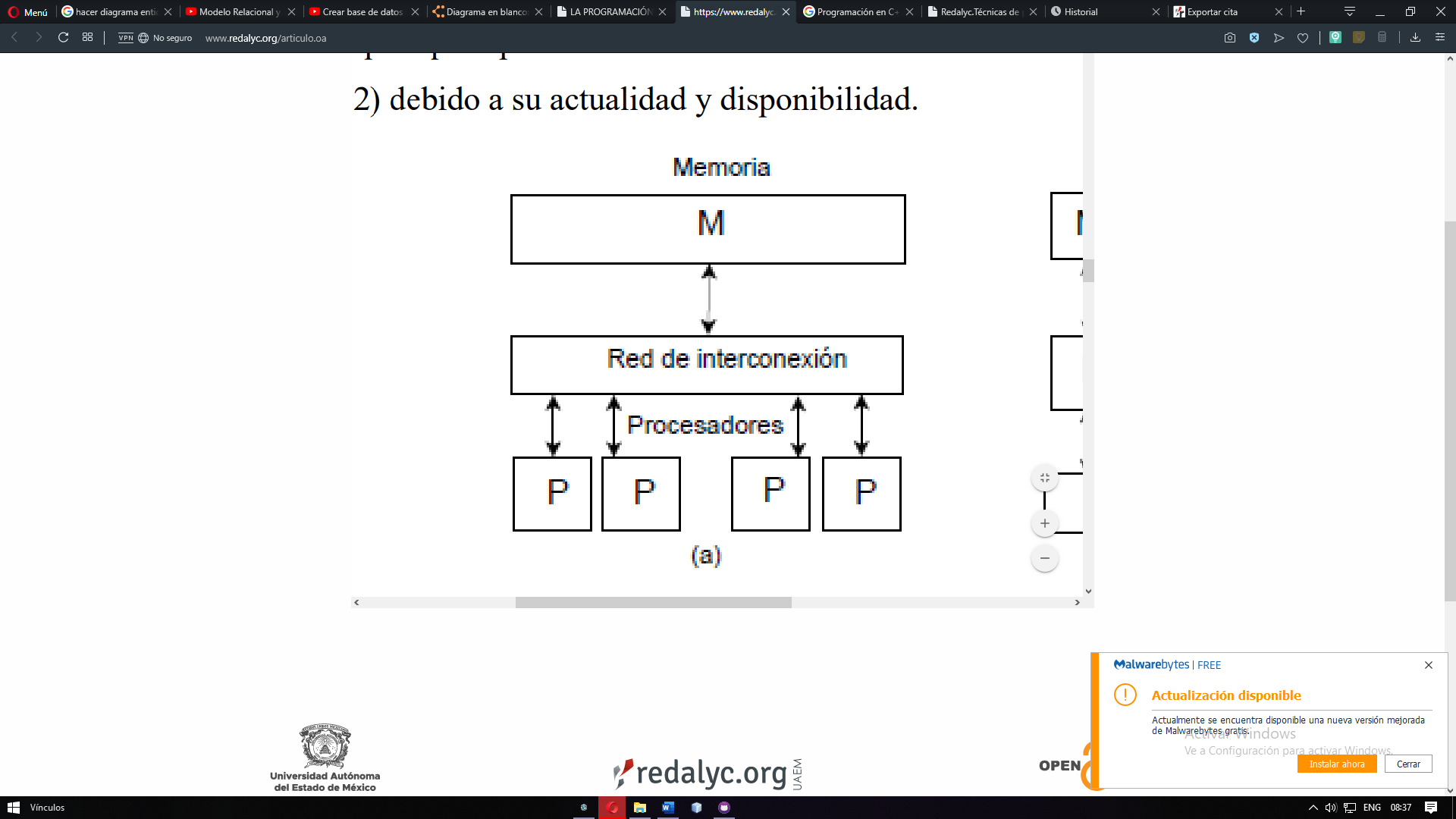
**Programación paralela**

El aumento de la demanda de problemas que exigen un alto costo computacional ha permitido el paso a computadores capaces de solucionar estos problemas de forma razonable, en el sentido que la computación secuencial fue incrementando continuamente para adaptarse a los tiempos actuales, llegando a un límite físico en su capacidad para resolver problemas de alto coste computacional.

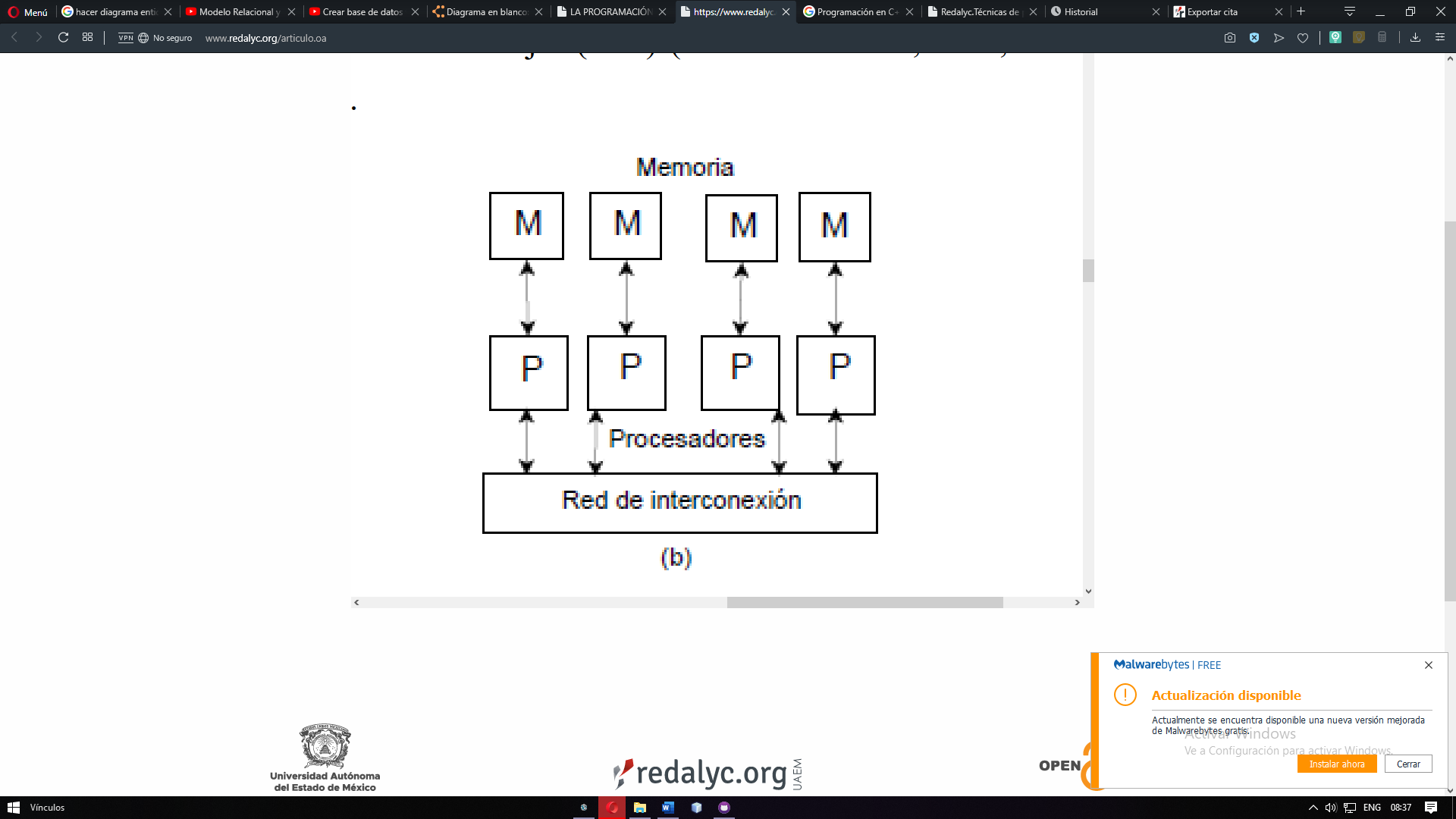
Para resolver este problema, surge el concepto de paralelismo, que consiste en usar múltiples recursos computacionales para resolver un problema, con el objetivo de lograr un menor tiempo de procesamiento computacional.

**Arquitecturas paralelas**

Memoria compartida: Todos los elementos de proceso acceden al mismo espacio de memoria común. El acceso a la memoria debe ser controlado por los propios algoritmos a través de diferentes métodos de sincronización. La comunicación entre los procesadores se realiza utilizando la memoria, en el momento en que un proceso escribe en espacio de memoria y este es leído por otro proceso.



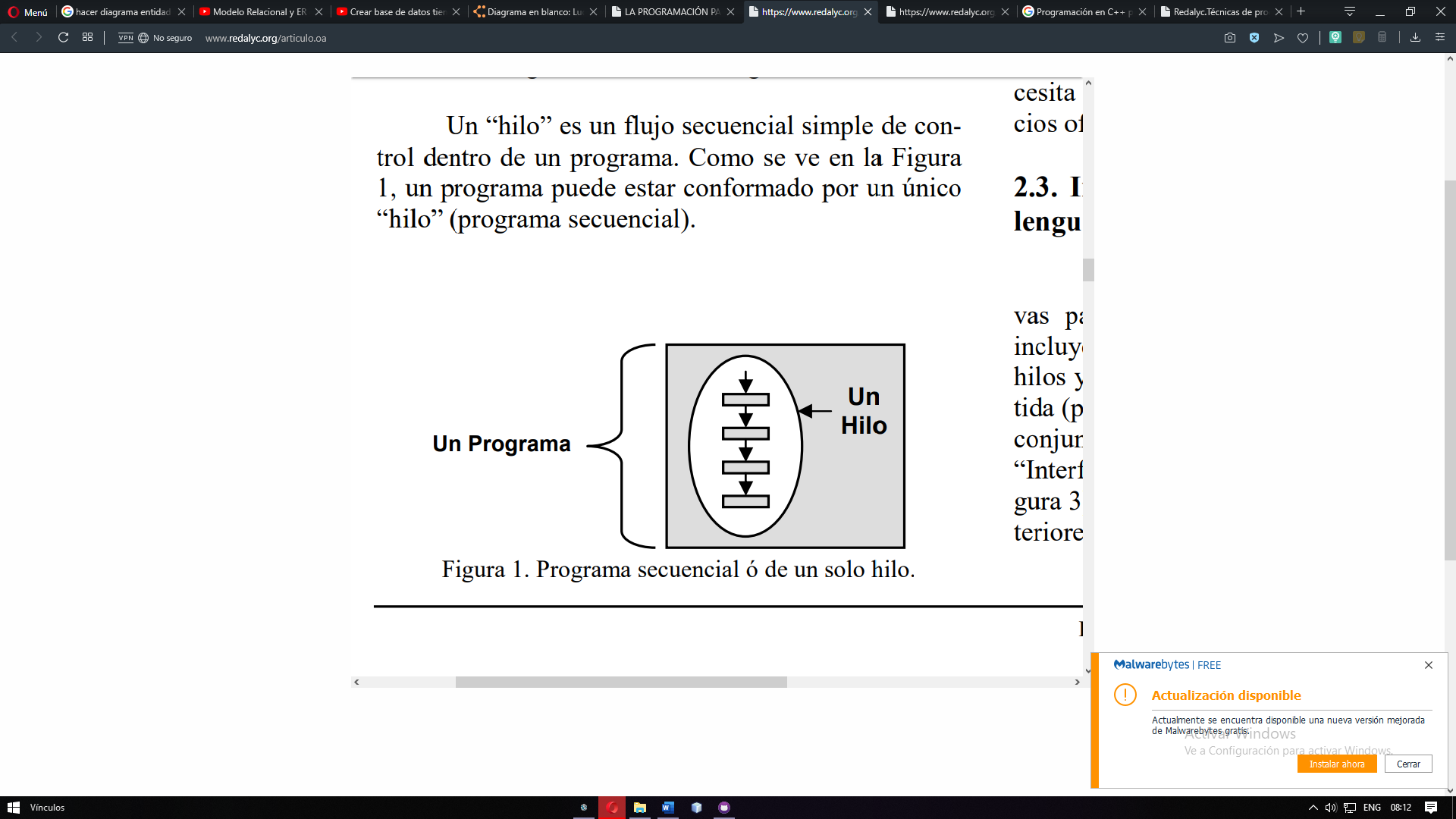
Arquitectura de memoria distribuida: Cada procesador está asociado a un espacio de memoria físico propio o local y existe una red de interconexión que une los procesos. La comunicación entre procesos se realiza mediante el mecanismo de paso de mensajes.



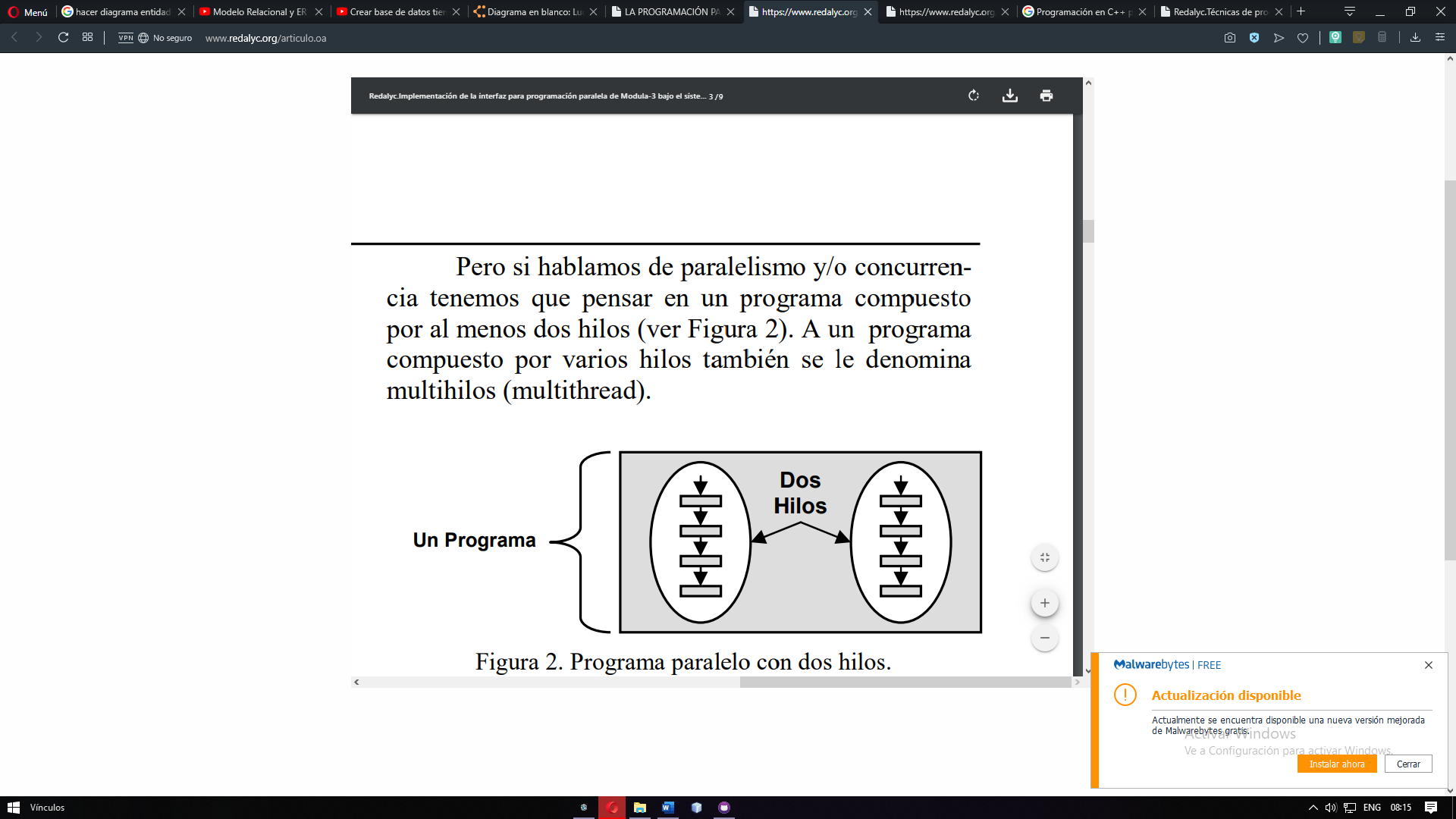
**Multihilos**

Para hacer paralelismo debemos hacer referencia a la unidad más básica para la ejecución de programas paralelos y/o concurrentes: esto es lo que se le denomina un hilo, trenza de ejecución, proceso ligero o por su nombre original en el idioma inglés “thread”.

Un hilo es un flujo secuencial simple de control dentro de un programa. Un programa puede estar conformado de un único hilo.



Pero para hablar de paralelismo se tiene que pensar que un programa debe estar compuesto por al menos 2 hilos. A un programa compuesto por dos hilos también se le denomina multihilos(multithread).



**Diseño de algoritmos paralelos**1. Particionamiento. Los cálculos se descomponen en pequeñas tareas. Usualmente es

independiente de la arquitectura o del modelo de programación. Un buen particionamiento

divide tanto los cálculos asociados con el problema como los datos sobre los cuales opera.

2. Comunicación. Las tareas generadas por una partición están propuestas para ejecutarse

concurrentemente pero no pueden, en general, ejecutarse independientemente. Los cálculos

en la ejecución de una tarea normalmente requerirán de datos asociados con otras tareas.

Los datos deben transferirse entre las tareas y así permitir que los cálculos procedan. Este

flujo de información se especifica en esta fase.

3. Aglomeración. Las tareas y las estructuras de comunicación definidas en las dos

primeras etapas del diseño son evaluadas con respecto a los requerimientos de ejecución y

costos de implementación. Si es necesario, las tareas son combinadas en tareas más

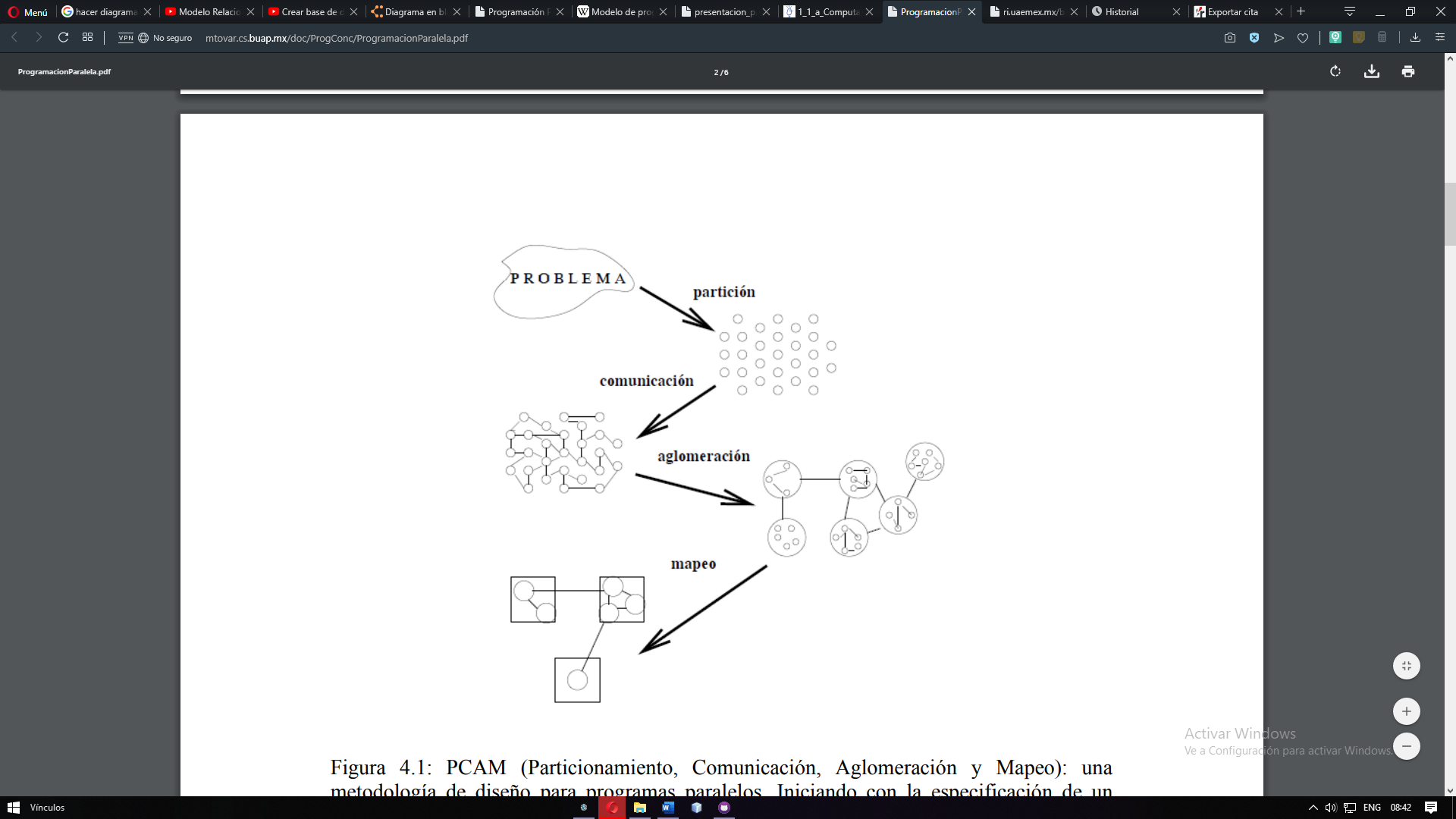
grandes para mejorar la ejecución o para reducir los costos de comunicación y

sincronización.

4. Mapeo. Cada tarea es asignada a un procesador de tal modo que intente satisfacer las

metas de competencia al maximizar la utilización del procesador y minimizar los costos de

comunicación.



Edwin Misael Vazquez Rivera s6