Algorithme [6]:

La mémoire fonctionne sur le principe d'une mémoire paginée. Lors d' un défaut de page, le processeur s'adresse à un **handler** qui recherche la page demandée auprès des autres processeurs du réseau. Le **mappeur** est la partie du système attachée à un processeur qui répond aux demandes de pages provenant des autres processeurs du réseau. Les mappeurs communiquent entre eux pour maintenir la cohérence entre les différentes copies des pages.

Les communications sont fiables: pas de perte de messages, pas de duplication de messages, pas de corruption de messages, pas de déséquencement des messages, ou alors, le protocole de transport est capable de le détecter et de le corriger.

Un processeur effectue une invalidation lorsqu'il devient le propriétaire d'une page suite à un défaut en écriture. Il effectue alors une demande d'invalidation. L'invalidation consiste à envoyer à l'ensemble des sites (copyset) qui ont une copie de la page une requête pour qu'ils changent le mode d'accès de celle-ci à "nil". Cette demande s'effectue à l'aide de la primitive :

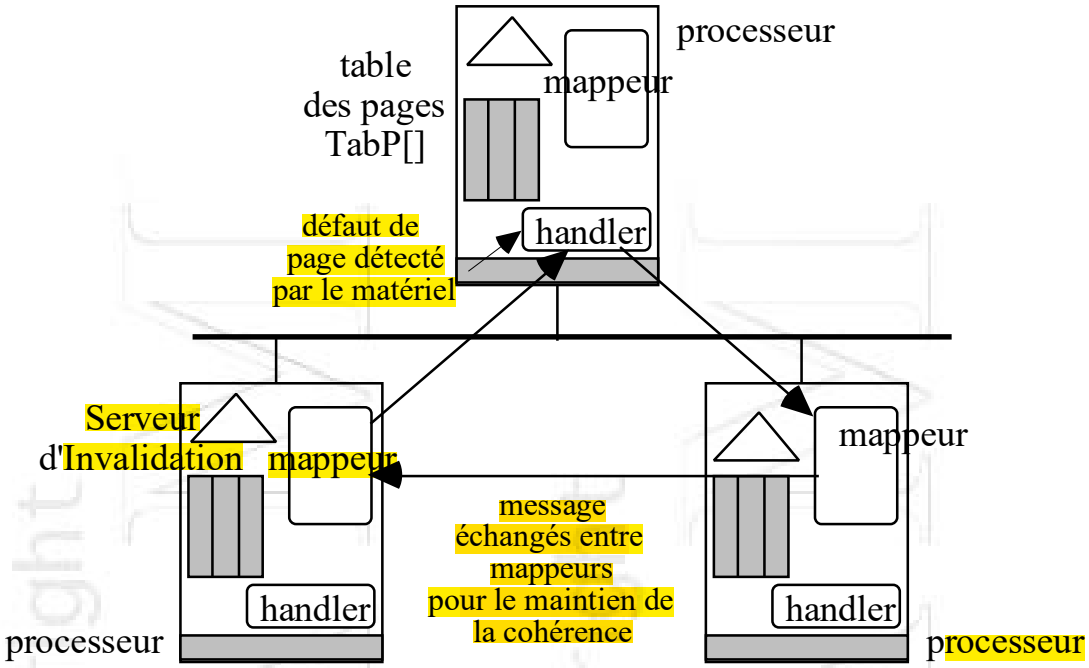
Invalider(page, ensemble de processeurs destinataires).

La réception et l'exécution de la requête d'invalidation sur un processeur est effectué par le **serveur d'invalidation** propre à celui-ci.

Les informations que maintiennent les mappeurs sur les pages sont contenues dans une table **TabP[]** , table des pages qui contient les champs suivants :

- **access** représente le mode d'accès à la page : **read**, **write**, **nil**,
- **copyset** contient l'ensemble des processeurs qui ont une copie (en lecture) de la page,
- **lock** joue le rôle de variable sémaphore pour l'accès en exclusion mutuelle à la page pendant un défaut, deux primitives sont utilisées :
  LOCK() pour tester le sémaphore et éventuellement bloquer le demandeur,
  UNLOCK() pour déverrouiller l'accès,
- **probOwner** indique un propriétaire estimé (probable) de la page.

Le schéma page suivante illustre ces éléments.



L'algorithme de gestion distribuée de la mémoire virtuelle répartie est le suivant :

Handler de défaut de page en lecture :

LOCK (TabP[p].lock);
Demander à TabP[p].probOwner l'accès en lecture pour la page p;
TabP[p].probOwner := processeur qui a répondu;
TabP[p].access := read;
UNLOCK (TabP[p].lock);

Mappeur en lecture:

```
LOCK ( TabP[p].lock);
IF je suis le propriétaire          THEN BEGIN
        TabP[p].copyset := TabP[p].copyset U {processeur demandeur};
        TabP[p].access := read;
        Envoyer (p au processeur demandeur);
        END
                                    ELSE BEGIN
        Rediriger la requête vers le processeur (TabP[p].probOwner);
        TabP[p].probOwner := processeur demandeur;
        END;
UNLOCK ( TabP[p].lock);
```

Handler de défaut de page en écriture :

```
LOCK ( TabP[p].lock);
Demander à TabP[p].probOwner l'accès en écriture pour la page p;
Invalider (p, TabP[p].copyset);
TabP[p].probOwner := moi-même;
TabP[p].access := write;
TabP[p].copyset := {Ø};
UNLOCK ( TabP[p].lock);
```

Mappeur en écriture:

```
LOCK ( TabP[p].lock);
IF je suis le propriétaire          THEN BEGIN
        TabP[p].access := nil;
        Envoyer (p et TabP[p].copyset au processeur demandeur);
        TabP[p].probOwner := processeur demandeur;
        END
                                    ELSE BEGIN
        Rediriger la requête vers le processeur (TabP[p].probOwner);
        TabP[p].probOwner := processeur demandeur;
        END;
UNLOCK ( TabP[p].lock);
```

Serveur d'Invalidation :

```
TabP[p].access := nil;
TabP[p].probOwner := processeur demandeur;
```

On forme le graphe des propriétaires probables pour une page p de la façon suivante:

Pi->Pj si TabP[p].probOwner de Pi := Pj.