Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Nom et prénom					
1	2	3			Total
NOTE					

DeSEm DeSEm

Examen DeSEm 2019-2020

20 janvier 2020

Durée : 120 minutes

1 (22)

DeSeNET

Questions concernant le protocole DeSeNET :

En laboratoire du cours DeSEm, vous avez implémenté le firmware d'un nœud "sensor" d'un réseau de capteurs qui sont tous reliés avec un master. Les sensors échantillonnent un nombre de valeurs de façon synchrone.

3 1. Expliquez avec une petit esquisse le mécanisme de synchronisation du protocole DeSeNET.

2 2. Quel est le nombre maximal de nœuds qui peuvent s'intégrer dans un tel réseau ? Justifiez votre réponse.

3 3. Quel élément du protocole limite la taille maximale d'une trame envoyée par un nœud sensor ? Justifiez votre réponse.

2 4. Quel élément physique limite la taille maximale d'une trame envoyée par un nœud sensor ? Justifiez votre réponse.

Questions concernant l'architecture DeSeNET :

3 5. Donnez un diagramme de séquence qui montre comment le `AccelerometerApplication` s'enregistre pour publier des valeurs échantillonnées auprès de `NetworkEntity`. On assume que tous les objets sont déjà créés et initialisés.

2 6. Pourquoi le `JoyStickApplication` ne doit pas s'enregistrer ?

5 7. Donnez un diagramme de séquence qui montre tout ce qui est exécuté quand un BEACON arrive (méthode `NetworkEntity::onReceive`).

2 8. Pourquoi c'est toujours la valeur échantillonnée à l'arrivée du BEACON précédent qui est envoyé à l'arrivé du BEACON actuel.

2 (22)

Ingénierie du logiciel embarqué, développement de protocole, patterns

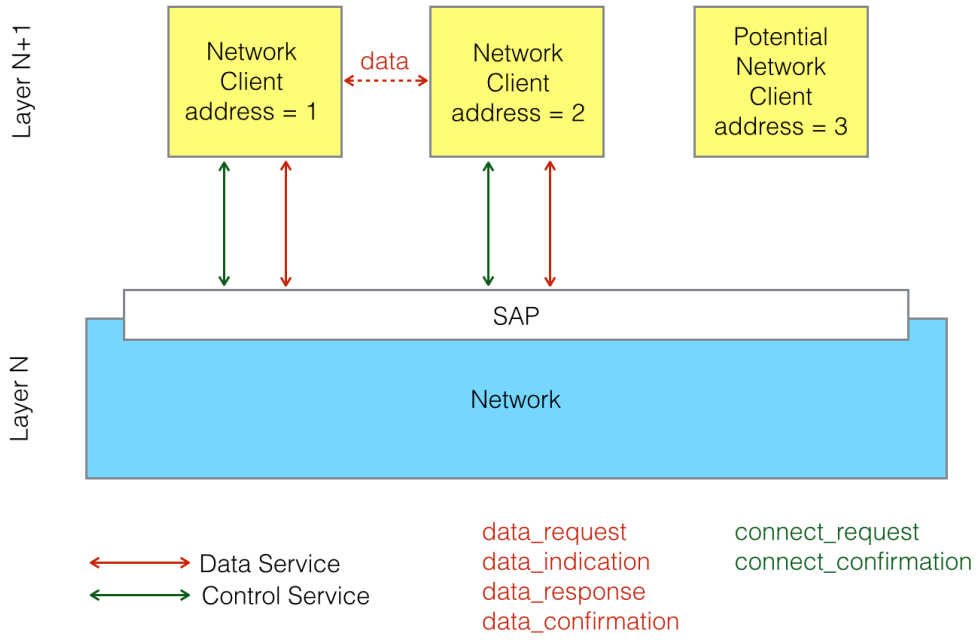


Figure 1: A simple communication infrastructure

La figure ci-dessus montre une installation de communication très simple.

La couche de réseau met à disposition un SAP simple. Il offre l'accès à plusieurs services :

Service de données (data) :

- data_request
- data_indication
- data_response
- data_confirmation

Service de connexion (data) :

- connect_request
- connect_confirmation

Les scénarios suivants démontrent comment fonctionne cette installation :

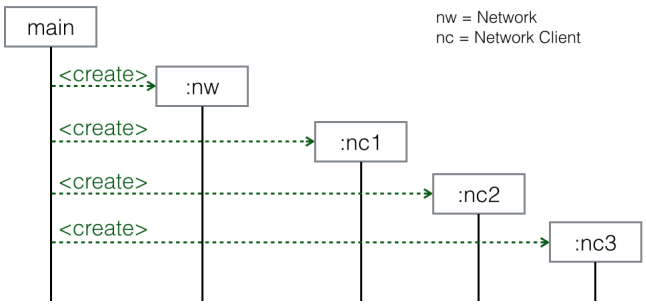


Figure 2: Initialization of the communication infrastructure

Le scénario ci-dessus montre comment le réseau et trois clients de réseau différents sont créés. Dans ce cas-ci, la fonction `main()` prend le rôle d'usine.

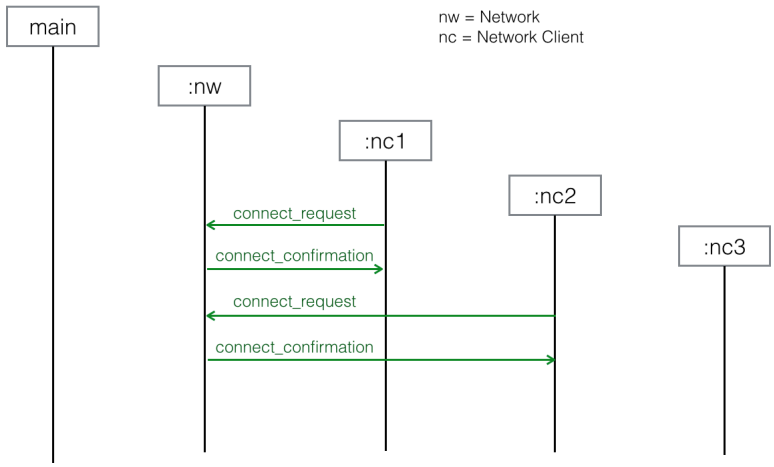


Figure 3: Connecting the network client 1 and 2 to the network

Le scénario précédent montre comment le client réseau 1 et le client réseau 2 se connectent au réseau. Ils utilisent pour ça le service *connect*. Comme on peut le voir, le réseau retourne une `connect_confirmation` comme réponse à un appel à un `connect_request`.

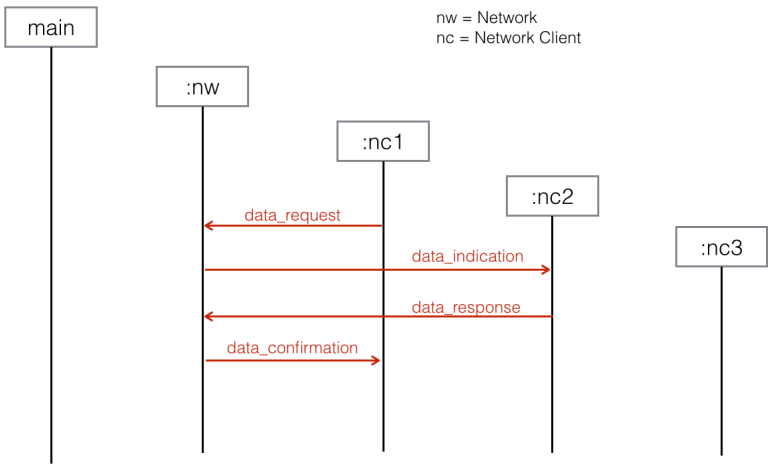


Figure 4: Data transmission from network client 1 to network client 2

Le scénario de la figure ci-dessus montre comment le client réseau numéro 1 envoie des données au client réseau numéro 2. Les deux utilisent pour ce faire les primitives du service *données*. Tout d'abord, le client numéro 1 utilise la primitive `data_request` pour demander au réseau de transmettre des données vers le client numéro 2. Il passe les données et l'adresse du client numéro deux comme donnée. Le réseau transporte les données et indique leurs arrivées au client numéro 2 en utilisant la primitive `data_indication`. Le client numéro 2 accepte les données et il quittance leur réception en envoyant une `data_response` avec l'adresse du client numéro 1 comme donnée. Le réseau transporte celle-ci et indique son arrivée au client numéro 1 avec une `data_confirmation`. Avec ceci, la transmission de données est complète.

On assume, que la couche `NW` fournit les classes `Network` et `Message`. La couche `App` fournit les classes `NWClient` et `Data`.

Questions :

6 1. Dessinez un diagramme de classes UML avec les classes `NWClient`, `Data`, `Network` et `Message`. Répartissez les classes dans les deux paquetages. Dessinez les paquetages et les classes avec leurs méthodes et relations. N'oubliez pas que les méthodes qui représentent des primitives de service portent des paramètres. Dessinez aussi les relations entre les classes.

6 2. Ajoutez à votre diagramme de classes un paquet avec une classe `Factory`. Elle prend la responsabilité de créer les objets d'un système comme par exemple montré en figure 3. Ajoutez aux classes concernées les méthodes qui font partie du pattern d'usinage.

6 3. L'on veut appliquer le pattern `SAP` du cours pour découpler les couches. Vous devez donc définir un nombre d'interfaces. Proposez ces interfaces. Donnez une liste avec chaque fois le nom de l'interface et ses méthodes. Proposez aussi un nom de paquetage pour les interfaces.

4 4. Redessinez un NOUVEAU diagramme UML de classes avec le pattern `SAP` intégré. Donnez les paquetages, les classes et les interfaces avec leurs noms et SANS leurs méthodes. Indiquez TOUTES les relations entre les classes et interfaces.

3 (10)

Ingénierie des protocoles

Des nœuds équipés de capteurs sont reliés par radio à un concentrateur connecté à un réseau filaire. Dans le contexte de cet exercice, on considère que seuls les nœuds capteurs sont producteurs de données de mesure. Ils ne reçoivent que des trames de contrôle.

Les nœuds capteurs ne sont pas synchronisés : ils peuvent transmettre des trames à tout moment.

Chaque nœud capteur a une adresse unique **`sensorId`** codée sur 16 bits. Le concentrateur n'a pas d'adresse.

On considère un modèle en quatre couches : Physical, MAC, Transport et Application (voir Figure).

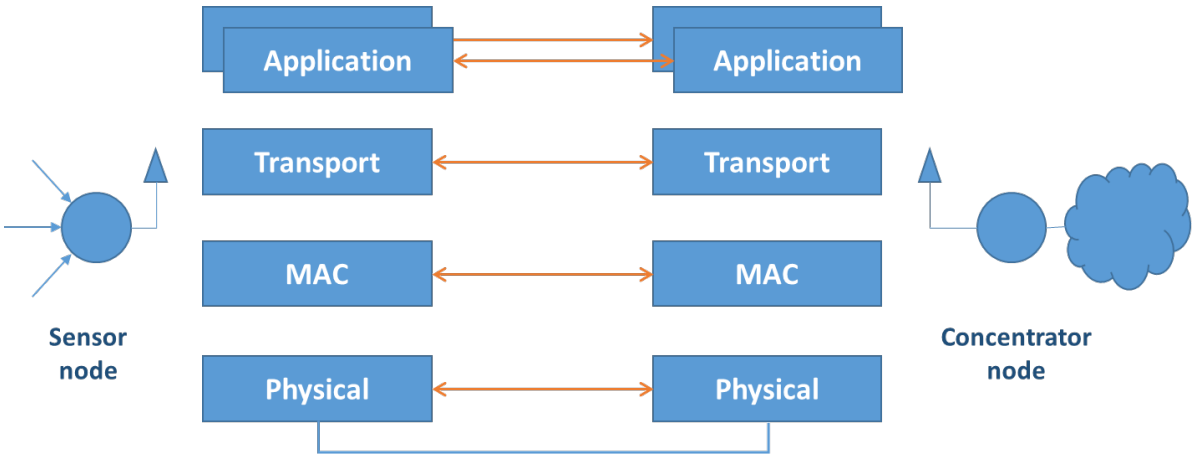


Figure 5 : Pile de communication

On dispose d'une entité implémentant un protocole Physical radio avec les services et primitives de services suivants :

```
PH_TRX.Request(ON | OFF)
// Turn on/off the radio transceiver
PH_DATA.Request(buffer)
PH_DATA.Indication(buffer)
PH_DATA.Confirm(status)
// Confirm that data in PH_DATA.Request has been sent
// (consider only "OK" status)
```

Lorsqu'une entité de la couche Application demande l'envoi de données, le protocole MAC applique la procédure suivante :

- Les données sont encapsulées dans une trame **`DATA`**.
- La trame **`DATA`** est transmise sans vérification.
- Une fois que la trame **`DATA`** a été transmise, la couche MAC attend la réception d'une trame pendant un temps **`AckWait`**.
- Si la trame reçue est une trame **`ACK`** (accusé de réception), la couche Transport est notifiée du succès de la transmission et le transceiver est déclenché.
- Dans les autres cas, on recommence l'étape 2 après un délai aléatoire **`RandWait`**.

La couche MAC dispose des primitives de service suivantes :

```
MAC_DATA.request(frameData)
MAC_DATA.confirm() // Confirm that an ACK from peer has been received
```

Deux timers sont définis :

Timer **RandWait**:**startTmRandWait**, **stopTmRanWait**	Timeout event: **evTmRandWait**
Timer **AckWait**: **startTmAckWait**, **stopTmAckWait**	Timeout event: **evTmAckWait**

La couche Transport attend **MAC_DATA.confirm()** avant un prochain **MAC_DATA.request()**.

La couche Transport permet à plusieurs entités de la couche Application de communiquer en parallèle. Concrètement elle permet à une entité Application :

- d'établir une connexion (service confirmé localement),
- de terminer une connexion (service non-confirmé), et
- de transporter des données (service non-confirmé).

Chaque entité Application dispose d'un identifiant **appID** (16 bits) unique au niveau d'un nœud capteur.

Questions :

Les questions se rapportent au nœud capteur (et non au concentrateur).

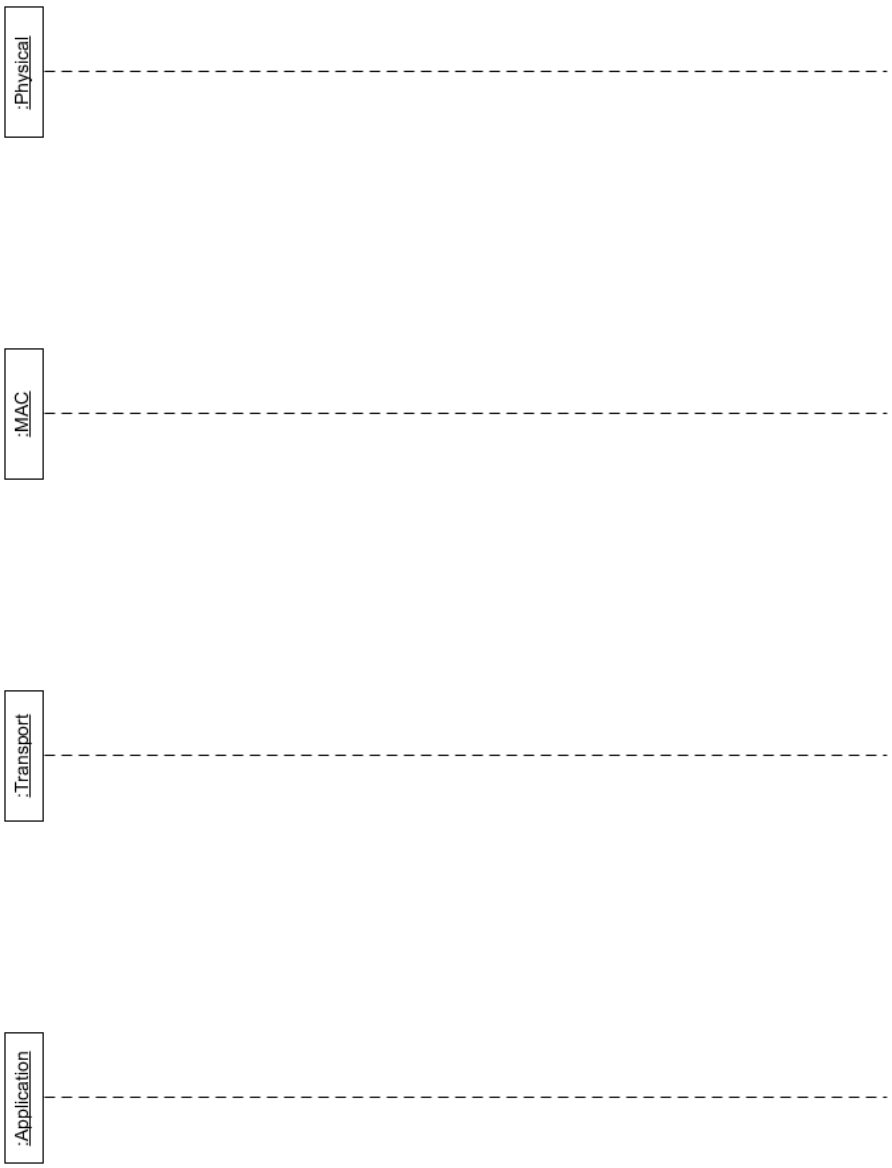
4 1. Dessinez la machine d'état du protocole de la couche MAC selon les spécifications ci-dessus.

1 2. Proposez un format pour les trames de la couche MAC.

1 3. Indiquez les services avec leurs primitives et leurs arguments pour la couche Transport.

1 4. Proposez un format pour les paquets de la couche Transport.

3 5. Représentez la phase de connexion d'une entité Application sur le diagramme de séquence annexé. Ne considérez que le scénario « sunny day ». Indiquez explicitement les trames émises et reçues.


:Physical
:MAC
:Transport
:Application