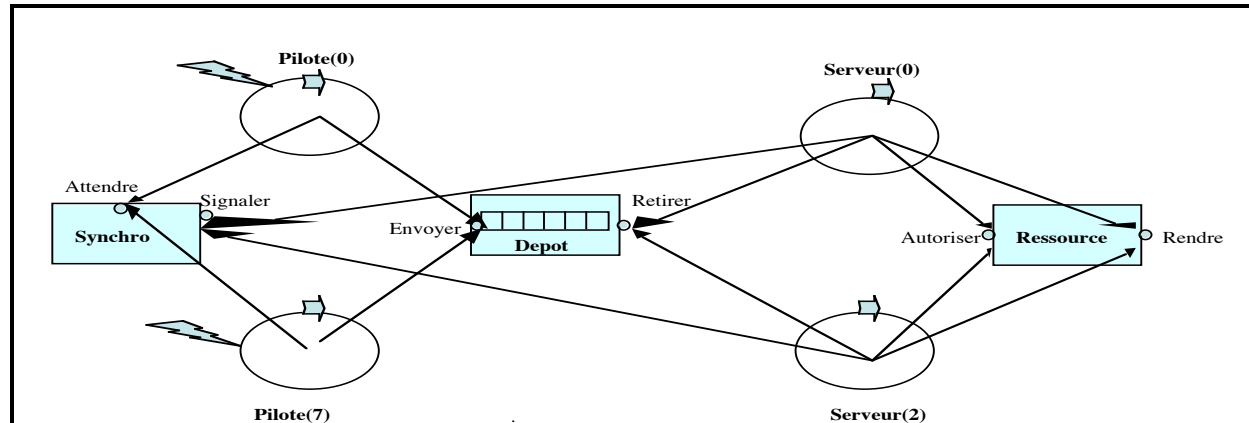


ÉTUDE DE CAS AVEC PARADIGMES : SURVEILLANCE PAR WEBCAMS

Un site Web, avec 8 appareils webcam de surveillance, récupère des images avec des serveurs connectés à des clients.



```
type Id_Cam is mod 8 ; -- les types de données
type Id_S is mod 3 ;
```

```
type Image is
  record
    Cam : Id_Cam ;
    Date : Time ;
    Paysage : Bitmap ;
  end record ;
```

```
-- les processus coopérants
```

```
task type Un_Pilote ;
  Pilote : array(Id_Cam) of Un_Pilote ;
```

```
task type Un_Serveur ;
  Serveur : array(Id_S) of Un_Serveur ;
```

```
-- les paquetages de coopération
```

```
package Synchro is
  procedure Attendre(X : in Id_Cam);
  procedure Signaler(X : in Id_Cam);
end Synchro;
```

```
package Ressource is
  procedure Demander(X : in Id_Cam);
  procedure Rendre(X : in Id_Cam);
end Ressource;
```

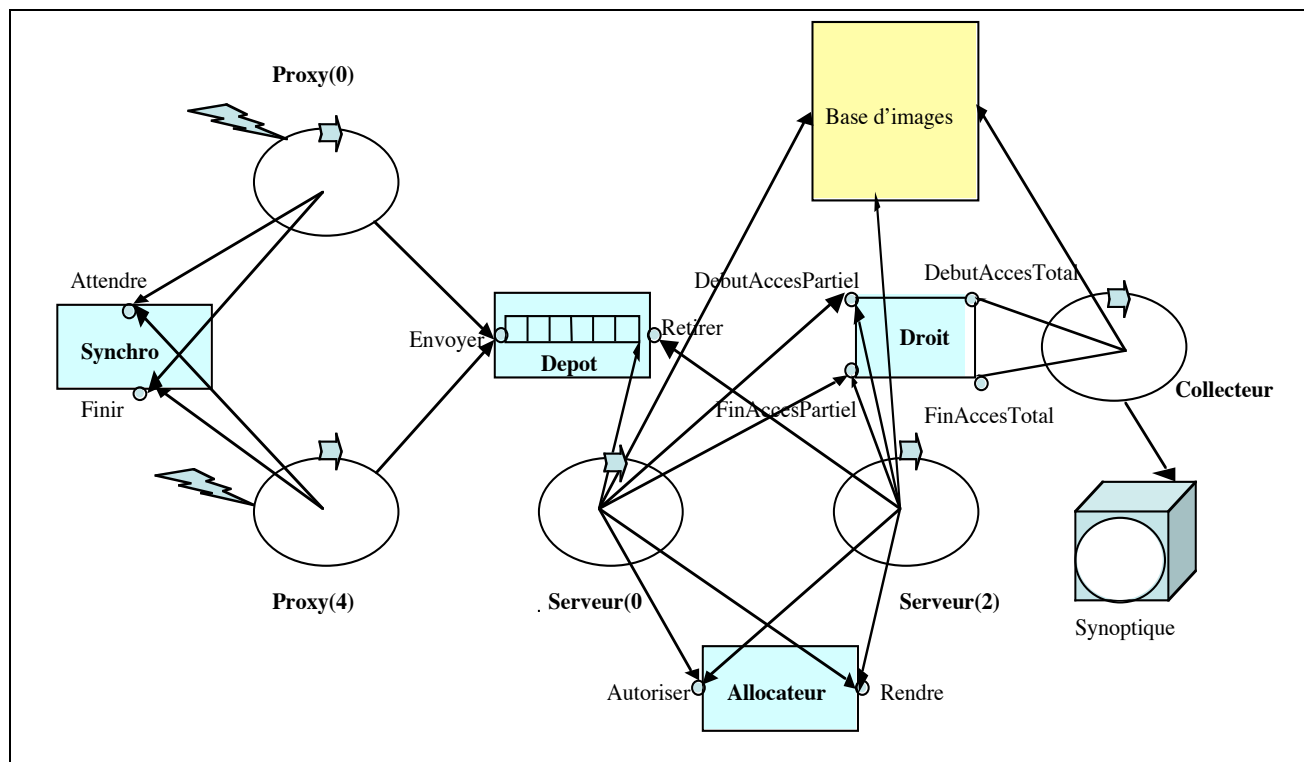
```
package Depot is
  procedure Envoyer(X : in Id_Cam ; Y : in Image);
  procedure Retirer(X : in Id_Cam ; Y : out Image);
end Depot;
```

```
task body Un_Pilote is  
  X : Id_Cam := Nom.UniqueCam; -- chaque pilote reçoit un nom unique qui le distingue des autres pilotes  
  Y : Image; Z : Bitmap ;  
begin  
  loop  
    Synchro.Attendre(X) ; -- synchronisation pour attendre le signal envoyé à X par un serveur  
    Lecture_de_la_Camera (Z) ; -- acquisition de l'image Bitmap par un accès réseau local  
    Y.Cam := X ; Y.Date := Clock ; Y.Paysage := Z ; -- préparation de Y  
    Depot.Envoyer(X, Y) ; -- envoi vers les Serveurs de l'image Y prise par X  
  end loop ;  
end Un_Pilote ;
```

```
task body Un_serveur is  
  I : Id_S := Nom.UniqueServeur ; ; -- chaque serveur reçoit un nom unique qui le distingue des autres serveurs  
  A, B, C : Id_Cam; Y, Z, T : Image ; -- consultation de 3 webcam  
begin  
  loop  
    Negociation_Avec_Le_Client(A, B, C) ; -- le client donne 3 noms de webcam  
    Ressource.Demander(A) ; Ressource.Demander(B) ; Ressource.Demander(C) ; -- réservation des webcam  
    Synchro.Signaler(A) ; Synchro.Signaler(B) ; Synchro.Signaler(C) ; -- réveil des pilotes  
    Depot.Retirer(A, Y) ; -- retrait de l'image de A qui est dans le dépôt et copie dans Y  
    Depot.Retirer(B, Z) ; -- retrait de l'image de B qui est dans le dépôt et copie dans Z  
    Depot.Retirer(C, T) ; -- retrait de l'image de C qui est dans le dépôt et copie dans T  
    Ressource.Rendre(A) ; Ressource.Rendre(B) ; Ressource.Rendre(C) ; -- libération des webcam  
  end loop ;  
end Un_Serveur ;
```

ÉTUDE DE CAS : SYNOPTIQUE POUR SUIVRE PLUSIEURS USINES

Une société qui possède 5 usines installe dans une salle de conseil un écran synoptique pour suivre leur activité



<pre>-- les types de données type Id_Usine is mod 5 ; -- noms des usines ----- type Id_S is mod 3 ; -- noms des serveurs</pre>	<pre>type Rapport is record Usine : Id_Usine ; Date : time ; Info : Donnees ; -- texte et images end record ;</pre>
---	---

<pre>-- les processus coopérants task type Un_Proxy ; Proxy : array(Id_Usine) of Un_Proxy ;</pre>	<pre>package Depot is procedure Envoyer(X : in Rapport); procedure Retirer(X : out Rapport); end Depot;</pre>
<pre>task type Un_Serveur ; Serveur : array(Id_S) of Un_Serveur ; task Collecteur ;</pre>	<pre>package Allocateur is procedure Autoriser(I : in Id_S ; X : in Integer); procedure Rendre(I : in Id_S ; X : in Integer); end Allocateur;</pre>
<pre>-- les paquetages de coopération package Synchro is procedure Attendre(X : in Id_Usine); procedure Finir(X : in Id_Usine); end Synchro;</pre>	<pre>package Droit is procedure Debut_Acces_Partiel(X : in Id_Usine); procedure Fin_Acces_Partiel(X : in Id_Usine); procedure Debut_Acces_Total; procedure Fin_Acces_Total; end Droit;</pre>

```
task body Un_Proxy is
    X : Id_Usine ; Y : Rapport ;
begin
    loop
        transfert_Du_Rapport_De_X(Y) ;    -- réception du rapport par un accès réseau
        Synchro.Attendre(X) ; -- synchronisation entre les proxys pour fixer l'ordre des dépôts : attend son tour
        Depot.Envoyer(Y) ;    -- maintenant l'envoi du rapport Y est possible
        Synchro.Finir(X) ;    -- permet de passer la main au successeur de X dans l'ordre fixé par Synchro
    end loop ;
end Un_Proxy ;
```

```
task body Un_serveur is  
  I : Id_S ; X : Integer; Y : Rapport ; Z : Image ;  
begin  
  I := Numero_Unique_Dans_Id_S ;      -- nom unique du Serveur, permet de le distinguer des autres serveurs  
  loop  
    Depot.Retirer(Y) ; -- retrait du plus ancien rapport présent dans le dépôt  
    X := Integer(Y.Usine) ; -- récupère le numéro d'usine et le convertit en un entier  
    -- si X > 0, demande le droit de prendre X pages de mémoire supplémentaires  
    if X > 0 then Allocateur.Autoriser(I, X) ; end if;  
    Calculs_Statistiques_Et_Preparation_Image(Z);    -- utilise un progiciel spécifique  
    Droit.Debut_Acces_Partiel(X); -- demande le droit d'accéder à l'image Z de X dans la base  
    Mise_A_Jour_De_Image_De_X_Dans_La_Base ;    -- accès à Z, réservé pour X  
    Droit.Fin_Acces_Partiel(X);                -- sort de cet accès  
    if X > 0 then Allocateur.Rendre(I, X); end if; -- n'a plus besoin de la mémoire supplémentaire  
  end loop ;  
end Un_Serveur ;
```

```
task body Collecteur is  
begin  
  loop  
    Attendre_Le_Reveil_Periodique ;      -- déclenchement externe  
    Droit.Debut_Acces_Total ; -- demande le droit d'accéder à la base en exclusion mutuelle  
    Travail_Dans_La_Base ;    -- avec un progiciel spécialisé  
    Visualiser_Les_Images_De_La_Base ;    -- avec un outil de visualisation  
    Droit.Fin_Acces_Total ;      -- libère l'accès à la base  
  end loop ;  
end Collecteur ;
```