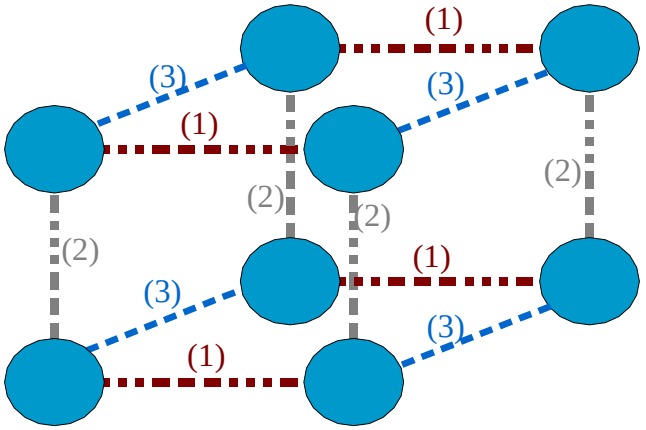
Ejercicio 6. Describir un algoritmo para resolver el problema de calcular si el número de ceros y de unos de una secuencia binaria es par o impar (tanto para el número de unos como para el número de ceros). Se asume que la secuencia se encuentra ya repartida equitativamente entre los procesadores y que el valor de N es conocido por todos los procesadores. Ilustrarlo con una secuencia de N=21 bits con P=4 y P=8. Se desea que el resultado se obtenga en todos los procesadores que intervienen en la computación. Modelar el tiempo de ejecución del algoritmo, suponiendo que el número de procesadores P es una potencia de 2, en función del tamaño de la secuencia binaria, N, y de los parámetros de comunicación ts (latencia) y tw (ancho de banda) de la arquitectura.

El problema equivale a sumar los dígitos de la secuencia binaria ya que si la suma es S, S es igual al número de unos y N-S sería el número de ceros. Por tanto, a partir de N y la suma S se determina si el número de ceros y unos es par/impar. Vamos a suponer que el reparto de los dígitos entre los prcesadores se hace por bloques, es decir, a cada procesador le corresponden ceil(N/P) elementos, y unos pocos procesadores tienen asignados menos dígitos.

Como P es potencia de 2, la suma se puede hacer en los mismos pasos que se describen en las diapositivas del Tema 1 sobre la suma N números en hipercubo. Los pasos son:

1. Cada proc. obtiene la suma de los dígitos que tiene asignados. Como mucho, un proceso calcula la suma de ceil(N/P) dígitos.

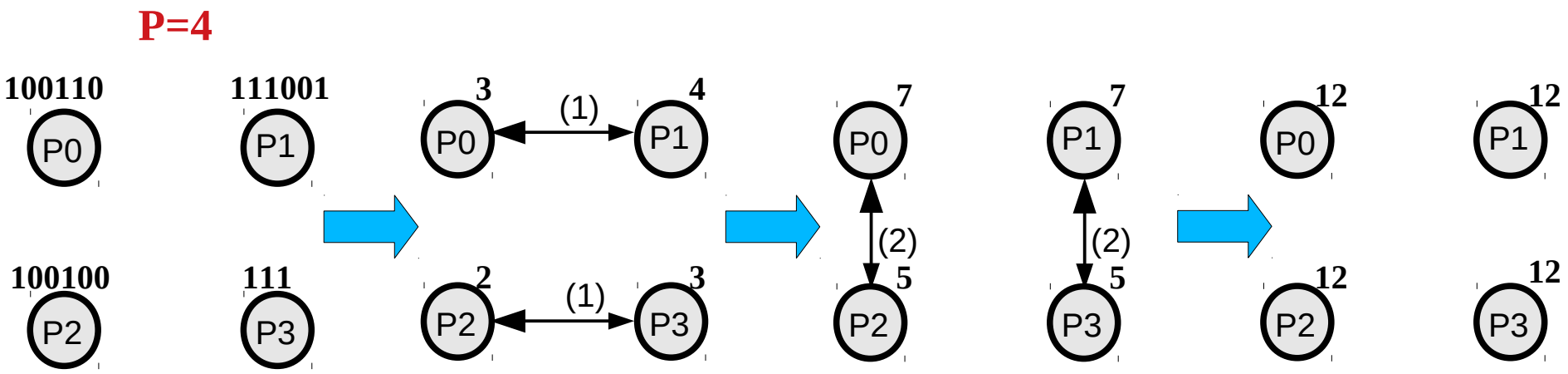
2. Para i=1,...,log(P), los procs. conectados en la i-ésima dimensión:
   - Intercambian sus resultados locales.
   - Actualizan el resultado sumando el valor local al valor obtenido.

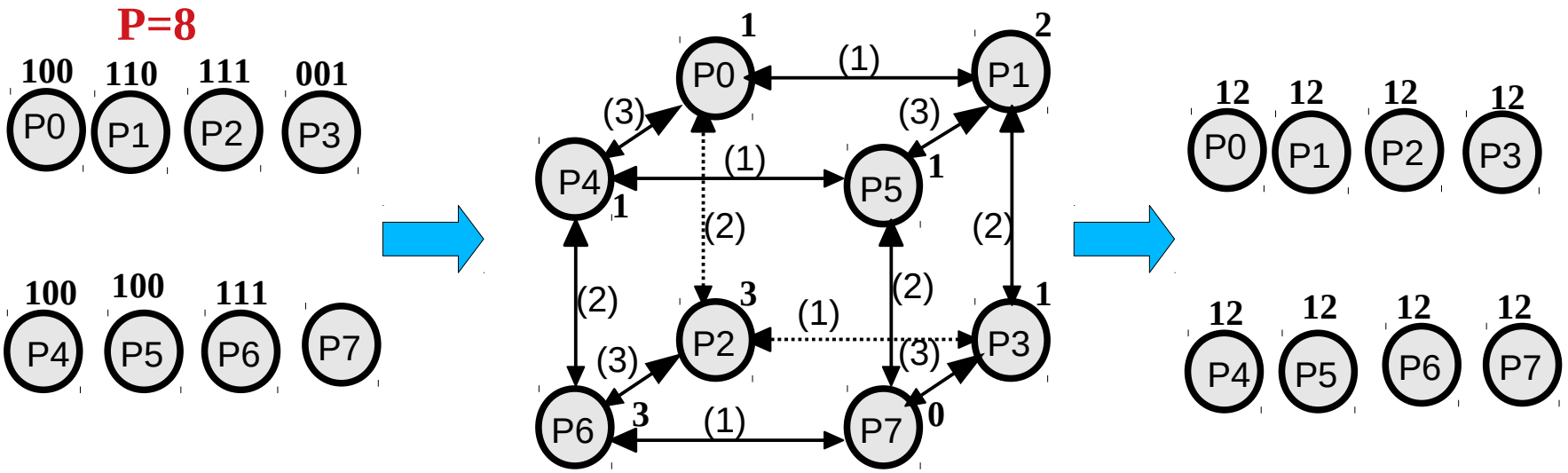


Si tc es el coste de la suma de dos dígitos, el tiempo de ejecución sería:

Tp = tc (ceil(N/P) – 1) + log P (tc + ts + tw)

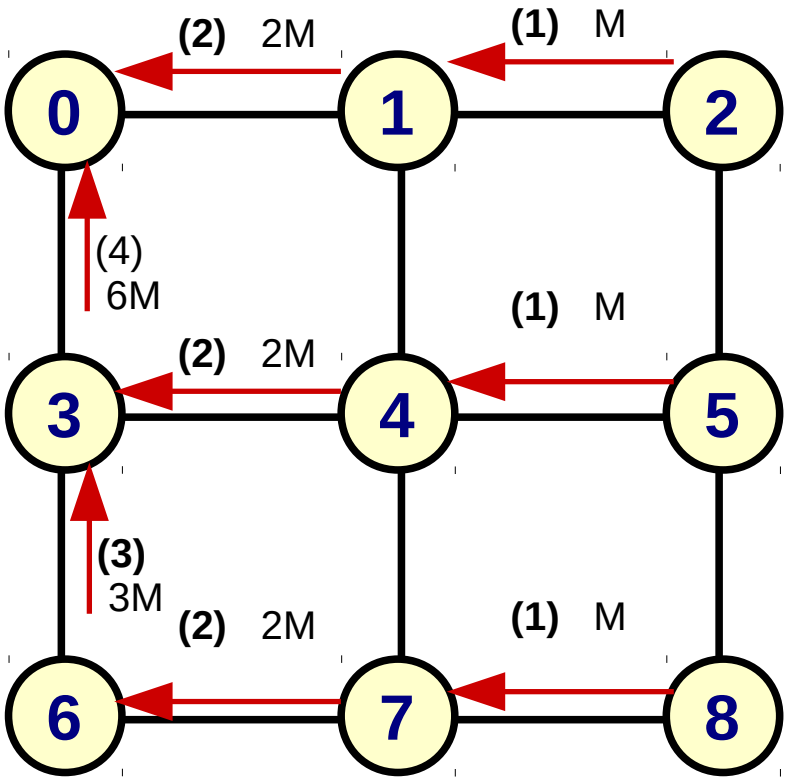
Supongamos que la secuencia es **100 110 111 001 100 100 111.** La ilustración del funcionamiento del algoritmo paralelo para P=4 y P=8 es:



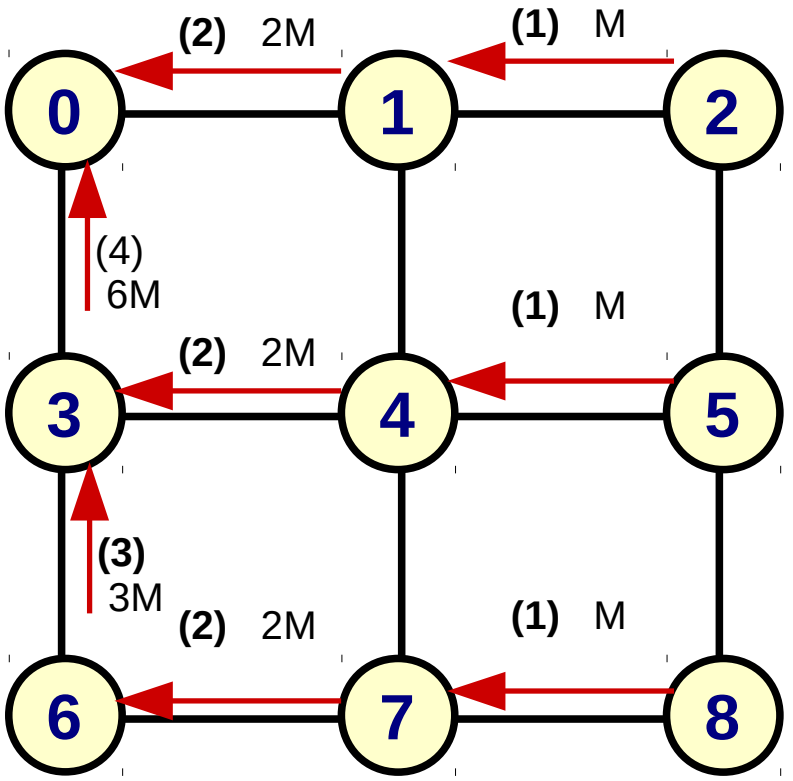


Ejercicio 7. Describir gráficamente cómo se podría implementar de forma eficiente una operación tipo MPI_Gather sobre P=9 procesadores conectados entre sí con una topología de interconexión malla cuadrada 3x3, asumiendo que el procesador 0 es el procesador raíz del Gather. (véase la figura de abajo). Para ello, mostrar en qué instante de tiempo se realizan los diferentes pasos de comunicación, etiquetando con instantes de tiempo (1, 2, 3,...) los enlaces donde se produce la comunicación, incluyendo también el tamaño de los mensajes que se transfieren. Se supone que cada procesador mantiene inicialmente un vector de M enteros y que al final de la operación el procesador 0 mantendrá un vector de 9M enteros

Para hacer un Gather de forma óptima, se puede recorrer la malla inicialmente en una dimensión, recogiendo los bloques de los procesos hasta llegar a los procesos de menor rango en dicha dimensión y después recorrer la dimensipón restante hasta llegar al proceso 0. Al final salen, para el caso de una malla 3x3, 4 pasos de comunicación donde se envían datos de diferente tamaño en cada paso. Solo se realizan envíos paralelos en las 2 primeras fases.



a) Obtener una fórmula de tiempo de ejecución para esta operación en función de los parámetros de comunicación ts (latencia) y tw (ancho de banda para envío de enteros) de la arquitectura.



Paso 1. Envío de M elementos con coste ts+Mtw
Paso 2. Envío de 2M elementos con coste ts+2Mtw
Paso 3. Envío de 3M elementos con coste ts+3Mtw
Paso 4. Envío de 6M elementos con coste ts+6Mtw

T(P=9, k=3)) = 4ts + (1+2+ 3+6) Mtw = 4ts + 12Mtw

b) Extender la fórmula para el caso en que P es el cuadrado de un entero positivo ($P=k^2$, con k entero positivo) y la malla tiene k filas y k columnas.

Siguiendo el mismo esquema, se comprueba que tenemos dos fases diferenciadas con (k-1) envíos cada una (2(k-1) envíos en total). En la primera fase se suben los datos por columnas hacia la fila superior y en la segunda se recogen los bloques avanzando hacia el proceso 0 desde la derecha. Así, para P= 16 (k=4), tenemos: T(P=16, k=4)) = 6ts + (1+2+3+ 4+8+12)Mtw = 4ts + 30Mtw

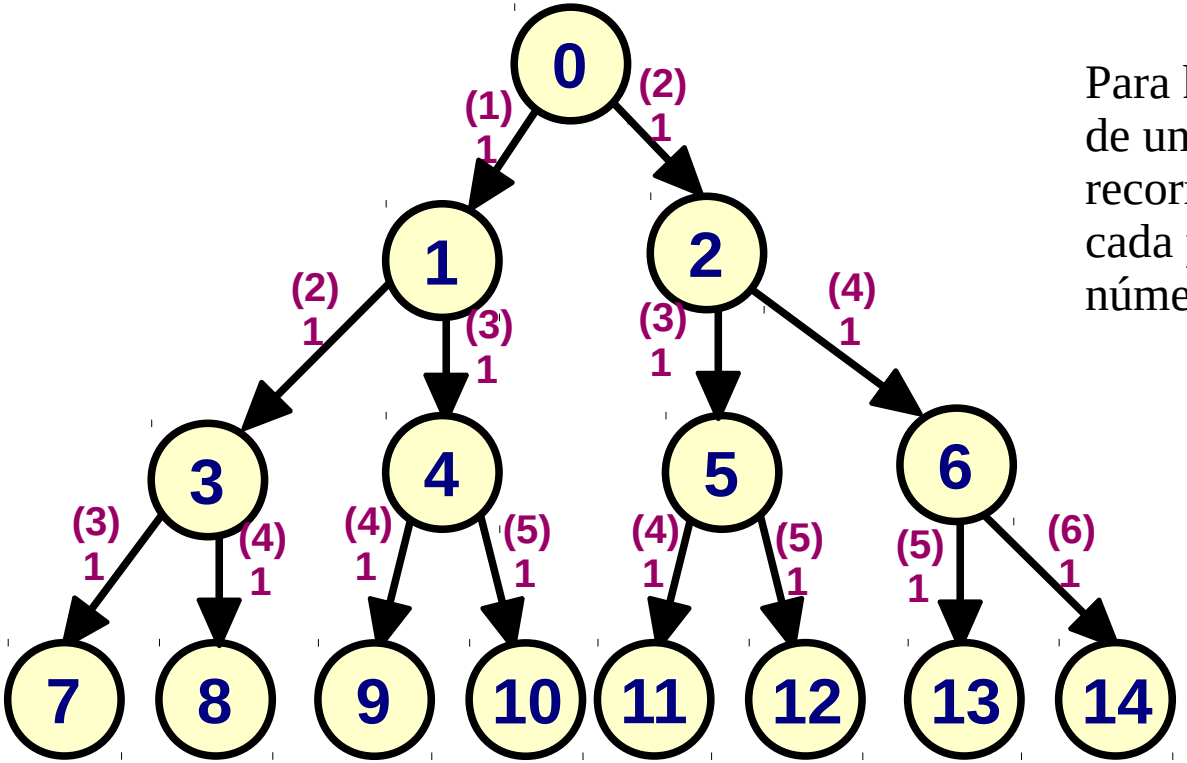
Se puede comprobar que la fórmula se puede generalizar para cualquier k de la siguiente forma:

$$T(P=k\times k)=2(k-1)t_s+t_w\left(M\sum_{i=1}^{k-1} i+kM\sum_{i=1}^{k-1} i\right)=2(k-1)t_s+M(k+1)t_w\sum_{i=1}^{k-1} i=2(k-1)t_s+M(k+1)t_w\sum_{i=1}^{k-1} i$$

$$T(P=k\times k)=2(k-1)t_s+M(k+1)t_w\frac{k(k-1)}{2}=2(k-1)t_s+Mt_w\frac{k(k^2-1)}{2}$$

Ejercicio 8. Se dispone de un algoritmo paralelo para calcular una aproximación de la integral de una función f en un intervalo [a,b] sobre P=15 procesadores. Obtener el tiempo de ejecución del algoritmo en función de N, P, ts, tw , tc1 y tc2 El algoritmo consta de los pasos siguientes:

a) Un procesador distinguido, que hace la función de procesador raíz, selecciona un entero N que es múltiplo de P. Este procesador raíz coopera con el resto de procesadores para difundir el valor de N a todos los procesadores, utilizando un esquema de difusión en árbol binario.



Para hacer un Broadcast en árbol binario de un elemento de forma eficiente, se recorren los nodos usando 6 pasos donde cada paso requiere el envío de un número entero. Por lo tanto:

$$T(Broadcast)=6(t_s+t_w)$$

b) Cada procesador selecciona N/P subintervalos disjuntos del intervalo [a,b] de tamaño d=(b-a)/N y calcula una aproximación a la integral en dichos subintervalos, utilizando la regla del trapecio. Se supone que el costo asociado al cálculo de la integral en un subintervalo es tc1. Cada procesador Pj calcula la suma local de las N/P aproximaciones calculadas. Podemos suponer que el costo de la suma aritmética de dos números reales es tc2.

$$T(fase\_cómputo) = (N/P)t_{c1} + (N/P-1)t_{c2}$$

c) La suma de las aproximaciones calculadas por cada procesador individual es obtenida en todos los procesadores mediante una operación de reducción todos a todos que sigue un esquema de hipercubo.

Con P=15 procesadores, no es posible formar un hipercubo que permita seguir esta topología de comunicación. Por ello, se debería incluir un proceso adicional que haga el papel de procesador virtual y que puede residir en el último procesador.Con este procesador, la reducción se puede hacer en Log_2 16 pasos, es decir, con 4 pasos donde cada paso incluye un intercambios sencillo de un entero y la suma de dos enteros. Por lo tanto, el tiempo de ejecución de la reducción sería:

$$T(\text{Reducción}) = \text{Log2}(P)(t_s+t_w+t_{c2}) = 4(t_s+t_w+t_{c2})$$