Tema2. Ejercicio 11. (a)

Versión 1	Versión 2

b=0; i=0;
while ((b==0)&& (i<M)) do {
  if (NP[i]==x) b=1;
  i++;
}
if (b==1) printf("%d ES primo", x);
else printf("%d NO es primo", x);
	b=0;
	for(i=0;i<M;i++) {
	  if (NP[i]==x) b=1;
	}
	if (b==1) printf("%d ES primo", x);
	else printf("%d NO es primo", x);

Tema2. Ejercicio 11. (a)

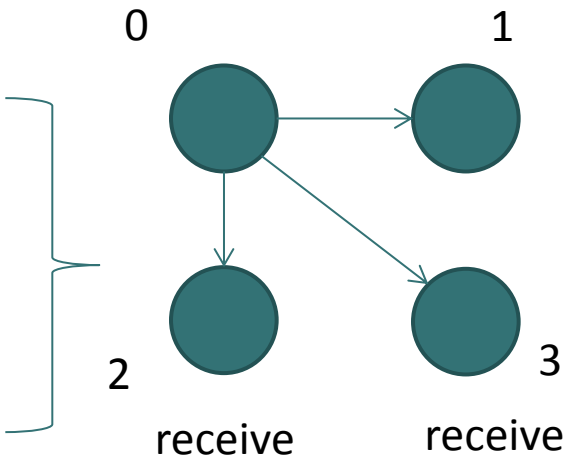
Versión 2

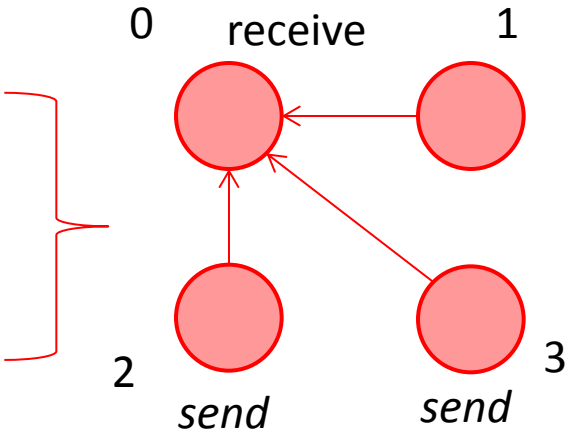
```
xr = sqrt(x);

if (xr>NP[M-1]) {
   print("%u supera el máximo primo a detectar (raíz %u) \n",
            x, NP[M-1]);
   exit(1);
}

for ( i=0 ;  (NP[i]<=xr) && (x % NP[i]!=0); i++) {} ; // % = módulo

if (NP[i]<=xr) printf("%u NO ES primo", x);
else printf("%u ES primo", x);
```

Tema2. Ejercicio 11. (b)

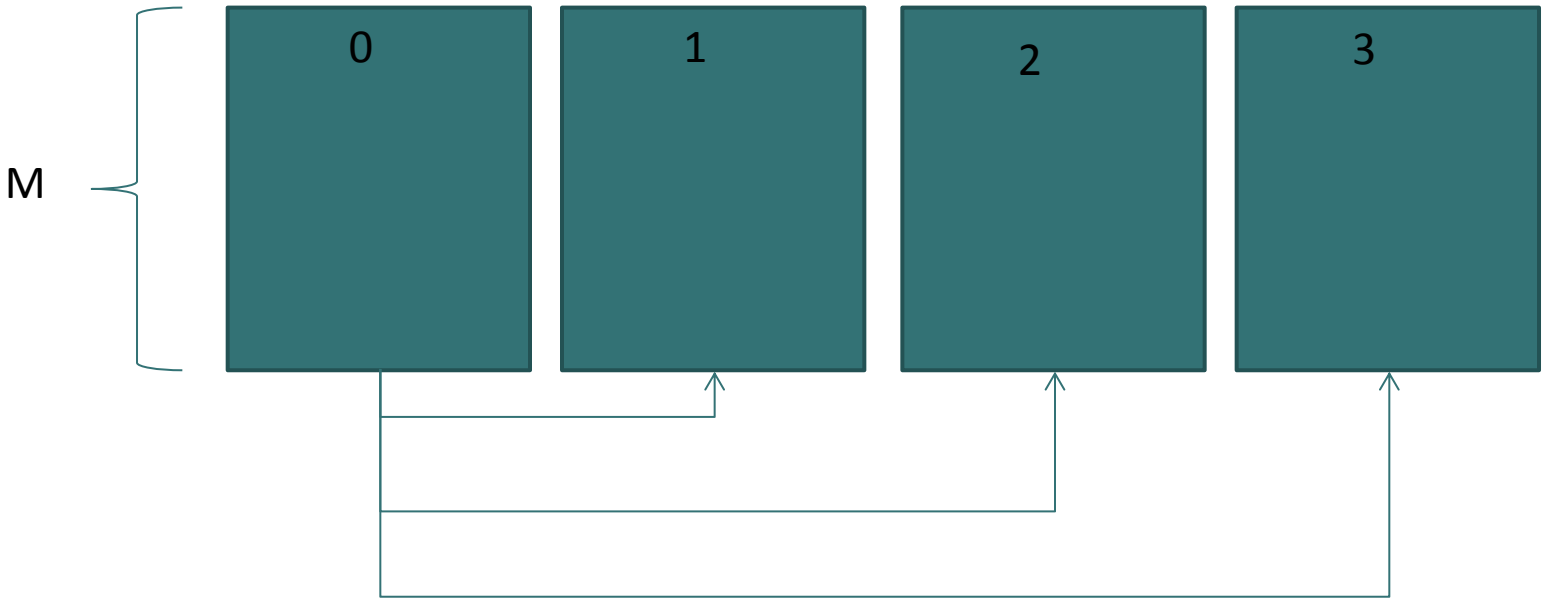
Versión 2. send/receive

```
//Difusión de x y NP
if (idproc==0)
  for (i=1; i<num_procesos; i++) {
     send(NP,M+1,tipo,i,grupo);   send(x,1,tipo,i,grupo);
  }
else {
  receive(NP,M+1,tipo,0,grupo);   receive(x,1,tipo,0,grupo);
}
 //Cálculo paralelo, asignación estática
b=0;
for(i=idproc; i<M; i=i+num_processos)     if (x==NP[i])  b=1;
//Comunicación resultados (reducción)
if (idproc==0)
  for (i=1; i<num_procesos; i++) {
     receive(baux,1,tipo,i,grupo);  b = b | baux;
  }
else  send(b,1,tipo,0,grupo);
 //Proceso 0 imprime resultado
if (idproc==0)    if (b!=0)   printf("%u ES primo", x);
  else  printf("%u NO es primo", x);
```





Tema2. Ejercicio 11. (b)

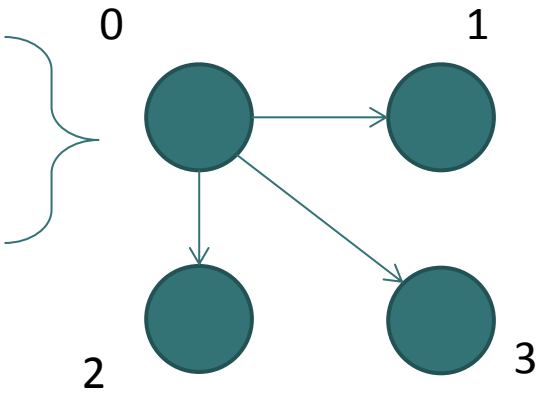


nprod=4	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
iprod=0	**0**	1	2	3	**4**	5	6	7	**8**	9	10	11	**12**	13	14	15
iprod=1	0	**1**	2	3	4	**5**	6	7	8	**9**	10	11	12	**13**	14	15
iprod=2	0	1	**2**	3	4	5	**6**	7	8	9	**10**	11	12	13	**14**	15
iprod=3	0	1	2	**3**	4	5	6	**7**	8	9	10	**11**	12	13	14	**15**

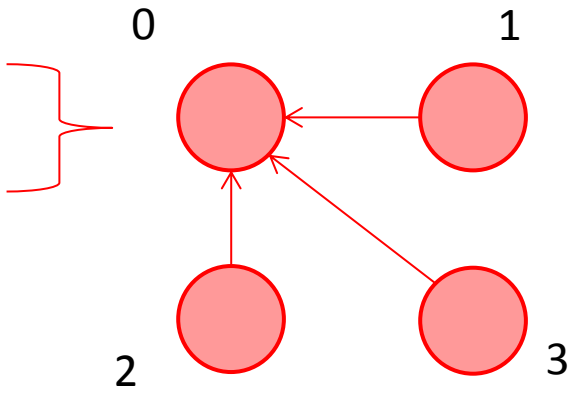
Tema2. Ejercicio 12

Versión 2. broadcast y reduction

```
//difusión del vector NP y de x
broadcast(NP, M,tipo,0,grupo);
broadcast(x, 1,tipo,0,grupo);

//Cálculo paralelo, asignación estática
b=0;
for(i=idproc; i<M; i=i+num_processos)
     if (x==NP[i])  b=1;

//Comunicación resultados (reducción)
reduction(b, b,1,tipo,OR,0,grupo);

//Proceso 0 imprime resultado
if (idproc==0)
  if (b==1)     printf("%u ES primo", x);
  else   printf("%u NO es primo", x);
```



Broadcast desde 0



Reducción en 0

Tema2. Ejercicio 12

Versión 2. broadcast y reduction

```
//difusión del vector NP y de x
broadcast(NP, M,tipo,0,grupo);
broadcast(x, 1,tipo,0,grupo);


//Cálculo paralelo, asignación estática
b=0;
for(i=idproc; i<M; i=i+num_processos)
     if (x==NP[i])  b=1;

//Comunicación resultados (reducción)
reduction(b, b,1,tipo,OR,0,grupo);


//Proceso 0 imprime resultado
if (idproc==0)
  if (b==1)     printf("%u ES primo", x);
  else  printf("%u NO es primo", x);
```

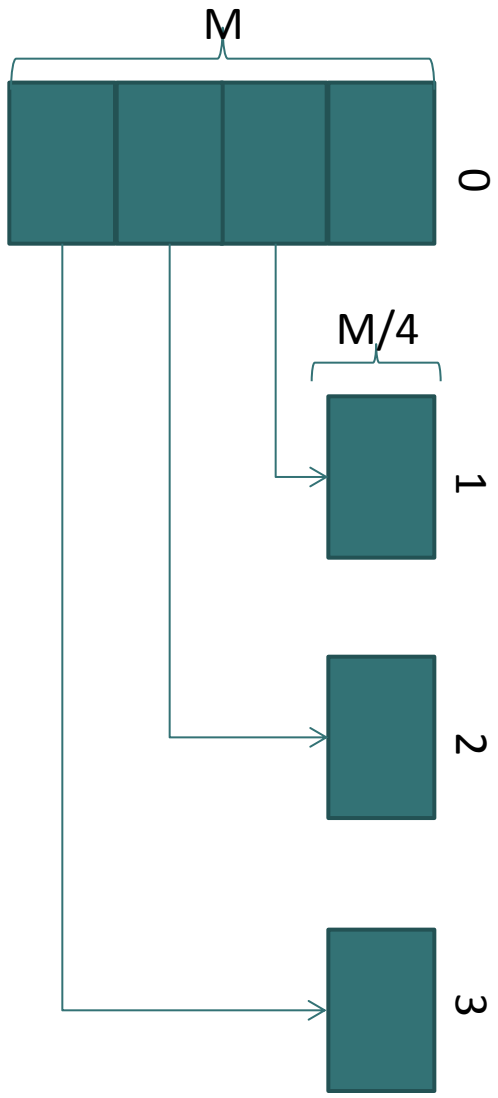
Versión 2. send/receive

```
//Difusión de x y NP
if (idproc==0)
  for (i=1; i<num_procesos; i++) {
     send(NP,M+1,tipo,i,grupo);   send(x,1,tipo,i,grupo);
  }
else {
  receive(NP,M+1,tipo,0,grupo);   receive(x,1,tipo,0,grupo);
}
 //Cálculo paralelo, asignación estática
b=0;
for(i=idproc; i<M; i=i+num_processos)     if (x==NP[i])  b=1;
//Comunicación resultados (reducción)
if (idproc==0)
  for (i=1; i<num_procesos; i++) {
     receive(baux,1,tipo,i,grupo);  b = b | baux;
  }
else  send(b,1,tipo,0,grupo);
 //Proceso 0 imprime resultado
if (idproc==0)    if (b!=0)   printf("%u ES primo", x);
  else  printf("%u NO es primo", x);
```

Tema2. Ejercicio 13

Versión 2. broadcast, reduction, scatter

```
//difusión del vector NP y de x
scatter(NP, M, NPl, M/num_procesos,tipo,0,grupo);
broadcast(x, 1,tipo,0,grupo);

//Cálculo paralelo, asignación estática
b=0;
for(i=idproc;i<M;i=i+num_processos)
     if (x==NP[i])  b=1;

//Comunicación resultados (reducción)
reduction(b, b,1,tipo,OR,0,grupo);

//Proceso 0 imprime resultado
if (idproc==0)
  if (b==1) printf("%u ES primo", x);
  else  printf("%u NO es primo", x);
```



Tema2. Ejercicio 14

Versión 1

```
b=0; i=0;
//solo un thread escribe en b
#pragma omp parallel
{  int ilocal;
  #pragma omp critical
      { ilocal=i; i++; }
  while ((b==0) && (ilocal<M)) do {
      if (NP[ilocal]==x) b=1;
     #pragma omp critical
        { ilocal=i; i++; }
    }
}
if (b==1) printf("%u ES primo", x);
else printf("%u NO es primo", x);
```

Versión 2

```
b=0;
//solo un thread escribe en b
#pragma omp parallel for reduction (|:b)
for (i=0;i<M;i++)  {
  if (NP[i]==x) b=1;
}

if (b==1) printf("%u ES primo", x);
else printf("%u NO es primo", x);
```