Universidad de Castilla-La Mancha

Escuela Superior de Informática

Diseño de Infraestructuras de Red

Grado en Ingeniería Informática
2019 - 2020

Práctica 1: Red Toroide e Hipercubo

Autor:
David Camuñas Sánchez

Fecha:
16 de marzo de 2020

Índice

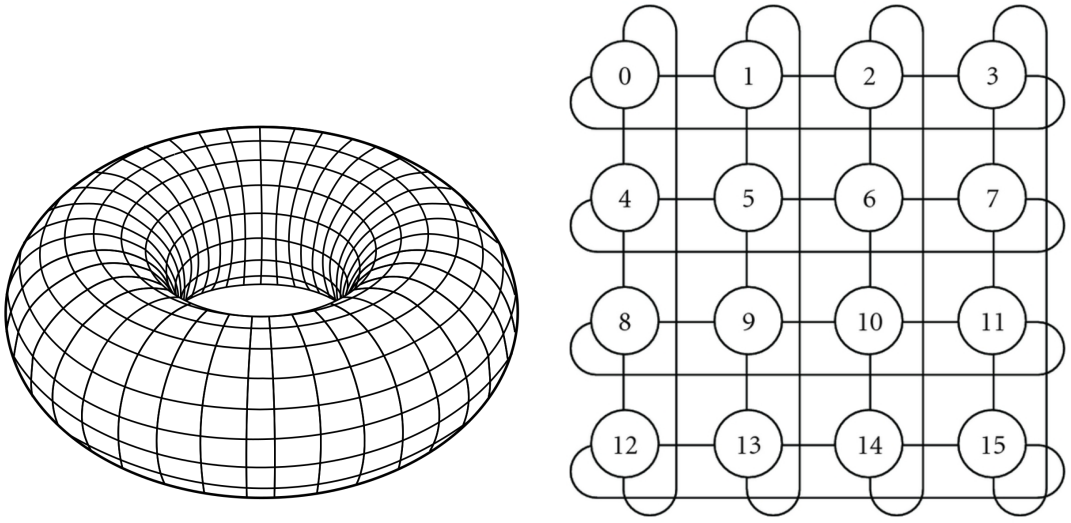
1. Introducción

El objetivo de esta primera fase de la práctica es la creación de uno de los tipos de redes de comunicación existente, esta es la ***red Toroide*** o *de anillo*.

A la hora de dibujar este tipo de red para entender su funcionamiento, se puede mostrar como una matriz.

Cuya peculiaridad es que son redes cuadradas, de *lado* ***L***, donde cada fila o columna de un extremo, conecta con su fila o columna inversa, es decir, los elementos que formarían la última fila de la matriz se conectarían (serían *"vecinos"*) con los elementos de la primera fila de la matriz y asi sucesivamente.

A continuación, se mostrará un ejemplo donde se puede observar la estructura de este tipo de red.



(a) Red toroide en 3D (b) Topología de una red Toroide

Figura 1: Imágenes Red Toroide

2. Planteamiento de la solución

Para solucionar el problema, se tiene en cuenta que una *red Toroidal* tiene un valor de ***lado L***. Donde el *número de procesos* **N** esta determinado por: L^2

En este caso el valor de **L** es **3**, por lo tanto, el valor de **N** será **9**, esto quiere decir que este toroide estaŕa formado por 9 procesos (denominados en el código como *ranks*).

El número de lado **L** del toroide se encuentra definido en */include/definitions*, como:

define L 3

Si se quiere realizar la simulación con un valor de lado distinto, se deberá cambiar el valor de esta constante. Además del número total de procesos a crear, pasado como argumento al comando de ejecución (*mpirun*) en la línea de ordenes. Este valor se puede encontrar en el *Makefile* del proyecto.

Otra constante importante, es la que determina el tamaño del buffer lectura del fichero (en este caso *datos.dat*), debido a que si se quiere leer una gran cantidad de números, y asi crear una gran cantidad de nodos (ranks), se debe de modificar su valor a uno mayor.

define MAX-SIZE 1024

2.1. Tipos de nodos

En este problema encontramos dos tipos de nodos: el ***rank o nodo 0*** y los demás ***nodos***.

- **Rank 0**: Este nodo corresponde al primer proceso creado. Encargado de la lectura del fichero ***datos.dat***, el cual contiene los números que más tarde asignara el mismo a los demás nodos que forman la red toroidal.

  A la hora de realizar la asignación de los números a los respectivos nodos restantes (incluyendose el mismo), debe de comprobar que la cantidad de números obtenidos del fichero es igual al tamaño del toroide **N** (nº de nodos que lo forman) si esta comprobación es exitosa, continuara la ejecución normal del programa.

  En caso contrario, a mi elección bien sea por que el tamaño del toroide o la cantidad de los números sea menor o mayor. El *rank 0* cero abortará la ejecución del programa. Tanto si la comprobación es correcta como si no, este difundirá el resultado a los demás nodos. Para ello se ha utilizado la función *Bcast()* de la libreria de **MPI**.

  Una vez asignados los respectivos números a cada nodo, tras calcular el número mínimo de la red toroidal, *rank 0* mostrara dicho número por pantalla, y el programa finalizará.

- **Los demás nodos**: Estos tipos de nodos recibirán del *rank 0* la decisión de continuar o no. Si continua la ejecución normal se les asignará un número real, el cual trás obtener cada uno sus respectivos vecinos, se llevará a cabo el algoritmo para obtener **el menor número de la red toroidal**.

Este documento ha sido generado con LaTeX utilizando la plantilla
desarrollada por José Ángel Martín Baos y disponible en
https://github.com/JoseAngelMartinB/PlantillaTrabajosLaTeX