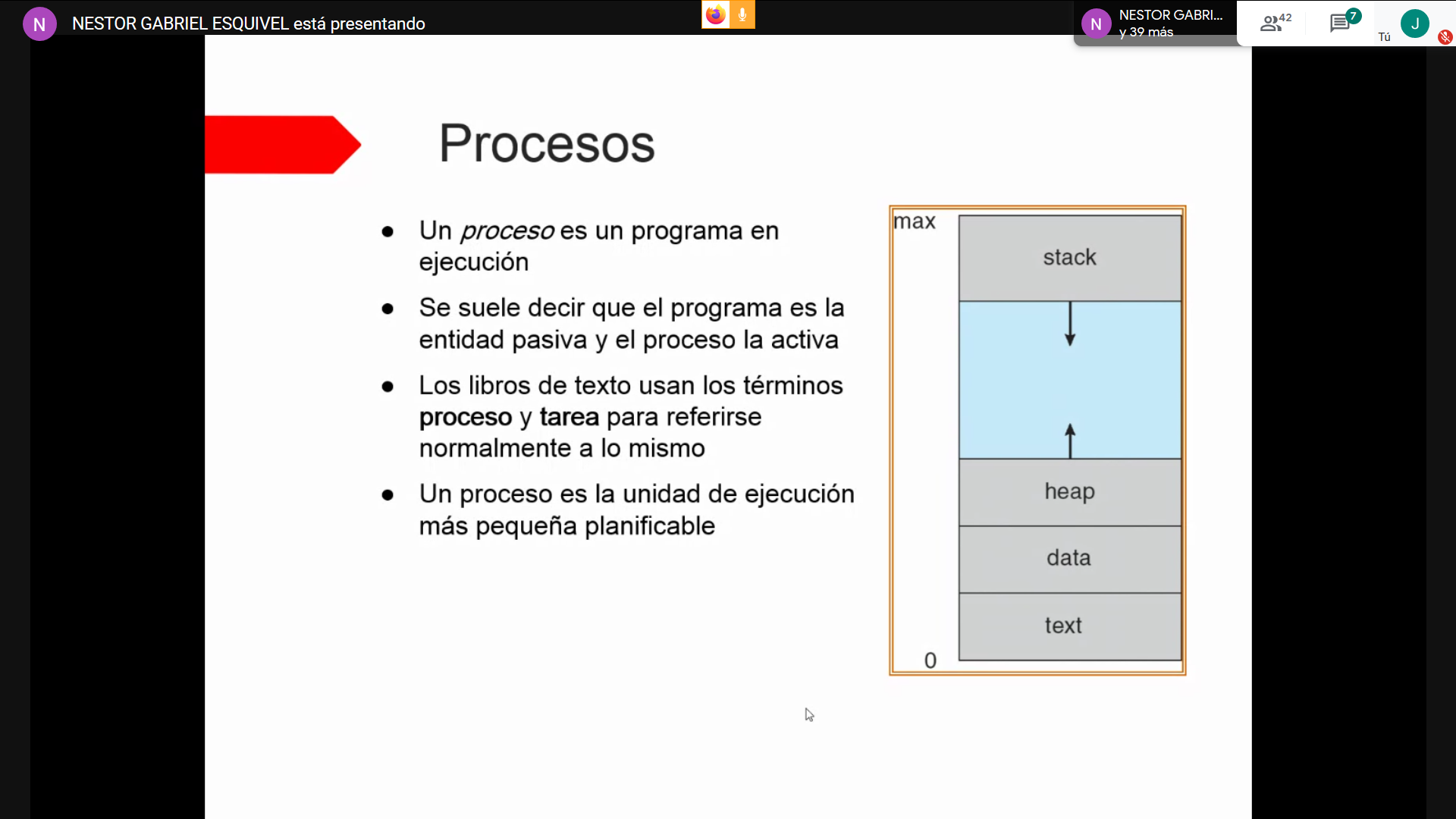


En RR mientras mas chico es el Q , mas cambios de contexto se va a tener.

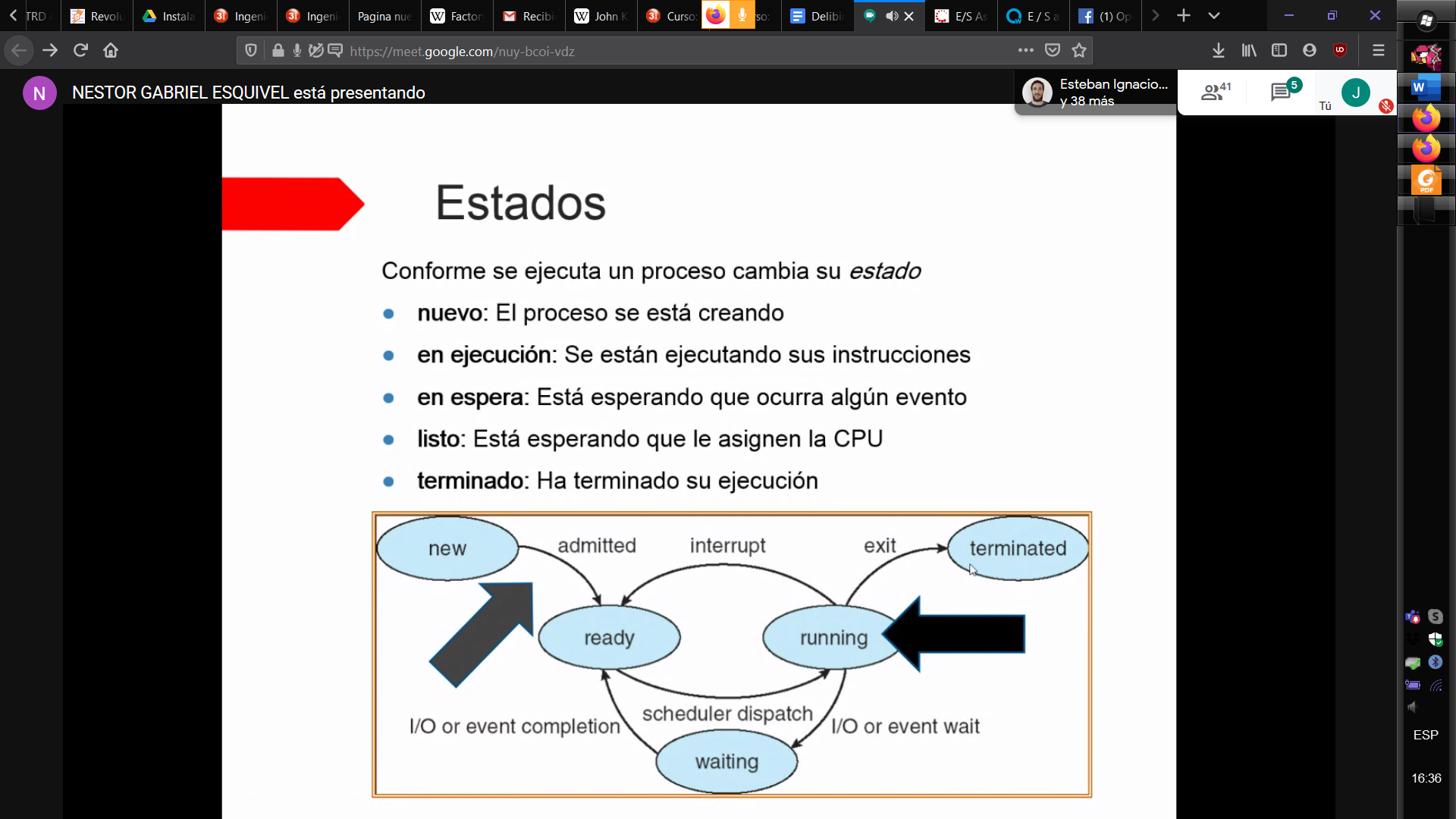
Cuadro de comparaciones de algoritmos:

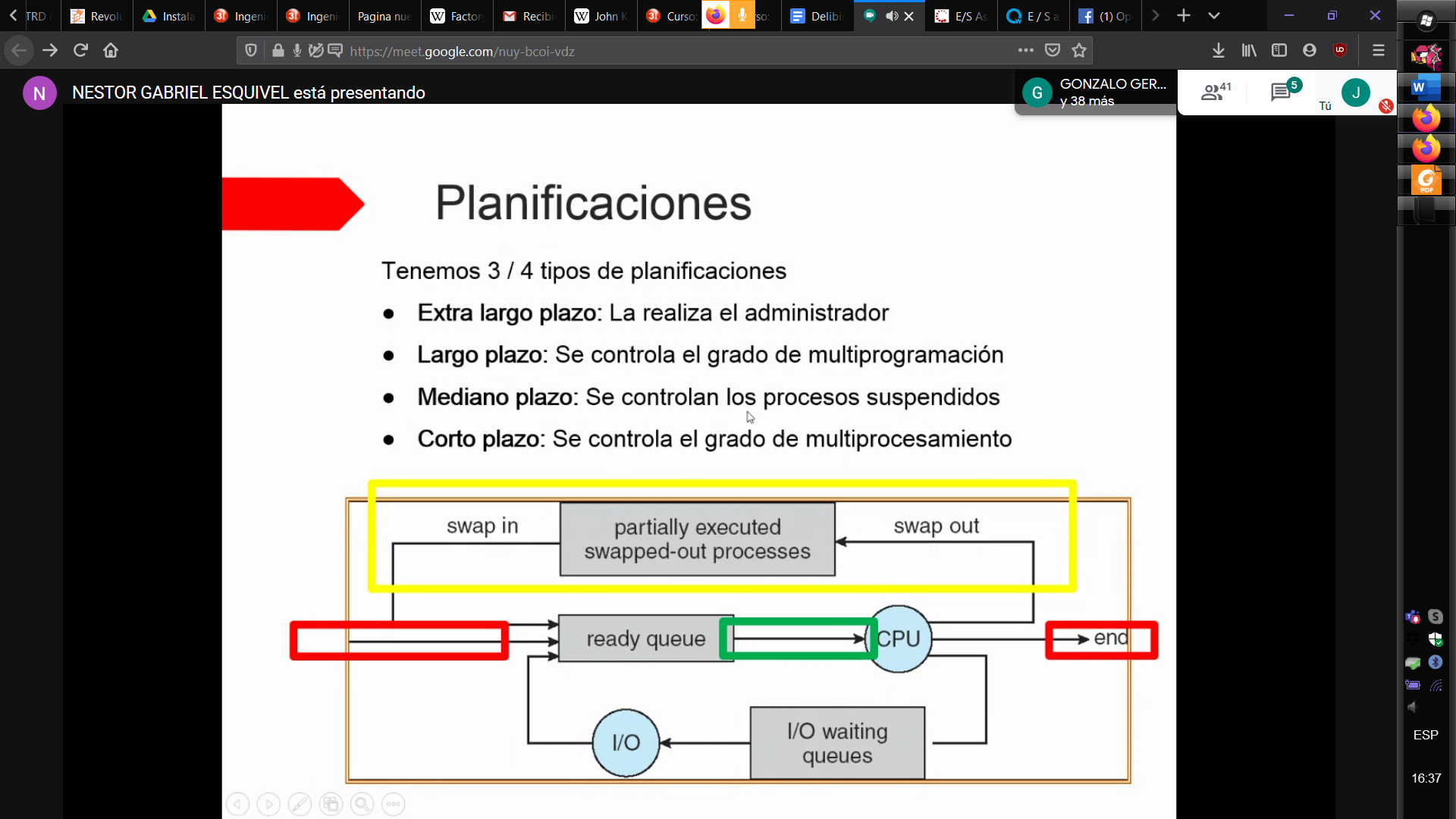


Repaso procesos:



Lo que se ve en la imagen es que el proceso tiene un espacio de direcciones.





Lo hilos tienen:

1. un estado de ejecución por hilo
2. un contexto de hilo en ejecución Una pila de ejecución
3. Locales por hilo
4. Acceso a la memoria y recursos de un proceso
5. Mientras el procesos se esta ejecutando , los registros del procesador se controlan por un proceso y cuando el proceso no se está ejecutando se almacena el contenido de los registros.

En un entorno multihilo sigue habiendo un único bloque de procesos y un espacio de direcciones de usuario asociados al proceso, pero ahora hay varias pilas separadas para cada hilo, así como un bloque de control para cada hilo que contiene los valores de los registros, la prioridad y otra información relativa al estado del hilo. De esta forma todos los hilos de un proceso comparten el estado y los recursos de ese proceso reciben en el mismo espacio de direcciones y tiene acceso a los mismos datos.

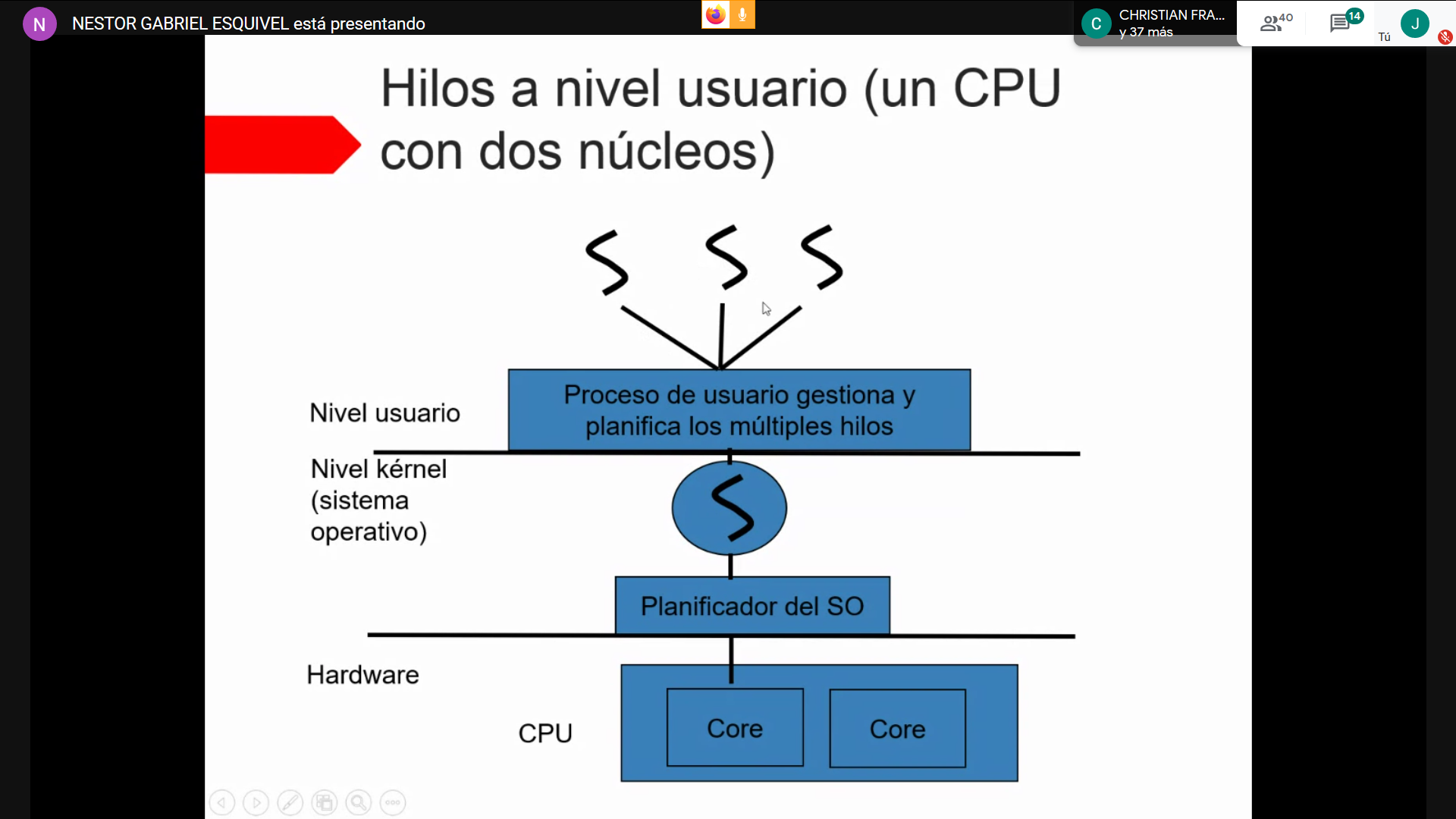
Rendimiento:

1. Lleva menos tiempo crear un nuevo hilo en un proceso existente que crear un proceso nuevo
2. Lleva menos tiempo finalizar un hilo que un proceso
3. Lleva menos tiempo cambiar entre dos hilos de un proceso
4. Mejora la eficiencia de la comunicación entre diferentes programas que se están ejecutando
5. Los hilos se pueden comunicar entre ellos sin necesidad de involucrar el núcleo
6. Suele ser más rápido usar hilos y memoria compartida que usar procesos y pasos de mensajes para coordinación

(No existe un hilo en estado de suspensión, tiene un pseudo estado de suspensión por parte del proceso cuando va a suspendido)

En cualquier momento que la aplicación se está ejecutando se puede crear un hilo a ejecutar dentro de un mismo proceso, pero tenemos que diferenciar si son ULTs o KLTs.

ULT: en que la planificación de hilos la lleva a cabo la biblioteca



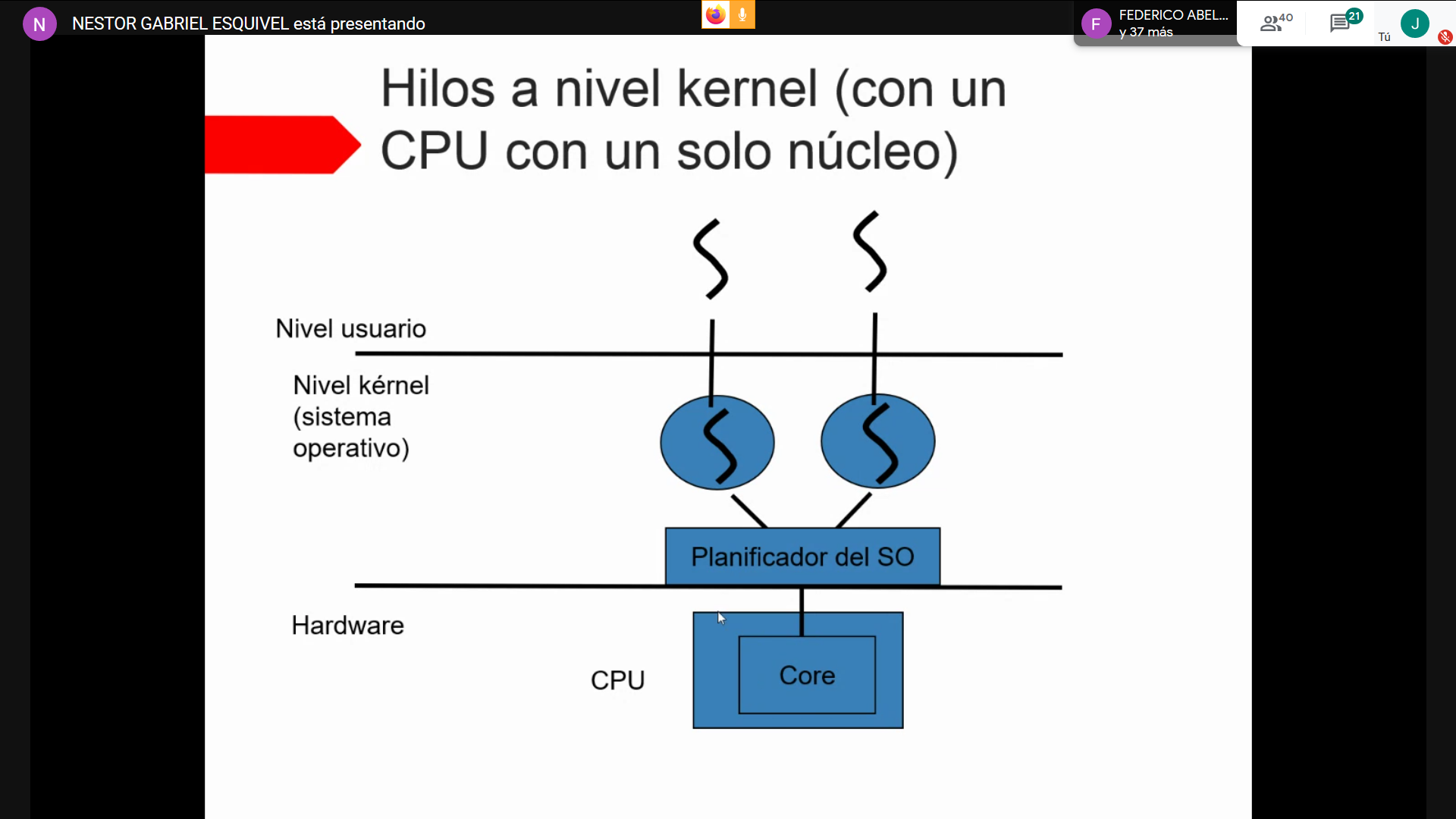
Ventajas:

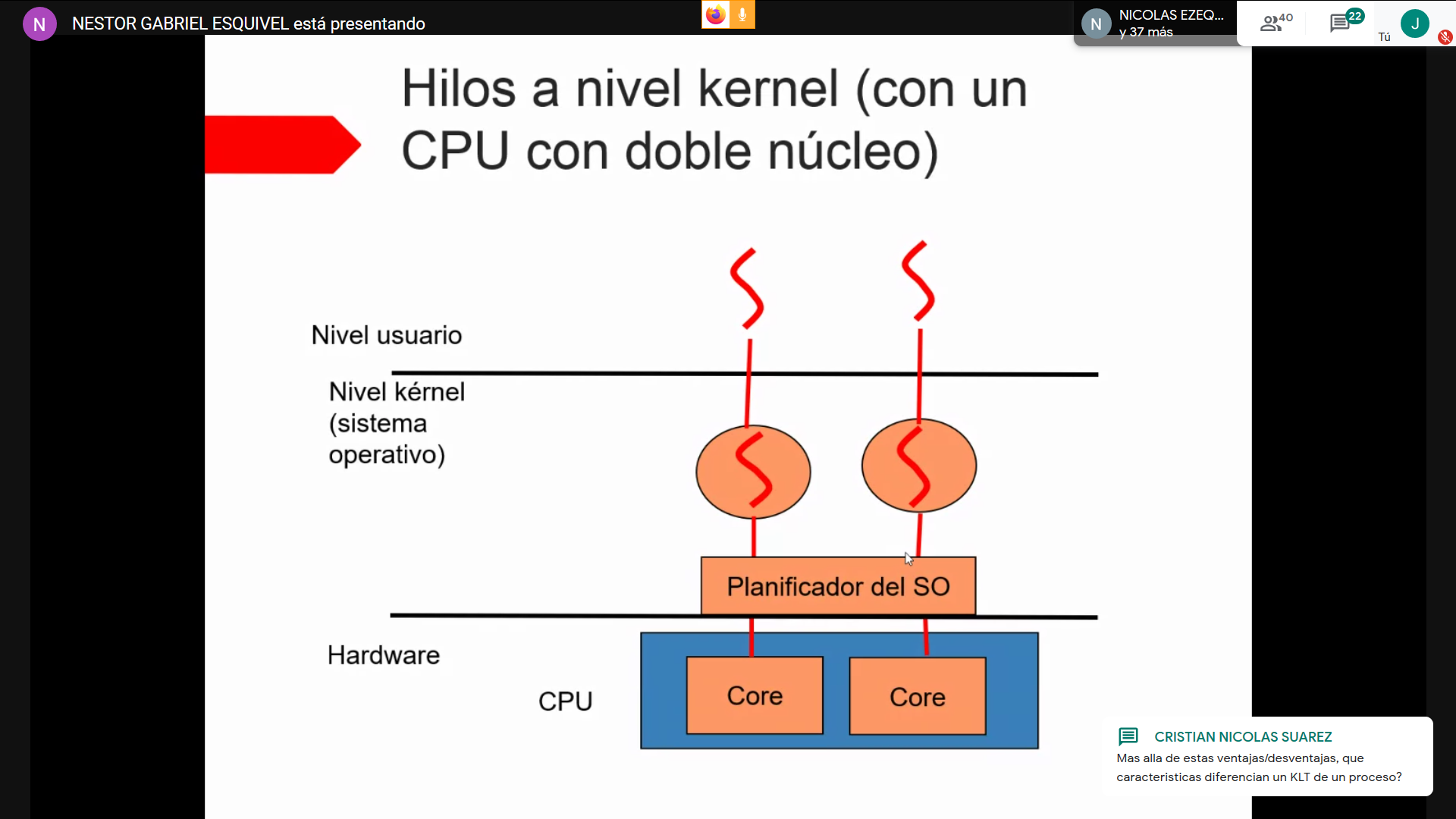
1. El cambio de hilo no requiere privilegios de modo kernel
2. Se puede especificar la planificación por parte de la aplicación
3. Se puede ejecutar en cualquier sistema operativo (los ULTs suelen decirse que son portables)

Desventajas:

1. Si los rpc son bloqueantes, al bloquear el hilo se bloquea el proceso
2. Una aplicación multihilos en una estrategia pura ULT no saca ventaja del multiproceso ya que el kernel asigna un proceso al procesador al mismo tiempo

KLT:





En un KLT puro el kernel gestiona todo el trabajo de gestión e hilos. No código de gestión de hilos en la aplicación, solamente hay una api para acceder a las utilidades de hilos del kernel. Todos los hilos de una aplicación se mantienen en un solo proceso y la planificación del núcleo se realiza a nivel hilo

La planificación de estos hilos la hace el SO, mas bien del dispatcher del SO

Ventajas:

1. El kernel puede planificar simultáneamente múltiples hilos de un solo proceso en múltiples procesadores
2. Si se bloquea un hilo de un proceso el kernel puede planificar otro hilo de un mismo proceso
3. Las rutinas del kernel pueden ser en si multihilo

Desventajas

1. La transferencia de control de un hilo a otro del mismo proceso requiere de un cambio del kernel

Cuadro de comparaciones:

