TEMA 2

Problema 11

NP contiene números primos (de 1 a 1000 en este ejemplo)

2	3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61	67
71	73	79	83	89	97	101	103	107	109	113	127	131	137	139	149	151	157	163
167	173	179	181	191	193	197	199	211	223	227	229	233	239	241	251	257	263	269
271	277	281	283	293	307	311	313	317	331	337	347	349	353	359	367	373	379	383
389	397	401	409	419	421	431	433	439	443	449	457	461	463	467	479	487	491	499
503	509	521	523	541	547	557	563	569	571	577	587	593	599	601	607	613	617	619
631	641	643	647	653	659	661	673	677	683	691	701	709	719	727	733	739	743	751
757	761	769	773	787	797	809	811	821	823	827	829	839	853	857	859	863	877	881
883	887	907	911	919	929	937	941	947	953	967	971	977	983	991	997			

TEMA 2. Problema 11

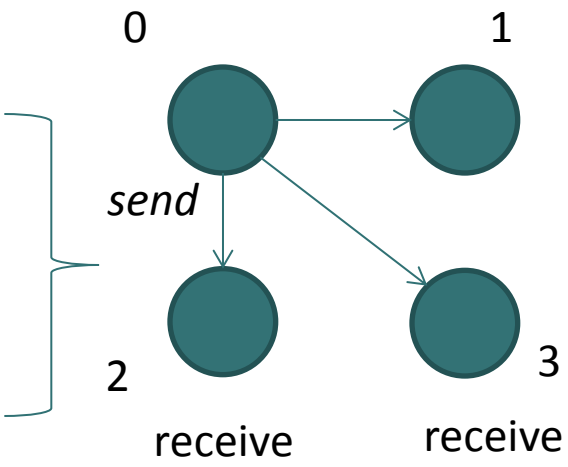
Versión 1 A	Versión 1 B

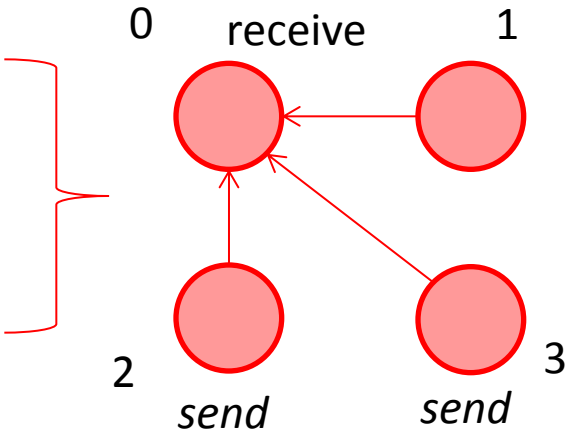
if (x>NP[M-1]) {
print("%u supera el máximo primo a detectar %u \n", x, NP[M-1]);
exit(1);
}
b=0;
for (i=0; i<N;++)
if (NP[i]==x) b=1;
 	
if (x==NP[i])
printf("%u ES primo\n", x);
else printf("%u NO ES primo\n", x);
	if (x>NP[M-1]) {
	print("%u supera el máximo primo a detectar %u \n", x, NP[M-1]);
	exit(1);
	}
	
	for (i=0; NP[i]<x;i++) {};
	
	if (x==NP[i])
	printf("%u ES primo\n", x);
	else printf("%u NO es primo\n", x);

Versión 1

```
if (x>NP[i]) {
   print("%u supera el máximo primo a detectar %u \n", x, NP[M-1]);
   exit(1);
}
//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
i=id_proc; NP[M+i]=x+1;
while (NP[i]<x) do  i=i+num_procesos;

b=(x==NP[i])?1:0;

//Comunicación resultados
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);
```

TEMA 2. Problema 11

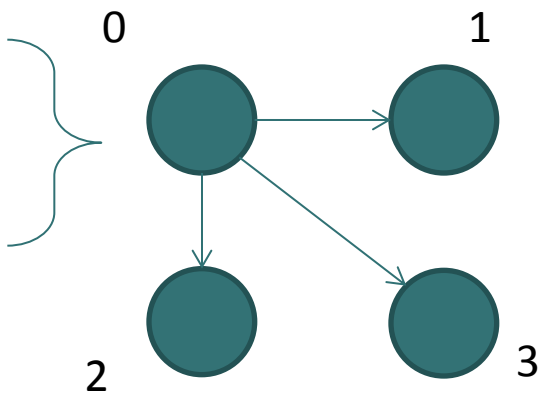




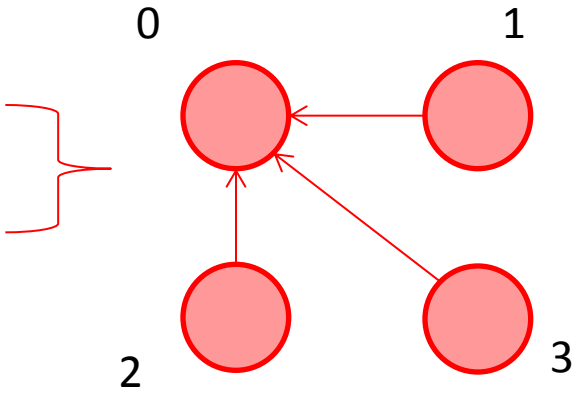
Versión 1

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


//Cálculo
i=id_proc; NP[M+i]=x+1;
while (NP[i]<x) do  i=i+num_procesos;
b=(x==NP[i])?1:0;
//Comunicación resultados
reduction(b, b,1,tipo,OR,0,grupo);


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

TEMA 2 Problema 12



Broadcast desde 0

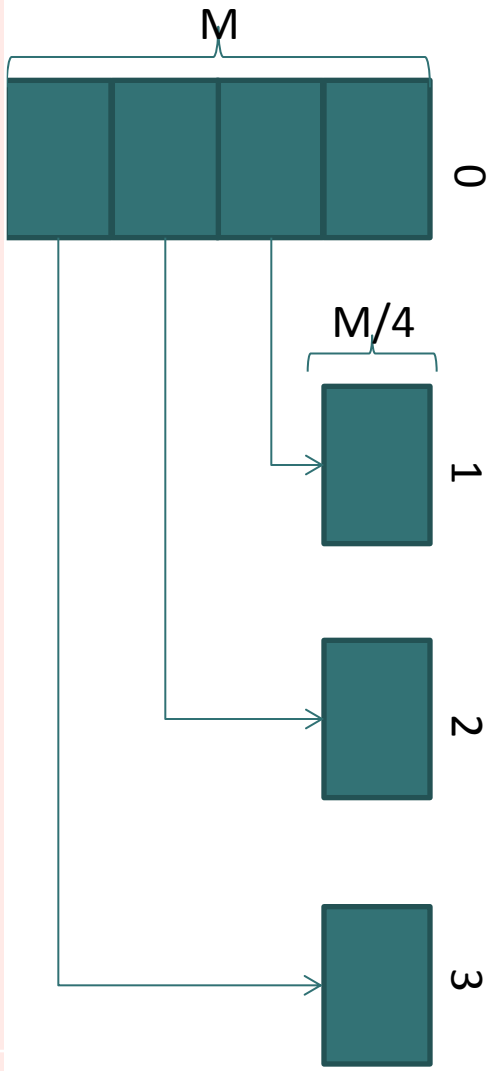


Reducción en 0

TEMA 2. Problema 13

Versión 1

```
//difusión del vector NP y de x
scatter(NP, M, NPl, M/num_procesos,tipo,0,grupo);
broadcast(x, 1,tipo,0,grupo);
//Cálculo
i=id_proc; NP[M/num_procesos]=x+1;
while (NP[i]<x) do  i=i+1;
b=(x==NP[i])?1:0;
//Comunicación resultados
reduction(b, b,1,tipo,OR,0,grupo);
if (idproc==0)
   if (b) printf("%u ES primo", x);
   else printf("%u NO es primo", x);
```



TEMA 2 Problema 14

Versión 1A

```
 b=0;
#pragma omp parallel for
for (i=0;i<M;i++)  {
   if (NP[i]==x) b=1;
}


if (b) printf(“%u ES primo”, x);
else printf(“%u NO es primo”, x);
```