

# Bus CAN & protocoles

# Sommaire

- Chapitre 1 : **Introduction CAN / CANopen**
  - ▶ Définitions, historique, concept, normes & domaines d'application
- Chapitre 2 : **Caractéristiques d'un nœud CAN**
  - ▶ Physiques & Logiques
- Chapitre 3 : **Les réseaux CAN**
  - ▶ Contraintes & Topologies
- Chapitre 4 : **Les couches protocole**
  - ▶ CANopen, Devicenet, J1939

# Introduction CAN / CANopen

```
70 K_OS_Init(); /* initialize ram and things */
71 status = K_Task_Create(0,&clock_t
72 status = K_Task_Name(clock_t_slot,
73
74 status = K_Task_Create(0,&led_t_slot,1
75 status = K_Task_Name(led_t_slot,"Led");
76 status = K_Task_Create(3,&io_t_slot,10_t
77 status = K_Task_Name(io_t_slot,"Led 10");
78
79 status = K_Task_Create(10,&bgnd_t_slot,10_t
80 status = K_Task_Name(bgnd_t_slot,"Bgnd");
81
82 status = K_Timer_Create(TIMER_CLOCK,2,clock_t_slot,EVENT_TIM_CK);
83 status = K_Timer_Create(TIMER_LED,1,led_t_slot,EVENT_TIM_LED);
84 status = K_Semaphore_Create(SEM_NUM,0);
85
86 K_Task_Start(bgnd_t_slot); /* trigger task 4...
```

- Temps réel  
- Service -  
- Logiciel -  
- Programmation  
- Informatique Industrielle -  
- Assistance technique -  
- Informatique Industrielle

- Terrain -  
- Industrielle  
- Assistance  
- Informatique  
- Qualité logi  
- Informatique Industrielle

## Définitions

- Qu'est ce qu'un bus (de communication):
  - ▶ lignes (physiques) de communication entre plusieurs équipements électroniques
- Que signifie CAN :
  - ▶ Controller Area Network : Réseau de contrôleurs électroniques
  - ▶ Protocole spécifique de communication
- bus CAN :
  - ▶ Terme générique désignant à la fois les médias physiques et le protocole

## Définitions

- Nœud CAN :
  - Equipement électronique capable de dialoguer sur un bus CAN
- Réseau CAN :
  - En simplifiant : Réseau d'équipements électroniques (nœuds) interconnectés utilisant le protocole CAN pour échanger des informations.
- CANopen / Devicenet / J1939 :
  - Protocoles de « haut-niveau » fonctionnant sur le bus CAN

## Historique

- Début années 80

- ▶ Intérêt général des constructeurs automobiles haut de gamme pour les Systèmes de communication temps réel (électronique embarquée).
- ▶ Faute de solutions dédiées, utilisation du bus I2C (Inter Integrated Circuits) développé par Philips.

- 1983

- ▶ La société Robert Bosch GmbH ([www.bosch.de](http://www.bosch.de)) en partenariat avec l'université de Karlsruhe choisit de développer un protocole/bus à haut niveau de sécurité orienté systèmes distribués temps réel :

### **Le bus CAN**

## Historique

- 1985
  - Bosch signe un partenariat avec Intel (pour les USA) puis avec Philips et Siemens (pour l'Europe) pour les composants.
- 1986
  - La SAE officialise le protocole et les premiers composants apparaissent l'année suivante.
- 1991
  - BOSCH publie les spécifications de la version 2.0
- Depuis...

# Introduction

## Historique

- 1985
  - Bosch signe un partenariat avec Intel (pour les USA) puis avec Philips et Siemens (pour l'Europe) pour les composants.
- 1986
  - La SAE officialise le protocole et les premiers composants apparaissent l'année suivante.
- 1991
  - BOSCH publie les spécifications de la version 2.0
- Depuis...



# Introduction

## Concept

- Bus faible coût
  - ▶ Composants et mise en œuvre « bon marché »
- Bus robuste
  - ▶ Médias robustes et détection d'erreur performante
- Bus flexible :
  - ▶ Multi-maîtres : Pas de relation spécifiques entre les nœuds
- Bus performant
  - ▶ Transfert « temps-réel » des informations

## Concept

- Transmission fiable dans les environnements perturbés
  - Longueur des données réduite :
    - Taille suffisante pour la plupart des applications
    - Contrainte : Segmentation nécessaire si taille informations > 8 octets
- Émission des messages sur évènement :
  - Charge du bus réduite
  - Temps de latence court pour les données temps réel
- Émission des messages en fonction du niveau de priorité
  - Réduction des temps de latence pour les messages important

## Concept

- Topologie du réseau
  - ▶ Le format « bus » est préconisé
  - ▶ Possibilité d'étoile ou de « backbone »
  - ▶ Longueur du bus limitée par le comportement temps réel
- Codage Numérique de l'information
  - ▶ Signal série codé en NRZ : Simplicité de codage
  - ▶ Deux niveaux « 0 » et « 1 » (resp. dominant et récessif)
  - ▶ Fonction de « ET » sur le bus

## Concept

- Accès au médium
  - ▶ Asynchrone
  - ▶ Sans collision
  
- Détection d'erreurs
  - ▶ Par l'émetteur et les récepteurs
  - ▶ Divers mécanismes dont CRC
  - ▶ Intégrité des données véhiculées

## Concept

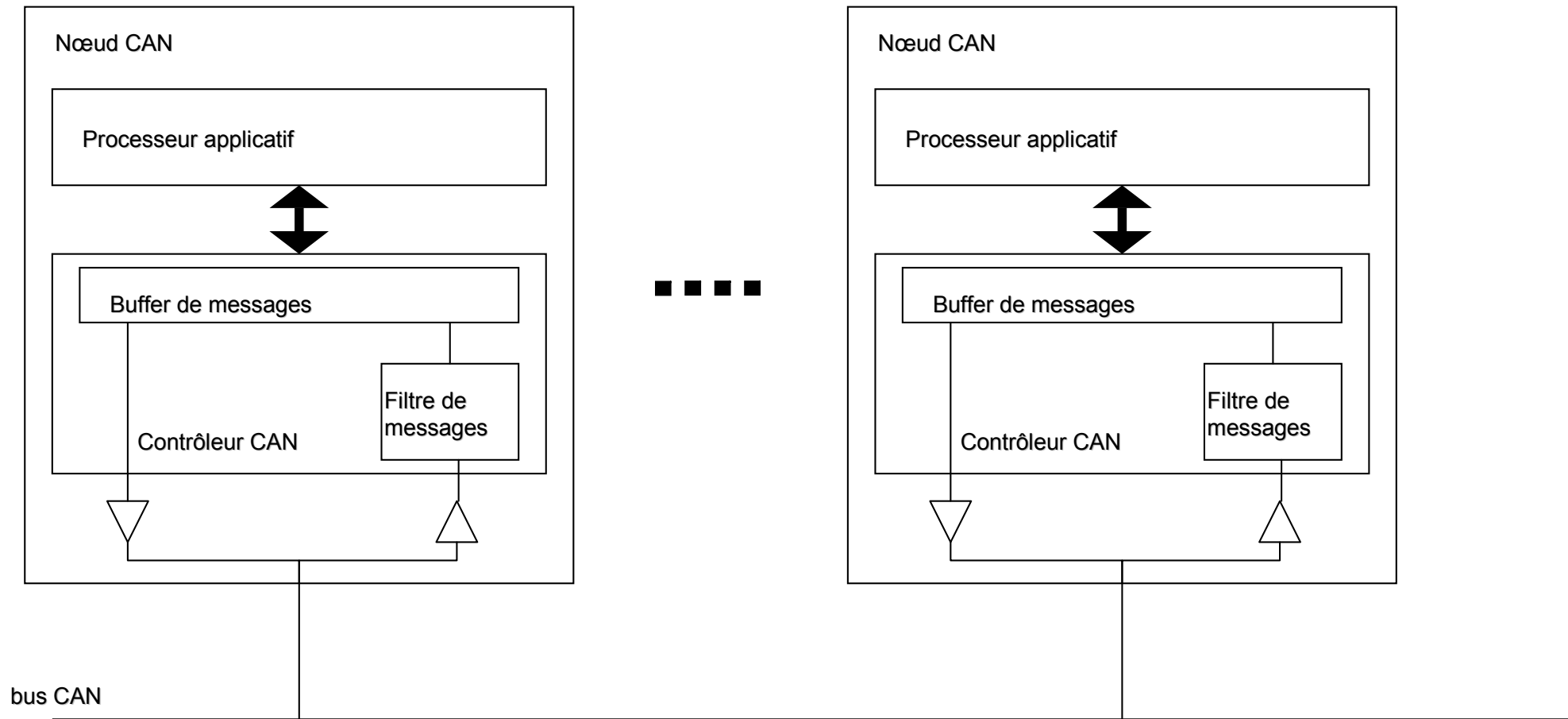
- Signalement d'erreur
  - Détection en temps réel
  - Temps de recouvrement d'erreur très court :
    - Réduction de la charge du bus
  - Mécanisme de confinement
    - Mise hors service automatique et autonome d'un nœud défectueux

## Concept

- Notions de fonctionnement de base
  - ▶ Tous les participants peuvent démarrer la communication dès que le bus est au repos (*IDLE*) :
    - Pas de mécanisme de gestion des collisions nécessaire grâce au principe d'arbitrage non destructif
  - ▶ Un seul émetteur une fois la phase d'arbitrage terminée
  - ▶ Tout nœud sur le bus qui n'est pas émetteur est récepteur
  - ▶ Tout nœud sur le bus est en charge de vérifier l'intégrité du message

# Introduction

## Représentation d'un réseau CAN



# Introduction

## Normes

- Norme actuelle :
  - CAN 2.0B
- Plusieurs standards ISO pour le CAN :
  - ISO 11898-1: CAN Data Link Layer and Physical Signalling
  - ISO 11898-2: CAN High-Speed Medium Access Unit
  - ISO 11898-3: CAN Low-Speed, Fault-Tolerant, Medium-Dependent Interface
  - ISO 11898-4: CAN Time-Triggered Communication
  - ISO 11898-5: CAN High-Speed Medium Access Unit with Low-Power Mode
  - ISO 11992-1: CAN fault-tolerant for truck/trailer communication



## Normes

- Principaux protocoles :
  - CANopen
  - DeviceNet
  - J1939
  - ISOBUS
  - NMEA2000
  - ARINC-825
- Organismes de gestion reconnus en europe :
  - CiA (CAN in Automation)

# Introduction

## Domaines d'application

- Industrial automation
- Home/Building automation
- Automotive (VL, PL)
- Transportation (Train, aviation, ...)
- Matériel agricole, Travaux public
- Maritime
- Medical
- Instrumentation

# Sommaire

- Chapitre 1 : **Introduction CAN / CANopen**
  - ▶ Définitions, historique, concept, normes & domaines d'application
- Chapitre 2 : **Caractéristiques d'un nœud CAN**
  - ▶ Physiques & Logiques
- Chapitre 3 : **Les réseaux CAN**
  - ▶ Contraintes & Topologies
- Chapitre 4 : **Les couches protocole**
  - ▶ CANopen, Devicenet, J1939

# Caractéristiques d'un nœud CAN

```
70 K_OS_Init(); /* initialize ram and things */
71 status = K_Task_Create(0,&clock_task,CLOCK_TASK_NAME,1024); /* create task 1 */
72 status = K_Task_Name(clock_task,CLOCK_TASK_NAME); /* Name for task */
73
74 status = K_Task_Create(0,&led_task,LED_TASK_NAME,1024); /* create task 2 */
75 status = K_Task_Name(led_task,LED_TASK_NAME); /* Name for task */
76
77 status = K_Task_Create(3,&io_task,IO_TASK_NAME,1024); /* create task 3 */
78 status = K_Task_Name(io_task,IO_TASK_NAME); /* Name for task */
79
80 status = K_Task_Create(3,&t5ms_task,T5MS_TASK_NAME,1024); /* create task 4 */
81 status = K_Task_Name(t5ms_task,T5MS_TASK_NAME); /* Name for task */
82
83 status = K_Timer_Create(TIMER_CLOCK,2,clock_task,slot_EVENT_TIMER_CLOCK);
84 status = K_Timer_Create(TIMER_LED,2,led_task,slot_EVENT_TIMER_LED);
85 status = K_Semaphore_Create(SEM_NUM,0);
86
87 K_Task_Start(bgnd_task); /* trigger task 1
```

- Temps réel

- Service

- Qualité logicielle

- Programmation

- Informatique Industrielle

- Service

le terrain -

industrielle

tion - Ass

- assistance technique - Informati

- Temps réel embarqué - Qualité logi

- assistance technique - Informatique Industr

# Caractéristiques physiques

## Contraintes principales

- Véhiculer deux niveaux logiques :
  - Dominants et Récessifs
- Etre au niveau récessif au repos
- Avoir une fonction de « ET » logique entre les différents nœuds :
  - Si un nœud impose un niveau dominant, alors le bus est à l'état dominant

# Caractéristiques physiques

## ISO 11898-3 - CAN High Speed

- Média « standard » :
  - ▶ Double paire torsadée blindée
  - ▶ Signaux :
    - CAN High
    - CAN Low
    - CAN GND
    - CAN Vcc (optionnel)
  - ▶ Possibilité d'alimentation des composants réseau par le CAN (Transceivers)

# Caractéristiques physiques

## ISO 11898-3 - CAN High Speed

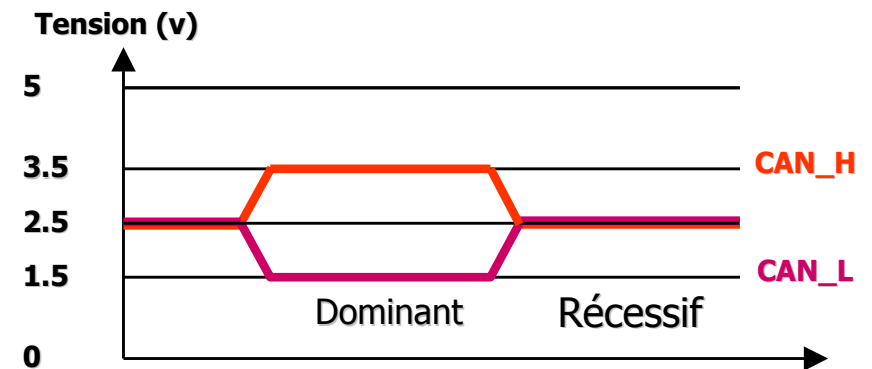
- Caractéristiques du signal :

- ▶ Tension différentielle avec retour commun

- Niveau Récessif :  $V_{CAN\_H} - V_{CAN\_L} = 0V$  (tolérance : de  $-500mV$  à  $+50mV$ )
- Niveau Dominant :  $V_{CAN\_H} - V_{CAN\_L} = 2V$  (tolérance : de  $+1,5V$  à  $+3V$ )

- ▶ Seuils de détection :

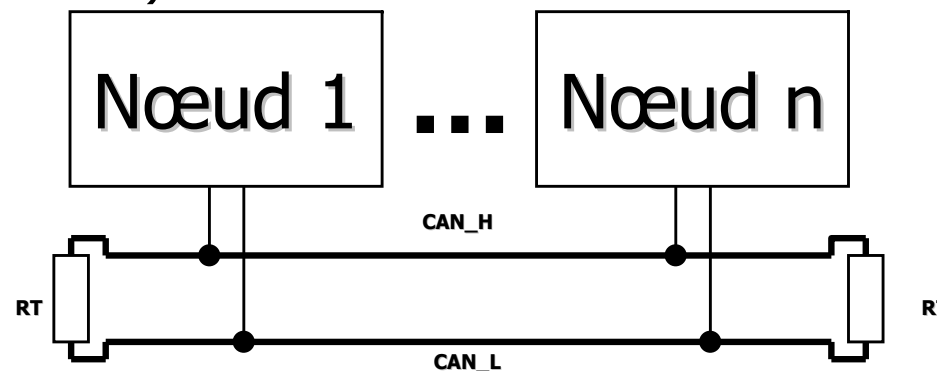
- Niveau Récessif :  
si  $V_{CAN\_H}$  inférieur ou égale à  $V_{CAN\_L} + 0,5V$
- Niveau Dominant :  
si  $V_{CAN\_H}$  est au moins supérieur à  $V_{CAN\_L} + 0,9V$



# Caractéristiques physiques

## ISO 11898-3 - CAN High Speed

- Caractéristiques du médium :
  - ▶ Câble
    - Vitesse de propagation nominale :  $5 \text{ ns.m}^{-1}$
    - Impédance nominale :  $70 \text{ mOhm.m}^{-1}$
    - Section : dépendante de la distance et du nombre de nœuds :
      - de  $0,1 \text{ mm}^2$  pour 40m à  $0,4 \text{ mm}^2$  pour une centaine de nœuds à 450m.
  - ▶ Adaptation de ligne nécessaire :
    - RT (Résistance de Terminaison) : 120 Ohm





# Caractéristiques physiques

## ISO 11898-3 - CAN High Speed

- Relation vitesse /distance :
  - ▶ Distance inversement proportionnelle à la vitesse : + la vitesse augmente, + la longueur max. du réseau diminue :
    - Causes :
      - le comportement temps-réel des phases d'arbitrage
      - le comportement temps-réel des phases de détection d'erreur,
  - ▶ Vitesse de communication :
    - CAN 2.0B : jusqu'à  $1\text{Mb.s}^{-1}$  (jusqu'à  $125\text{ kb.s}^{-1}$  uniquement en CAN 2.0A)
  - ▶ Distances théoriques :
    - $> 1\text{km}$  pour  $10\text{kb.s}^{-1}$
    - $< 40\text{m}$  pour  $1\text{Mb.s}^{-1}$

# Caractéristiques physiques

## Connectique

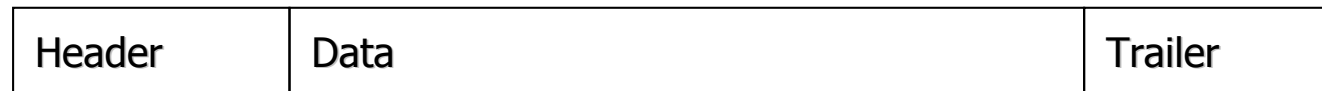
- Connectiques industrielles standardisées :
  - ▶ DB-9
  - ▶ Borniers industriels
  - ▶ M-12
  - ▶ Connecteur HE10

	DB-9	Bornier	M-12	HE10
1	<i>Reserved</i>	GND	Shield (opt.)	<i>Reserved</i>
2	CAN-LOW	CAN-LOW	CAN V+	GND (opt.)
3	GND	Shield (opt.)	GND	CAN-LOW
4	<i>Reserved</i>	CAN-HIGH	CAN-HIGH	CAN-HIGH
5	Shield (opt.)	CAN V+	CAN-LOW	GND
6	GND (opt.)			CAN V+
7	CAN-HIGH			<i>Reserved</i>
8	<i>Reserved</i>			<i>Reserved</i>
9	CAN V+			<i>Reserved</i>
10				<i>Reserved</i>

# Caractéristiques logiques

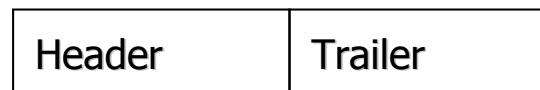
## Les types de messages

- Trames de données (représentation simplifiée) :



- ▶ Header (simplifié) : Identificateur + bits de contrôle (dont RTR)
- ▶ Data : données (8 octets max.)
- ▶ Trailer : CRC + ACK + fin de trame

- Trames de requête :



- ▶ Header (simplifié) : Identificateur + bits de contrôle (dont RTR)
- ▶ Trailer : CRC + ACK + fin de trame

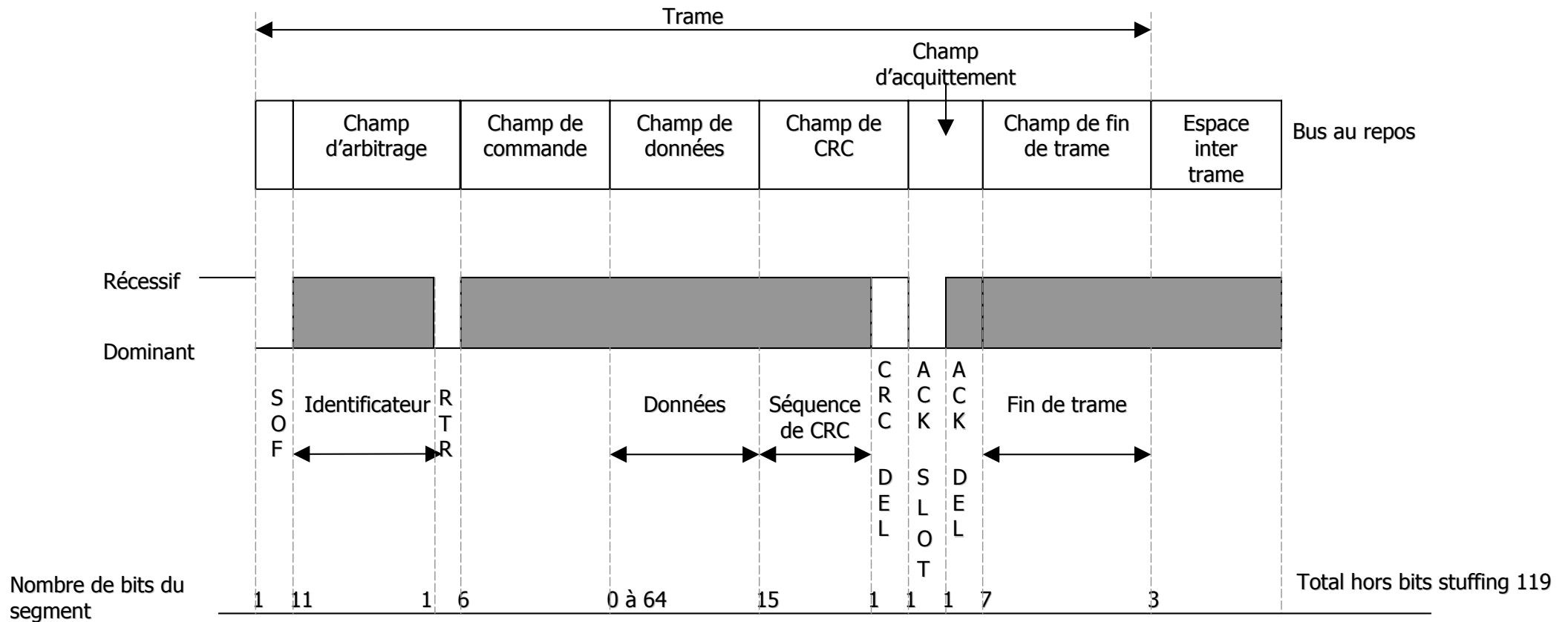
# Caractéristiques logiques

## Les types de messages

- Caractéristiques :
  - ▶ Identificateur : 11 ou 29 bits
  - ▶ Bits de contrôle : taille de données émises ou requises + type ID
  - ▶ Fin de trame : 7 bits récessifs
- Identificateur unique sur le réseau pour les trames de données
  - ▶ Chaque nœud doit avoir un jeu d'identificateurs unique

# Caractéristiques logiques

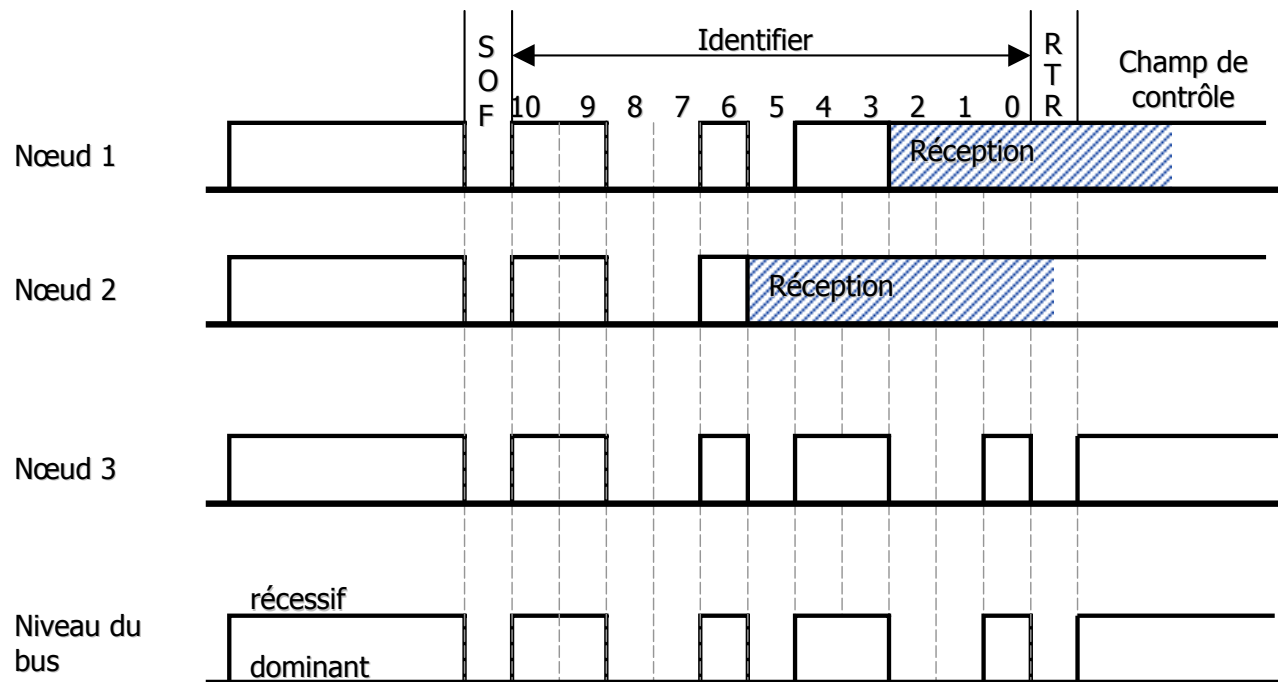
## Représentation trame de données (version 11 bits)



# Caractéristiques logiques

## Principe d'arbitrage à l'émission

- Chaque nœud émet son champ d'arbitrage et contrôle le niveau sur le bus :
  - ▶ Si le niveau observé est celui émis, il continue l'émission
  - ▶ Si le niveau observé n'est pas celui émis, il stoppe l'émission et devient récepteur





# Caractéristiques logiques

## Principe d'encodage des messages

- Le message est représenté au format binaire en NRZ :
  - ▶ Problème du NRZ asynchrone :  
Sur de longs messages, possibilité de désynchronisation des horloges de chaque nœud entraînant des corruptions de message
  - ▶ Solution du protocole CAN : Le bit Stuffing
    - Un bit de niveau inverse est rajouté à l'émission tous les 5 bits consécutifs de même niveau
    - Supprimé du message final par le récepteur
    - Intérêt : Génère un front de resynchronisation au niveau physique



# Caractéristiques logiques

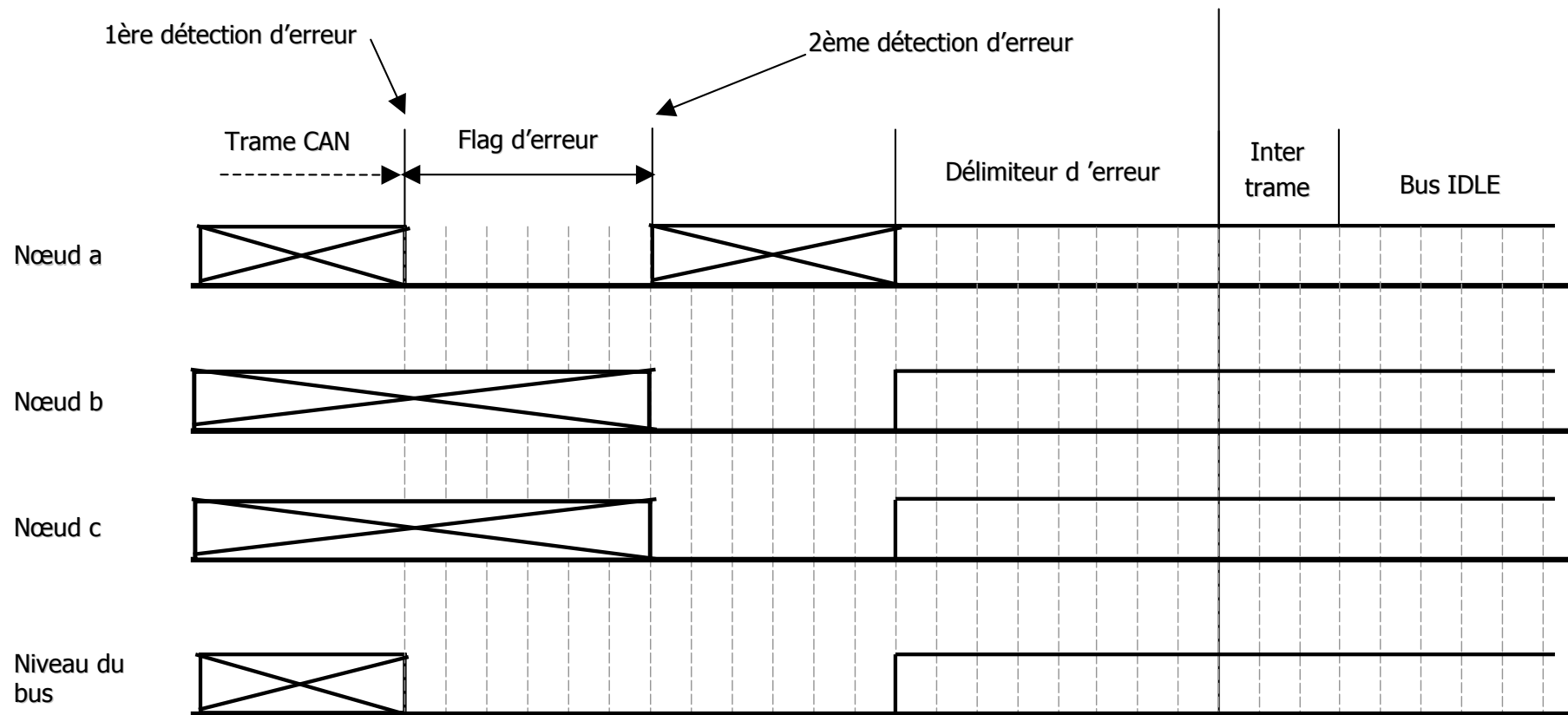
## Gestion des erreurs

- Erreurs détectées :
  - ▶ Erreur Bit
  - ▶ Erreur de CRC
  - ▶ Erreur de Format
  - ▶ Erreur de Stuff
  - ▶ Erreur Acknowledge
- Mécanisme de confinement automatique :
  - ▶ Entretien local de compteurs erreurs Tx et Rx
  - ▶ Si compteur Tx ou Rx  $> 128$ , mode Erreur passive
  - ▶ Si compteur Tx  $> 255$ , confinement (BUS OFF)

# Caractéristiques logiques

## Principe de signalisation des erreurs

- Tout nœud qui détecte une erreur doit la signaler :



# Sommaire

- Chapitre 1 : **Introduction CAN / CANopen**
  - ▶ Définitions, historique, concept, normes & domaines d'application
- Chapitre 2 : **Caractéristiques d'un nœud CAN**
  - ▶ Physiques & Logiques
- Chapitre 3 : **Les réseaux CAN**
  - ▶ Contraintes & Topologies
- Chapitre 4 : **Les couches protocole**
  - ▶ CANopen, Devicenet, J1939

# Les réseaux CAN

```
70 K_OS_Init(); /* initialize ram and things */  
71 status = K_Task_Create(0, &clock_task, slot, Clock_task, 1024); /* create task 1 */  
72 status = K_Task_Name(clock_t_slot, "Clock"); /* Name for task */  
73  
74 status = K_Task_Create(0, &led_t_slot, slot, Led_t_slot, 1024); /* create task 2 */  
75 status = K_Task_Name(led_t_slot, "Led"); /* Name for task */  
76  
77 status = K_Task_Create(3, &io_t_slot, slot, IO_t_slot, 1024); /* create task 3 */  
78 status = K_Task_Name(io_t_slot, "IO"); /* Name for task */  
79  
80 status = K_Task_Create(3, &t5ms_t_slot, slot, t5ms_t_slot, 1024); /* create task 4 */  
81 status = K_Task_Name(t5ms_t_slot, "t5ms"); /* Name for task */  
82  
83 status = K_Task_Create(10, &bgnd_t_slot, slot, bgnd_t_slot, 1024); /* create task 5 */  
84 status = K_Task_Name(bgnd_t_slot, "Bgnd"); /* Name for task */  
85  
86 status = K_Timer_Create(TIMER_CLOCK, 2, clock_t_slot, EVENT_TIMER_CLOCK, 1);  
87 status = K_Timer_Create(TIMER_LED, 1, led_t_slot, EVENT_TIMER_LED, 1);  
88 status = K_Semaphore_Create(SEM_NUM, 0);  
89  
90 K_Task_Start(bgnd_t_slot); /* trigger task 5 */
```

- Temps réel

- Service

- Logiciel

- Programmation

- Industrielle

- Service

- Assistance technique

- Informatique Industrielle

le terrain -

industrielle

tion - Ass

- Informati

Qualité logi

- Informatique Industr

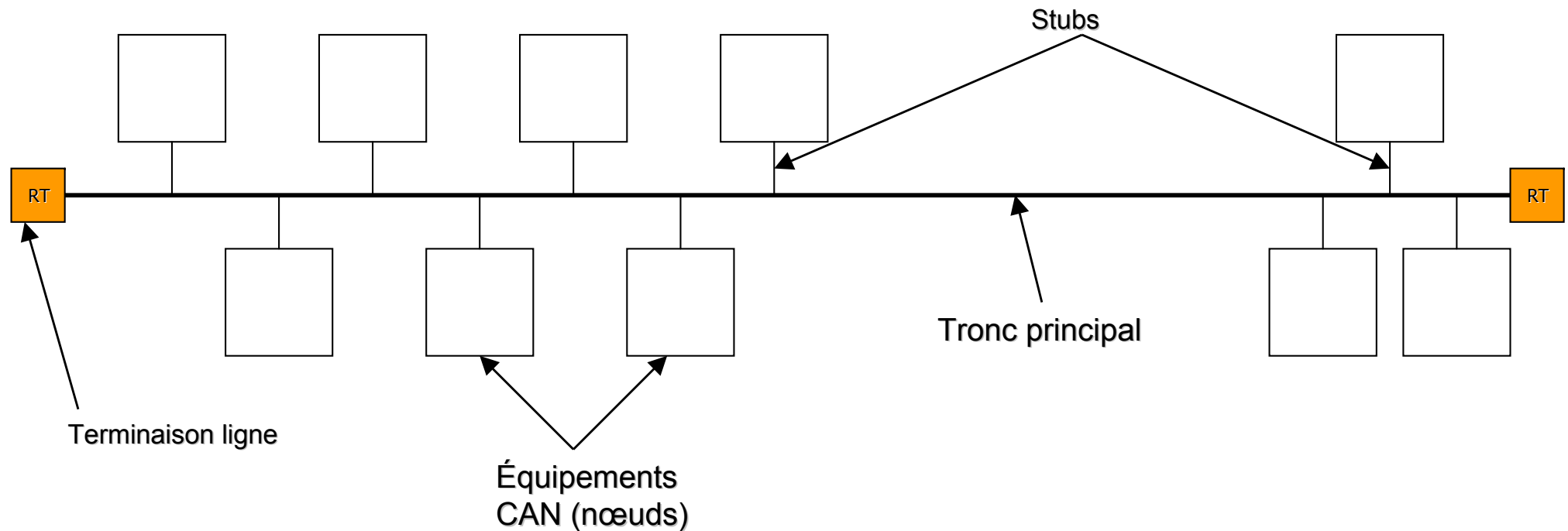
# Les réseaux CAN

## Contraintes de mise en œuvre

- Tous les nœuds d'un même réseau doivent communiquer à la même vitesse
- La vitesse impacte sur la longueur maximum du réseau
- La charge du réseau doit être évaluée afin de déterminer la vitesse nécessaire
- Les topologies autres que le bus (arbre, étoile) apportent des contraintes supplémentaires :
  - Perturbations du signal original par réflexions sur les branches secondaires

# Les réseaux CAN

## Topologie Bus standard



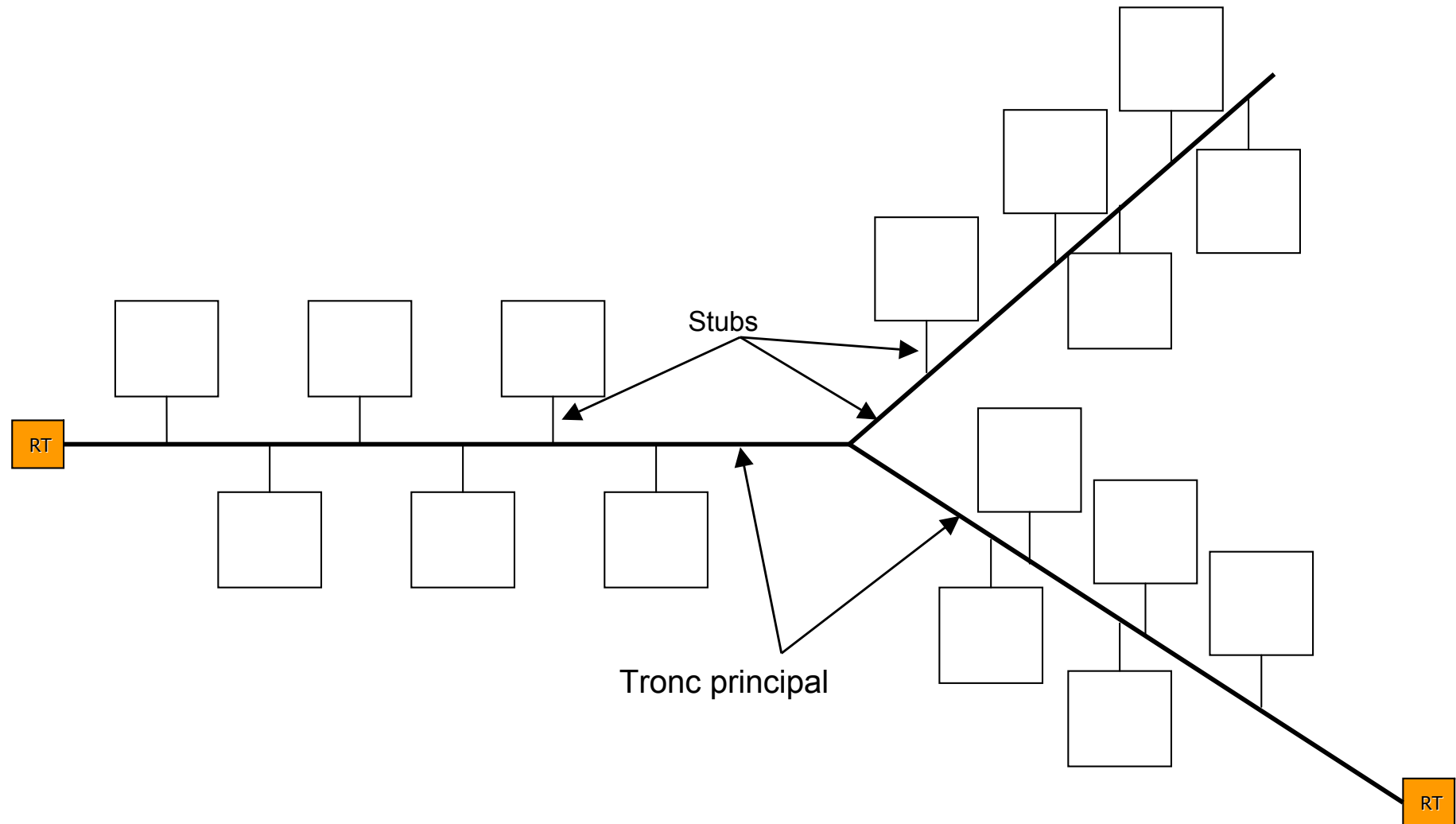
# Les réseaux CAN

## Topologie Bus standard

- Longueur de bus aisée à calculer :
  - ▶ Stubs : longueurs négligeables (< 30 cm)
  - ▶ Longueur = longueur du tronc principal (env.)
- Terminaisons faciles à positionner :
  - ▶ Identification facile des deux extrémités du câble
- Remarque :
  - ▶ Pas forcément adapté aux lignes de production avec câblage complexe : le cheminement du câble va probablement rajouter de la longueur « inutile »

# Les réseaux CAN

## Topologie Bus étoile





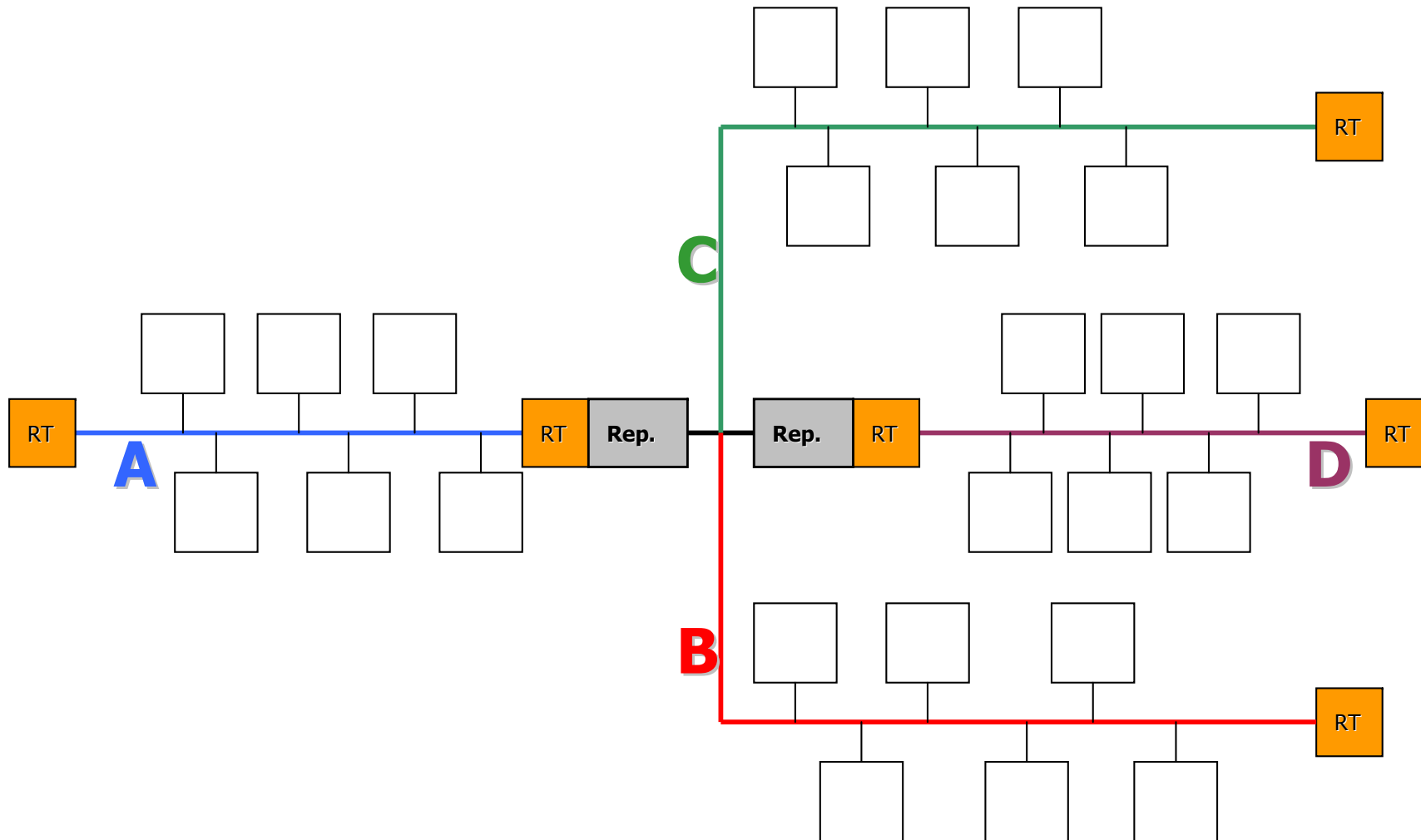
# Les réseaux CAN

## Topologie Bus étoile

- Longueur de bus plus compliquée à calculer :
  - Longueur = longueur des différents troncs
  - Nécessité de comptabiliser une partie en stub qui vont générer des perturbations sur le tronc principal
- Terminaisons plus difficiles à positionner
  - En fonction des branches les plus longues
- Remarque :
  - Plus adapté que le bus aux lignes de production avec câblage complexe
  - Problème contraignant des stubs et longueurs associés :
    - Nécessite parfois l'utilisation d'interfaces d'adaptation électrique spécifiques qui « simplifient » la topologie

# Les réseaux CAN

## Topologie Bus étoile optimisé (1)



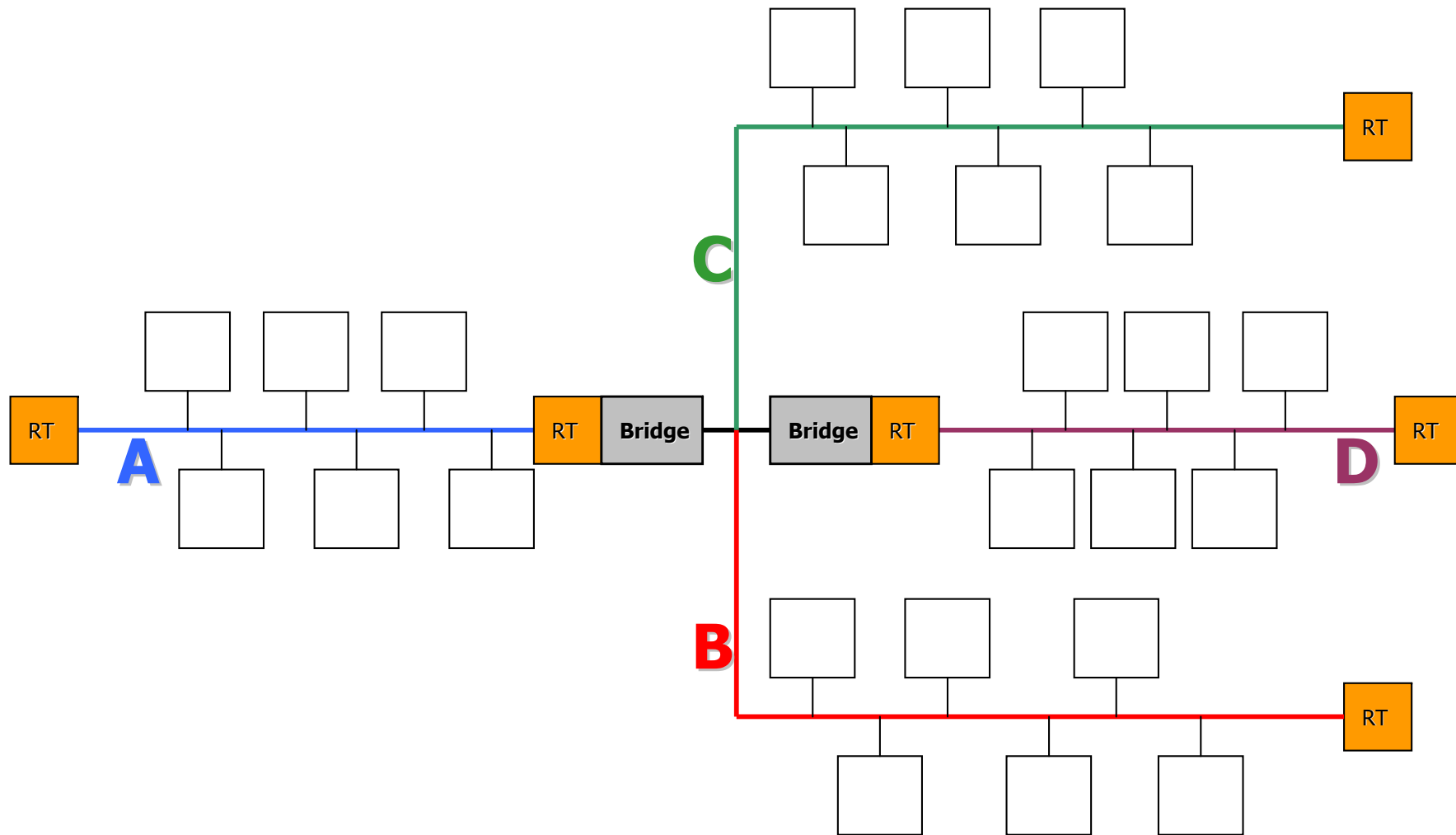
# Les réseaux CAN

## Topologie Bus étoile optimisé (1)

- Utilisation de « répéteur » CAN :
  - Différenciation de segments => création de plusieurs troncs principaux
  - Calcul simplifié pour la longueur max. :
    - A-B, A-C, A-D, B-C, B-D, C-D
    - Penser à rajouter la « longueur équivalente ligne » d'un répéteur
- Terminaisons aisées à positionner
- Remarque :
  - **Même domaine temps réel**
  - Meilleur compromis pour les lignes de production avec câblage complexe
  - Moins de contraintes au niveau stub

# Les réseaux CAN

## Topologie Bus étoile optimisé (2)



# Les réseaux CAN

## Topologie Bus étoile optimisé (2)

- Utilisation de « Bridge » CAN :
  - Différenciation de réseaux => création plusieurs réseaux CAN distincts
  - Calcul simplifié pour la longueur max :
    - Chaque réseau a ses propres contraintes de distance
- Terminaisons aisées à positionner
- Remarque :
  - **Réseaux CAN différents : disparition du temps réel**
  - Possibilité de filtrage des messages (réduction de charge sur certains tronçons)
  - Possibilité d'utilisation de vitesses différentes selon réseaux

# Sommaire

- Chapitre 1 : **Introduction CAN / CANopen**
  - ▶ Définitions, historique, concept, normes & domaines d'application
- Chapitre 2 : **Caractéristiques d'un nœud CAN**
  - ▶ Physiques & Logiques
- Chapitre 3 : **Les réseaux CAN**
  - ▶ Contraintes & Topologies
- Chapitre 4 : **Les couches protocole**
  - ▶ CANopen, Devicenet, J1939

# Les couches protocole

```
70 K_OS_Init(); /* initialize ram and things */
71 status = K_Task_Create(0,&clock_task,1024); /* create task 1 */
72 status = K_Task_Name(clock_task,"clock"); /* Name for task */
73
74 status = K_Task_Create(0,&led_task,1024); /* create task 2 */
75 status = K_Task_Name(led_task,"Led"); /* Name for task */
76
77 status = K_Task_Create(3,&io_task,1024); /* create task 3 */
78 status = K_Task_Name(io_task,"IO"); /* Name for task */
79
80 status = K_Task_Create(3,&timer_task,1024); /* create task 4 */
81 status = K_Task_Name(timer_task,"Timer"); /* Name for task */
82
83 status = K_Task_Create(3,&bgnd_task,1024); /* create task 5 */
84 status = K_Task_Name(bgnd_task,"Bgnd"); /* Name for task */
85
86 K_Timer_Create(TIMER_CLOCK,2,clock_task,slot_EVENT_TIMER_CLOCK);
87 K_Timer_Create(TIMER_LED,1,led_task,slot_EVENT_TIMER_LED);
88 K_Semaphore_Create(SEM_NUM,0);
89
90 K_Task_Start(bgnd_task); /* trigger task 5
```

- Temps réel

- Service

- Qualité logicielle

- Programmation - Service

- Informatique Industrielle - Temps réel embarqué - Qualité logicielle

- Assistance technique - Informatique Industrielle

# Couches protocole

## Les principales couches protocole

- CANopen
- Devicenet
- J1939



# Protocole CANopen

# Protocole CANopen : *Introduction*

## Historique

- 1993 : Un programme de recherche européen (ASPIC) permet de développer une spécification de couche applicative : CAL
- 1994-95 : Création du CiA Interest Group de CANopen définition de documents concernant CANopen Communication Profile et I/O Device profile. Parution du CiA-301 v 2.0
- 1996 : Parution du CiA-301 v3.0
- 1996-99 : Création de plusieurs SIG pour la spécification du Communication profile, du Device Profile et des standards d'applications
- 1999 : Parution du CiA-301 v4.0
- Octobre 2006 : Parution du CiA-301 v4.1

*Depuis 1996... De nouveaux SIG apparaissent pour définir de nouveaux Device Profiles et des fonctionnalités de communication additionnelles.*

# Protocole CANopen : *Introduction*

## Principal objectif

Fournir une solution standardisée pour des systèmes d'automatisation industrielle distribués.

- Architecture système normalisée
- Interopérabilité des matériels de fabricants différents sur un même réseau :
  - ▶ Pas de dépendance vis-à-vis des fabricants
  - ▶ Utilisation des matériels les mieux adaptés au système/contraintes
- Apprentissage technique pérenne
- Utilisation d'outils, de logiciels de communication et de couches protocoles sur étagère

# Protocole CANopen : *Introduction*

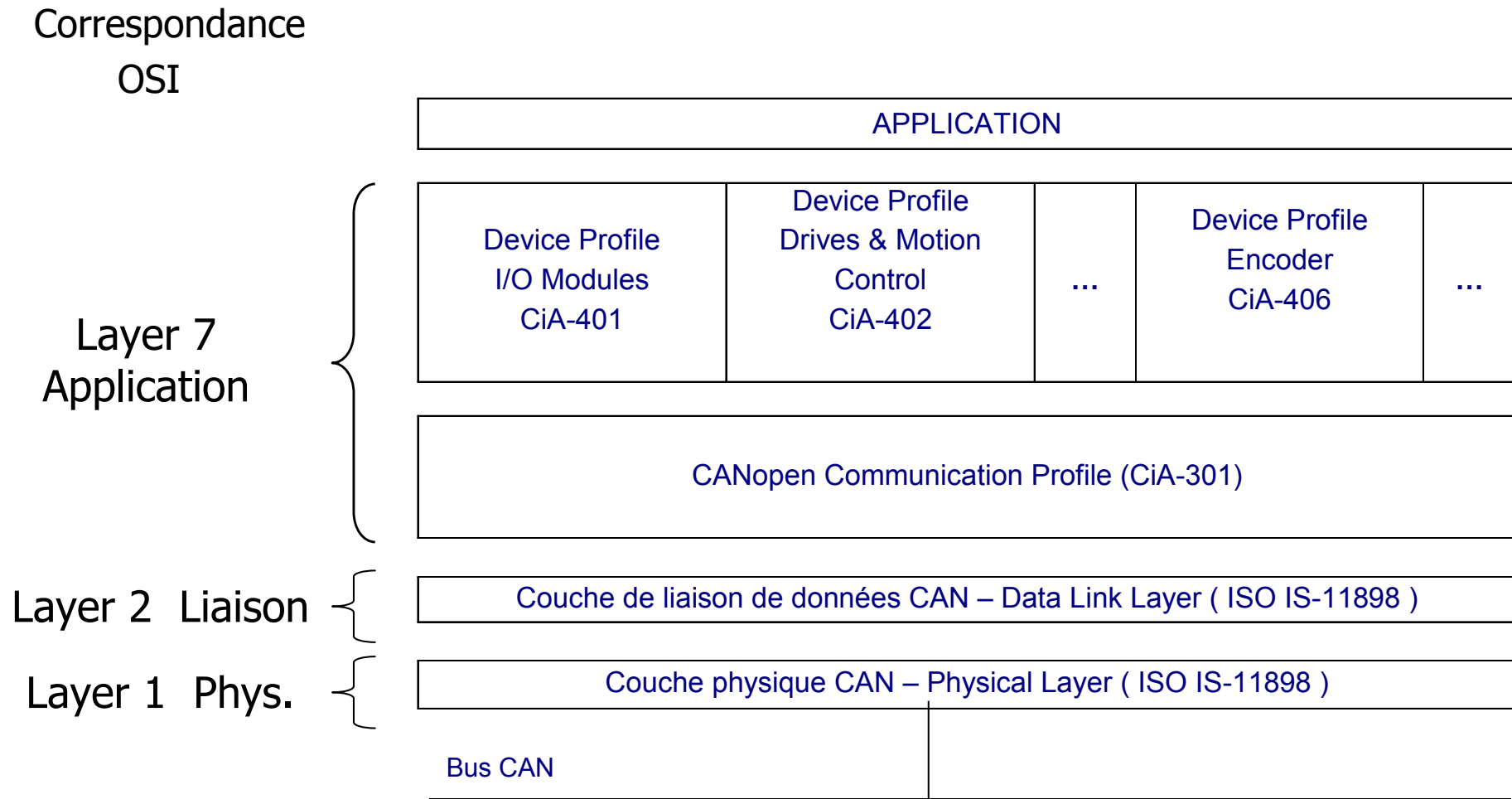
## Les éléments principaux

Standardisation maximum à l'aide de :

- ▶ Gestion réseau normalisée
- ▶ Messagerie réseau normalisée
  - Messages de service (SDO)
  - Messages de process (PDO)
- ▶ Périphériques (matériels) normalisés :
  - Fonctionnalités
  - Informations

# Protocole CANopen : *Introduction*

## Modèles de référence d'un matériel CANopen



# Protocole CANopen : *Introduction*

## Principaux documents disponibles

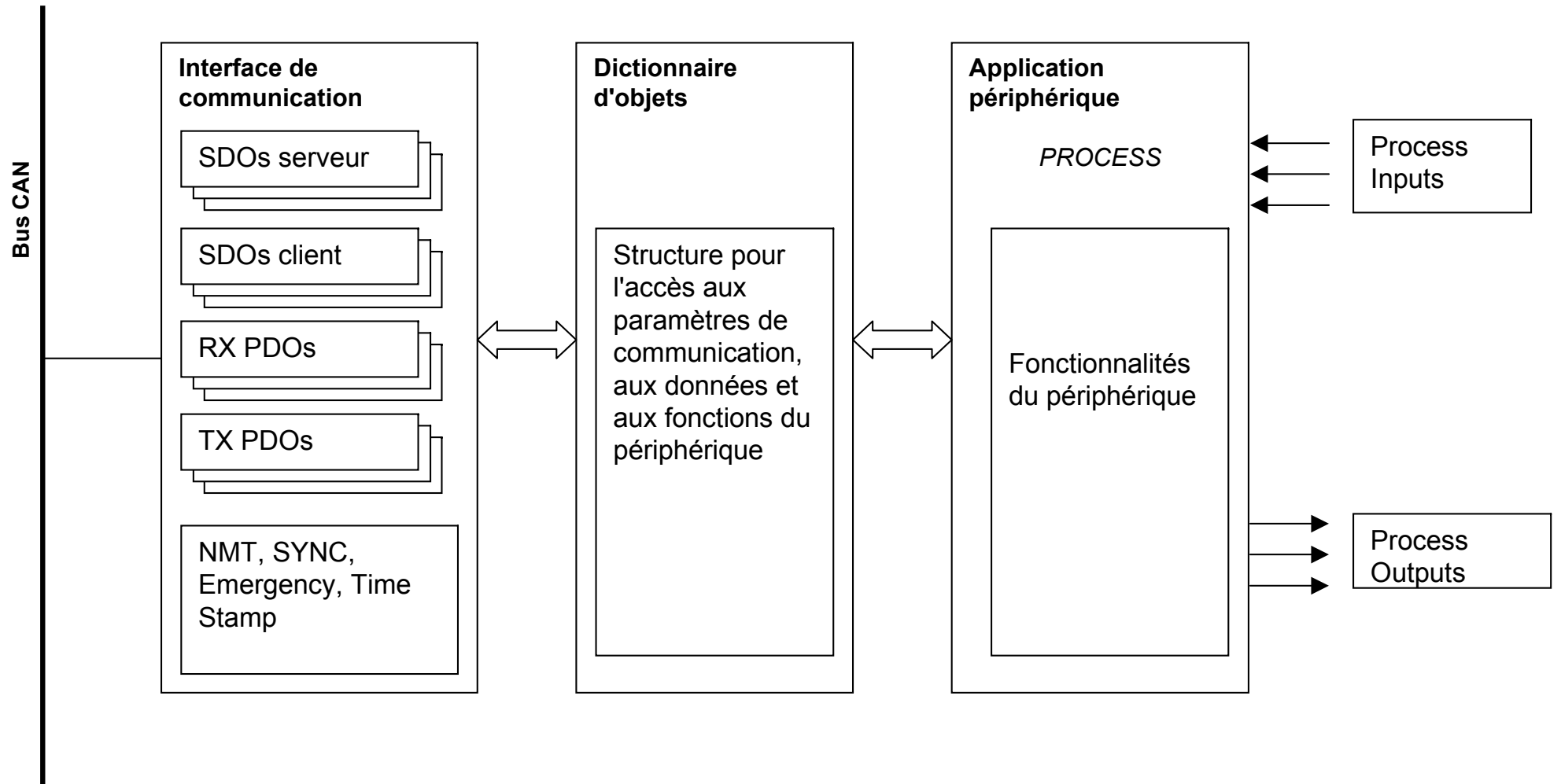
- **CiA-301**

- ▶ Couche d'application CANopen et Communication profile.

- **CiA-4xx**

- ▶ Device/Application profiles (Description de type d'équipements compatibles CANopen)
- ▶ Nombreux profiles :
  - CiA-401 : Entrées/sorties
  - CiA-402 : Contrôle mouvement/moteurs
  - CiA-404 : Systèmes de mesure (°C, hpa)
  - CiA-406 : Codeurs
  - CiA-407 : Informations passagers
  - CiA-412 : Systèmes médicaux
  - CiA-416 : Contrôle portes de bâtiments
  - CiA-417 : Ascenseurs
  - CiA-418 : Batteries / CiA-419 : Chargeurs
  - CiA-423 : Véhicules diesels sur rails
  - ...

# Protocole CANopen : *Modèle de Périphérique*



# Protocole CANopen : *Dictionnaire d'objets*

## Le dictionnaire d'objets

- Structure virtuelle d'accès au périphérique :
  - ▶ Informations du périphérique
  - ▶ Paramètres de configuration
  - ▶ Données de process
- Contient des objets accessibles en lecture et/ou écriture
- Objets communs à la norme ou spécifiques au périphérique
- Un dictionnaire d'objet par périphérique



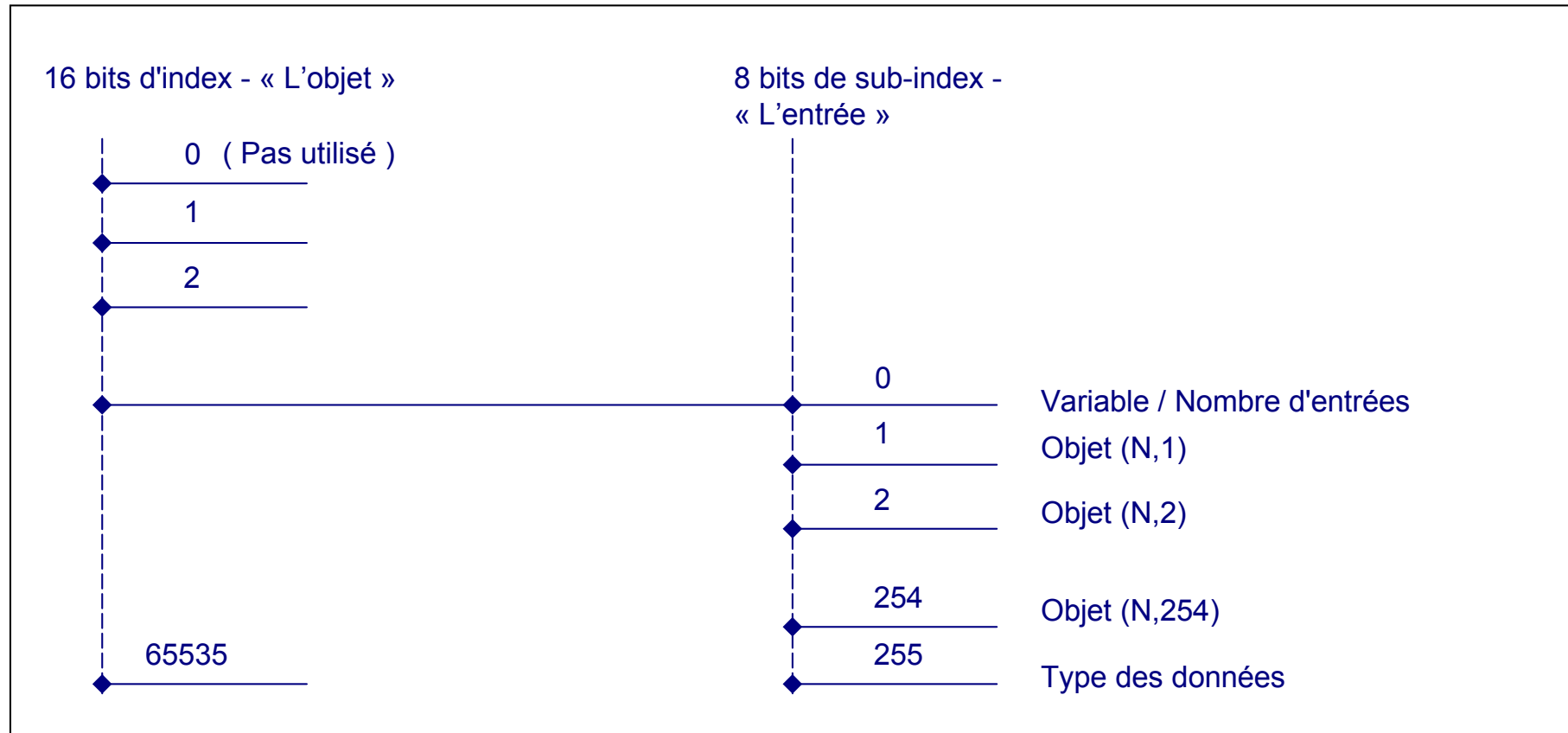
# Protocole CANopen : *Dictionnaire d'objets*

Structure contenant toutes les informations du périphérique pouvant transiter sur le réseau, soit en lecture, soit en écriture ou les deux.

Index	Objets
0000 <sub>h</sub>	Non utilisé
0001 <sub>h</sub> -025F <sub>h</sub>	Description des types de donnée
0260 <sub>h</sub> -0FFF <sub>h</sub>	Réservé
1000 <sub>h</sub> -1FFF <sub>h</sub>	<b>Zone du communication profile</b>
2000 <sub>h</sub> -5FFF <sub>h</sub>	<b>Zone spécifique fabricant</b>
6000 <sub>h</sub> -9FFF <sub>h</sub>	<b>Zone standardisée Device/Application profiles</b>
A000 <sub>h</sub> -AFFF <sub>h</sub>	Zone standardisée des variables réseau (PLC)
B000 <sub>h</sub> -BFFF <sub>h</sub>	Zone standardisée des variables système (multi-réseaux)
C000 <sub>h</sub> -FFFF <sub>h</sub>	Réservé

# Protocole CANopen : *Dictionnaire d'objets*

## Méthode d'accès



# Protocole CANopen : *Dictionnaire d'objets*

- **Points clés :**

- ▶ Structure d'objets à deux dimensions <index> / <sub-index>
- ▶ Séparation des variables d'application spécifiques et des informations partagées (réseau).
- ▶ Normalisation du contenu par les documents du CiA
- ▶ Possibilités d'objets customs
- ▶ Types d'objets disponibles variés :
  - Normalisés (variables simples,...)
  - Libres (nécessite alors une description par l'auteur)
- ▶ Une entrée (information) unique par sous-index

# Protocole CANopen : *Modèle de communication*

## Mécanismes d'échange de donnée :

Messages de type SDO

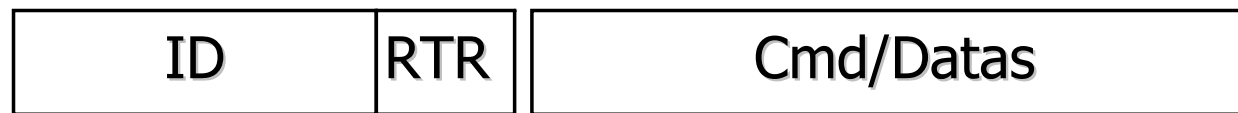
Messages de type PDO

# Protocole CANopen : *Modèle de communication*

## RAPPEL

### Correspondance Trames CAN - CANopen

- Chaque trame CANopen est composée de :



- ▶ ID : Identificateur CAN Layer 2 (11bits)
- ▶ RTR : Bit spécifiant une trame de requête (idem CAN Layer 2)
- ▶ Cmd/Datas : 8 octets de donnée CAN Layer 2

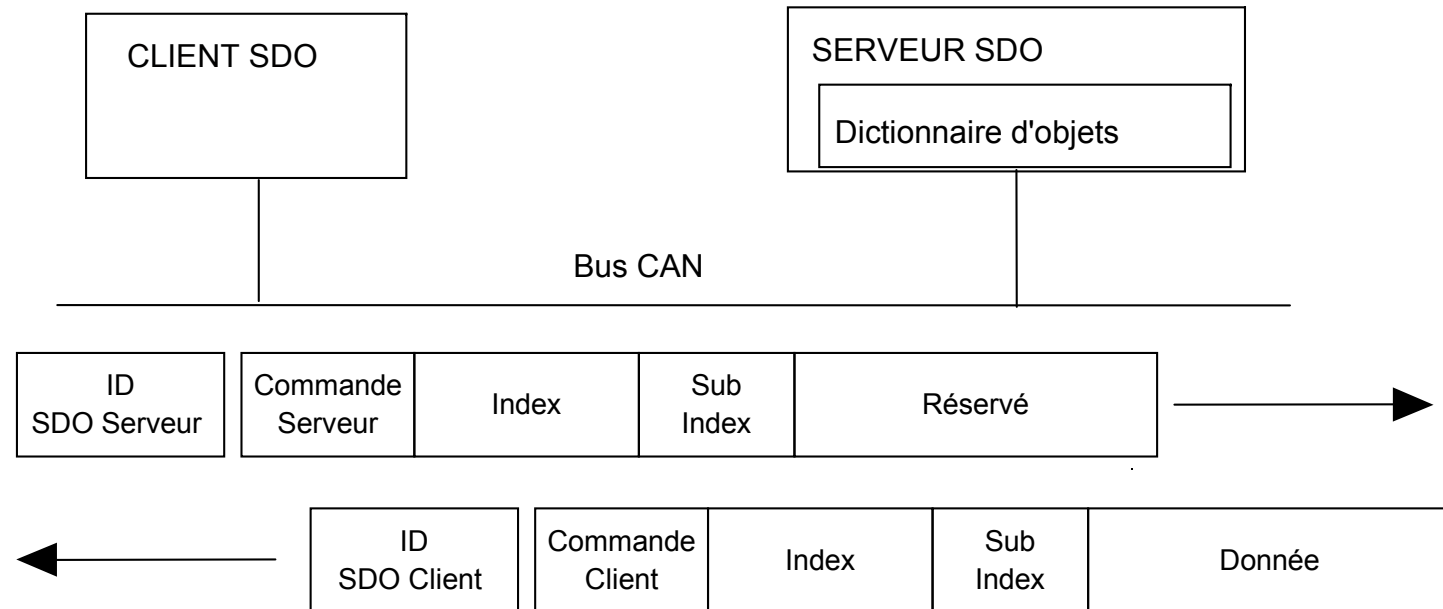
# Protocole CANopen : *Modèle de communication*

## Service Data Object (SDO)

- Service Client / Serveur :
  - ▶ Le client initie la liaison et le serveur répond
  - ▶ Transfert « confirmé »
  - ▶ 2 identificateurs CAN utilisés par canal SDO
- Permet l'accès au dictionnaire d'objets d'un périphérique par l'adressage d'une entrée via un index et un sub-index
  - ▶ Pas d'interprétation nécessaire
- Une seule information (entrée) accédée par transaction
- Possibilité de transfert de données de longueur illimitée (protocole segmenté)
- Principalement utilisé pour la configuration de périphériques

# Protocole CANopen : *Modèle de communication*

## Protocole SDO



Exemple: Un client SDO lit 4 octets de données dans le dictionnaire d'objets d'un périphérique (serveur SDO). Utilisation du protocole SDO.

# Protocole CANopen : *Modèle de communication*

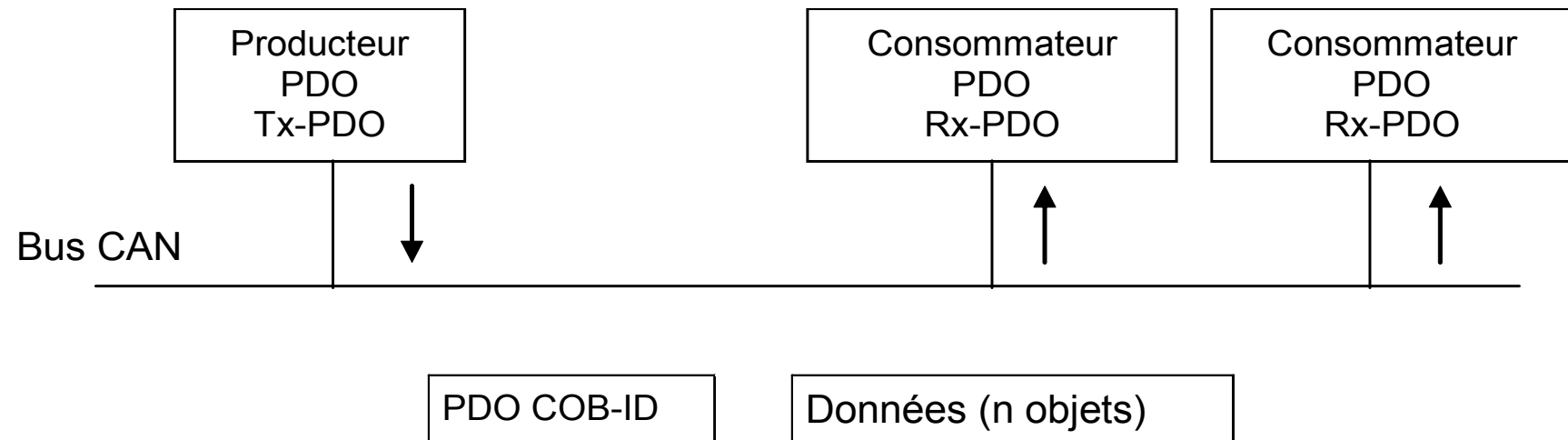
## Process Data Object (PDO)

- Liaison de type Producteur - Consommateur(s) :
  - Plusieurs modes de déclenchement de transmission
  - Un seul identificateur CAN par PDO
- Transfert de 8 octets de données maximum :
  - pas de protocole donc pas de segmentation possible.
  - Plusieurs informations (entrées) dans un même message (en fonction du mapping)
- Utilisé pour la transmission de données en temps réel
- Configuration des contenus de message par défaut
- Reconfiguration possible suivant périphérique



# Protocole CANopen : *Modèle de communication*

## Protocole PDO

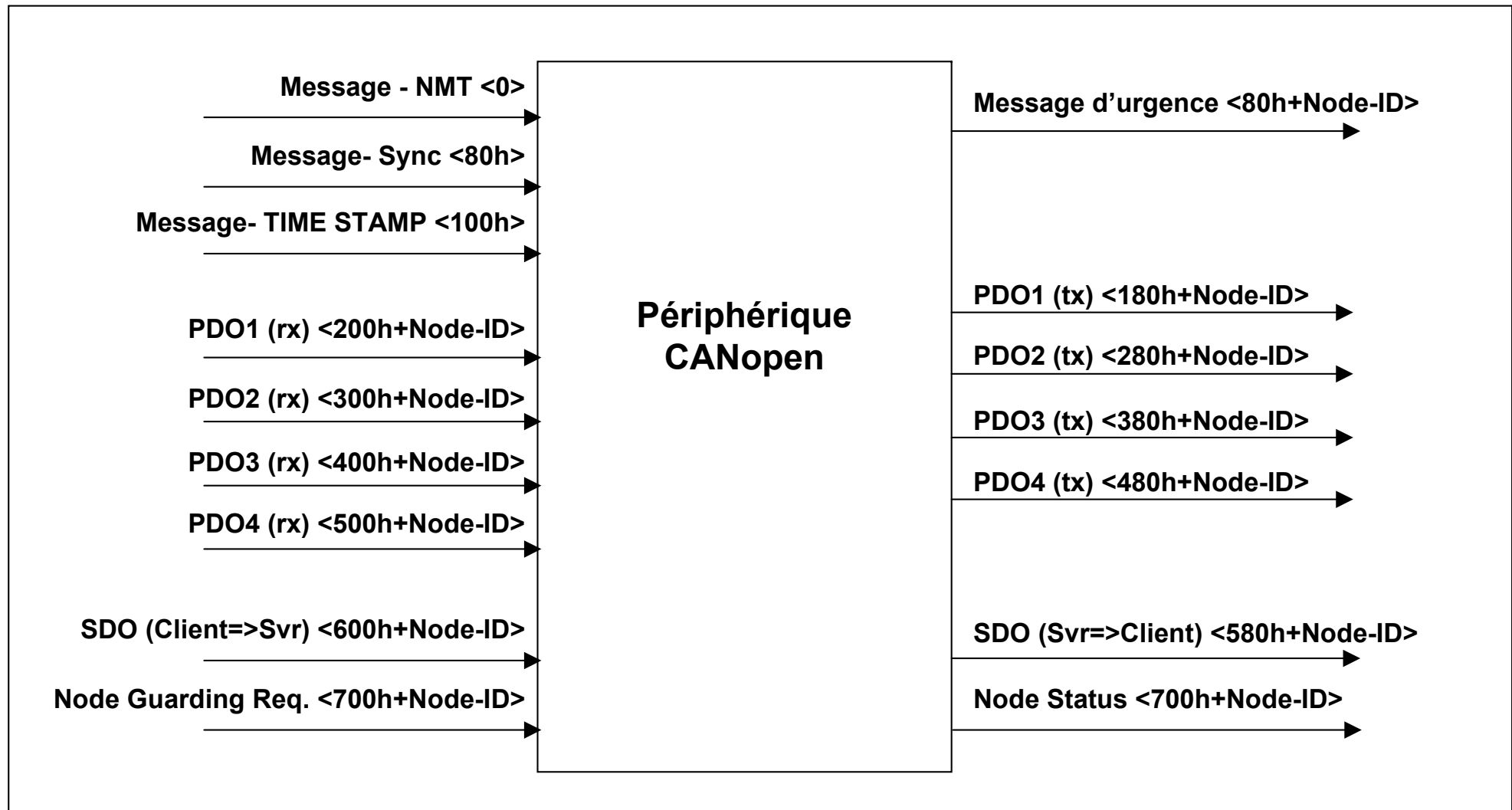


# Protocole CANopen : *Modèle de communication*

## Les Identificateurs CANopen

- Codés sur les ID 11 bits du bus CAN
- Basés par défaut sur l'adresse de l'équipement (Node-ID) :
  - Autorise jusqu'à 127 équipements sur le réseau
- Identificateurs spécifiques réservés pour les messages réseau génériques standard :
  - NMT (Network Management)
  - Synchronisation
  - Messages Time Stamp
- Identificateur par défaut périphériques basés sur le « Predefined Connection Set » :
  - Permet l'accès à un périphérique non configuré.
  - Permet l'implémentation de structures de systèmes simples (1:N)
- Libre configuration de certains identificateurs de messages :
  - Nécessaire pour les structures de communication complexes

# Protocole CANopen : *Modèle de communication*



# Protocole Devicenet

# Protocole Devicenet : Introduction

- Standard de réseau ouvert spécifié par l'Open Devicenet Vendor Association inc. (ODVA).

[www.odva.org](http://www.odva.org)

- Développé à l'origine par Allen Bradley.
- Marque déposée de l'ODVA.
- Adhésion à l'ODVA est obligatoire pour tout fabricant de périphérique DeviceNet
- Documents de spécifications :
  - Volume 1 (Communication Model and Protocol)
  - Volume 2 (Device Profiles and Object Library)

# Protocole Devicenet : Introduction

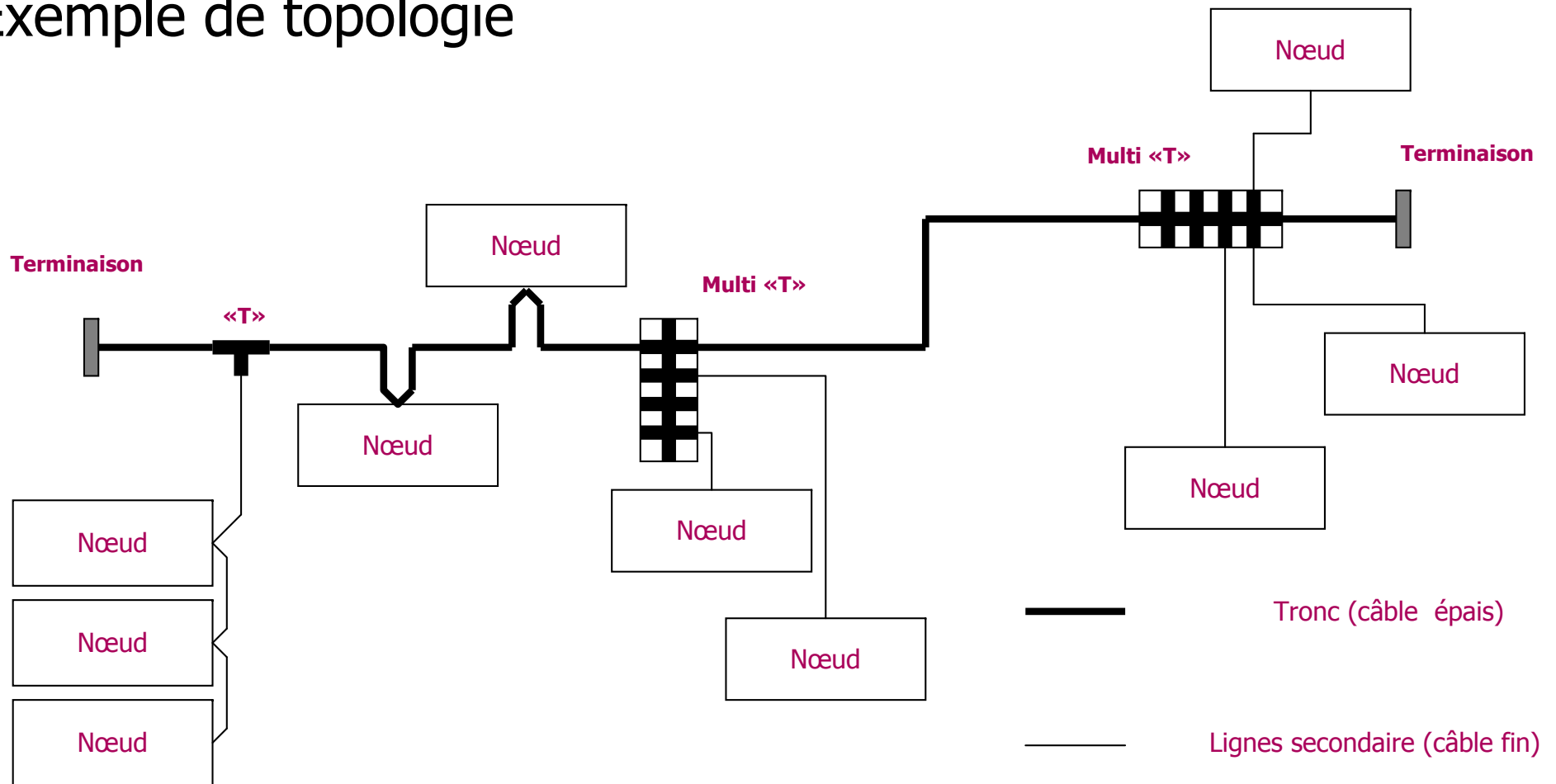
- Basé sur la technologie C.I.P. (Common Industrial Protocol)
  - ▶ comme ControlNet, CompoNet et EtherNet/IP
  - ▶ Document normatif supplémentaire : DeviceNet adaptation of CIP
  - ▶ Principe de modélisation des équipements à base d'objets
  - ▶ Chaque objet CIP possède :
    - des attributs (données)
    - des services (commandes)
    - des connexions
    - des comportements (relations entre les services et les valeurs des attributs).
  - ▶ Existence de bibliothèques d'objets regroupées en device profiles

# Protocole Devicenet : Caractéristiques techniques

- Topologie de type bus ou arborescence mais standardisée
- Supporte jusqu'à 64 nœuds
- 3 vitesses configurables
  - ▶ 125kBaud (longueur maximum 500 m )
  - ▶ 250kBaud (longueur maximum 250 m )
  - ▶ 500kBaud (longueur maximum 100 m )
- Signaux et alimentation distribués par le même câble
- Branchement et remplacement de périphériques à chaud possible.

# Protocole Devicenet : Caractéristiques techniques

- Exemple de topologie





# Protocole Devicenet : Modèle de communication

- Modèle ISO et DeviceNet

ISO-layer7	Application layer	DeviceNet Specification Vol. II
ISO-layer2	Data link layer	Bosch CAN Specification 2.0
ISO-layer1	Physical signaling	Bosch CAN Specification 2.0
ISO-layer1	Transceiver	DeviceNet Specification Vol. I
ISO-layer0	Transmission media	DeviceNet Specification Vol. I

# Protocole Devicenet : Modèle de communication

- Les structures de communication sont :
  - ▶ Master/Slave
  - ▶ Multimaster
  - ▶ D'égal à égal
- Les modèles de communication sont :
  - ▶ Connexion point à point
  - ▶ connexion multicast
  - ▶ modèle producteur/consommateur
- Deux types de message.
  - ▶ Explicit Messages (fonctions de type Client/Serveur)
  - ▶ I/O Messages (échange de données rapide)

# Protocole Devicenet

## Terminologie Objets

- DeviceNet utilise un **Object Model** abstrait pour décrire :
  - ▶ l'ensemble des services de communication disponibles
  - ▶ le comportement d'un nœud visible de l'extérieur
  - ▶ un moyen commun par lequel l'information est échangée.
- Un nœud DeviceNet est constitué d'**Objects**.
- Un **Object** est une représentation abstraite d'un comportement particulier d'un produit

## Terminologie Objets

- Class

- ▶ Le terme de **Class** représente un ensemble d'**objects** représentant le même type de composant système.
- ▶ Une **Class** est une généralisation d'un **object**
- ▶ Tous les **objects** d'une classe sont identiques en forme et en comportement, mais peuvent avoir des valeurs d'attribut différentes.

- Instance

- ▶ Une **Instance** représente d'un des objets d'une **Class**.
- ▶ Les termes **Object**, **Instance** ou **Object Instance** se réfèrent tous à une Instance X d'une Classe Y.

# Protocole Devicenet

## Terminologie Objets

- Une instance d'objet et/ou une classe d'objet
  - ▶ contient des attributs (***Attributes***)
  - ▶ fournit des ***Services***
  - ▶ implémente un Comportement (***Behaviour***)
- Chaque instance d'objet d'une classe a le même jeu d'attributs, de services et de comportements, mais existe dans son propre état.

# Protocole Devicenet

## Terminologie Objets

- Les attributs sont des variables contenant les objets de données.
- Typiquement les attributs fournissent une information d'état ou gouvernent l'action d'un objet :
  - ▶ état d'un objet
  - ▶ valeur de temps
  - ▶ nom de produit
  - ▶ numéro de série
  - ▶ données de processus
  - ▶ ...

# Protocole Devicenet

## Terminologie Objets

- Services

- ▶ Les services sont appelés pour réaliser des opérations sur l'état d'un objet ou sur ses attributs.
- ▶ Les services courants sont par exemple :
  - Read Data ( ex. Get\_Attribute\_Single )
  - Write Data (ex. Set\_Attribute\_Single )
  - Reset
  - Create
  - Delete
  - ...

# Protocole Devicenet

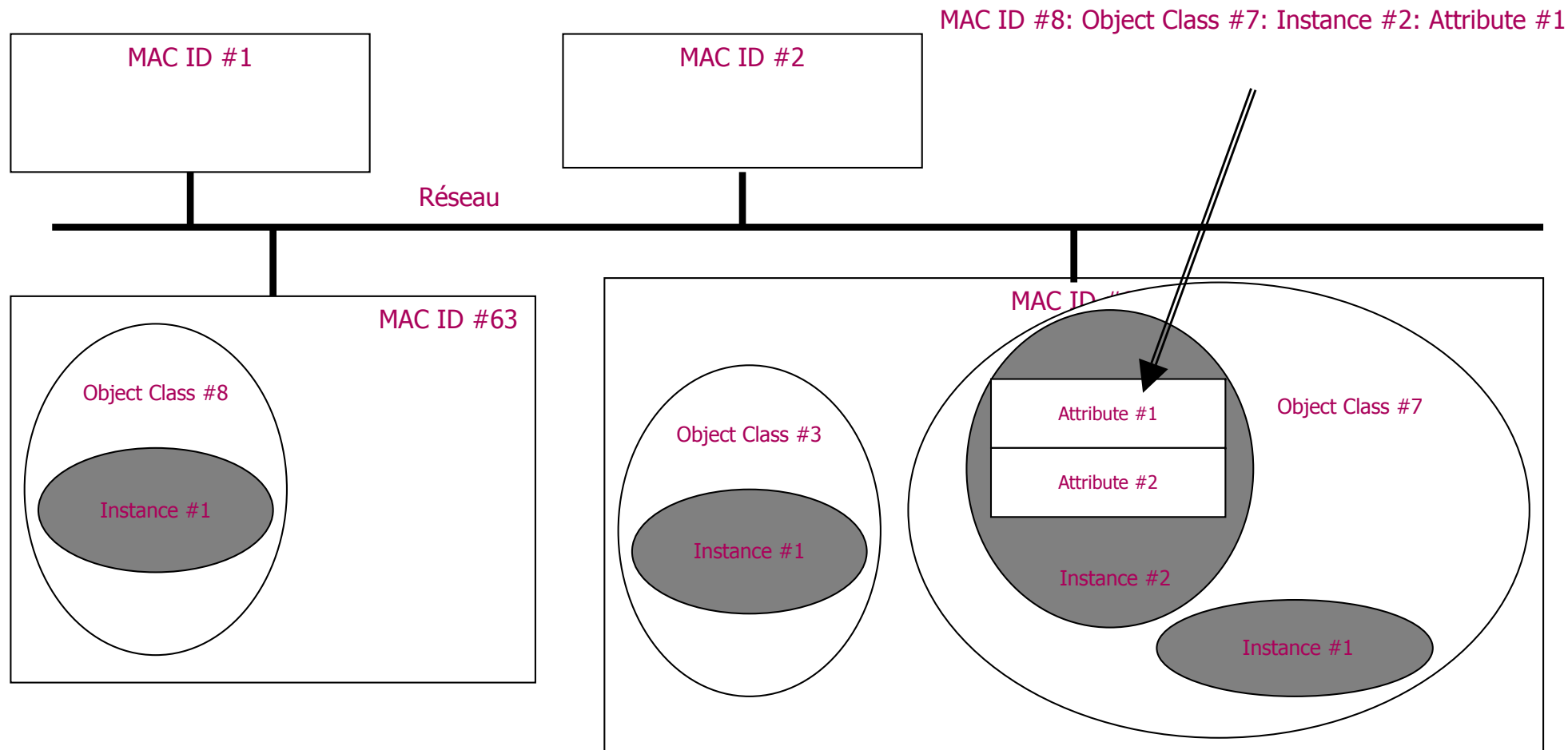
## Terminologie Objets

- Le comportement d'un objet indique comment il réagit sur un événement particulier.
- Ceci peut être implémenté par l'application en fonction d'un appareil :
  - ▶ allumer une ampoule
  - ▶ fermer une vanne
  - ▶ démarrer un moteur
  - ▶ ...



# Protocole Devicenet

## Modes d'adressage



# Protocole Devicenet

## Modes d'adressage, exemple pour le nœud n°5

Class of the coupler:

Object	Class	Instance	Description
Identity	0 x 01	1	Device type, vendor ID, serial number etc.
Message Router	0 x 02	1	Routes
DeviceNet	0 x 03	1	Maintains object connections
Assembly	0 x 04	9	
Connection class	0 x 05	3	Allows
Acknowledge handler	0 x 2B	1	The Acknowledge handler receives commands within the protocol
Coupler configuration object	0 x 64		
Discrete input point	0 x 65		
Discrete output point	0 x 66		
Analog input point	0 x 67		
Analog output point	0 x 68		

Identity Class (0 x 1):						
Instance 0:						
Number	Used in buscoupler	Access rule	Name	Data type	Description	Value
1	required	get	Revision	UINT	Revision of the Identity Object, Range 1-65535, class definition upon which the implementation is based.	0 x 01

Instance 1:						
Attribute ID	Used in buscoupler	Access rule	Name	Data type	Description	Value
1	required	get	Vendor	UINT	Identification of vendor	40 (0 x 28)
2	required	get	Device Type	UINT	Indication of general type of product	12 (0 x C)
3	required	get	Product Code	UINT	Identification of particular product of an	306(0 x 132)

L'objet identification du vendeur à comme "adresse" : MAC ID #5: Object Class ID #0x01: Instance ID #1: Attribute ID #1

# Protocole Devicenet

## Modes d'adressage

- Des ***Service Codes*** sont associés à chaque Class.
- Ils sont appelés par des ***Explicit Messages***.
- Les 2 principaux codes sont :
  - ▶ 0x0E : *Get\_Attribute\_Single*, pour lire une valeur d'attribut
  - ▶ 0x10 : *Set\_Attribute\_Single*, pour écrire une valeur d'attribut

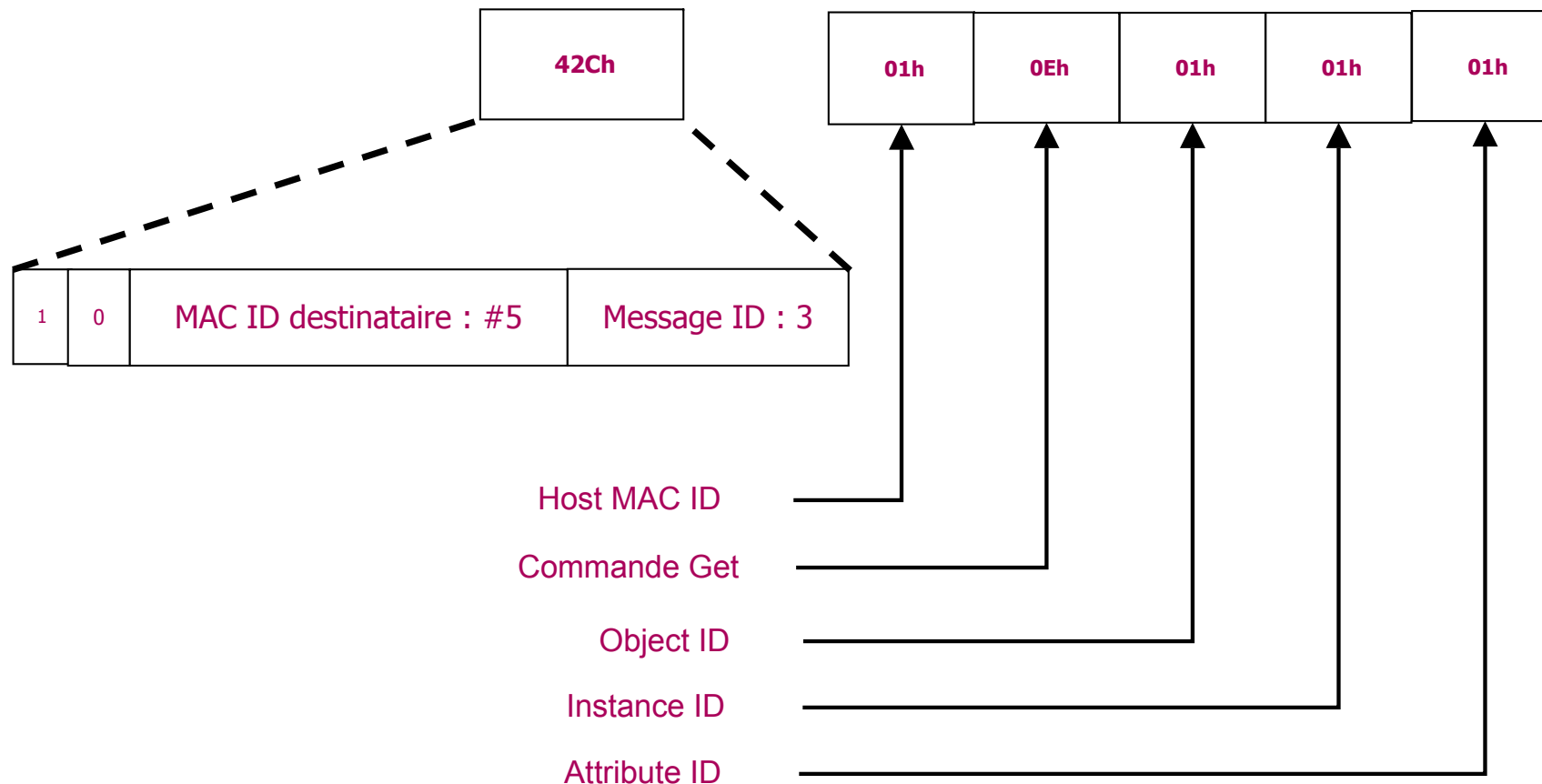
# Protocole Devicenet

## Utilisation des identificateurs CAN

Identificateur sur 11 bits											plage d'ID		
10	9	8	7	6	5	4	3	2	1	0			
0	Groupe 1 Message ID			Source MAC ID							000 - 3FF (1024 Id)	Groupe de message 1	
1	0	Source/destinataire MAC ID					Groupe 2 Message ID				400 - 5FF (512 Id)	Groupe de message 2	
1	1	Groupe 3 Message ID			Source MAC ID							600 - 7BF (448 Id)	Groupe de message 3
1	1	1	1	1	Groupe 4 Message ID						7C0 - 7EF (48 Id)	Groupe de message 4	
1	1	1	1	1	1	1	X	X	X	X	7F0 - 7FF	ID non valide	

# Protocole Devicenet

## Exemple de trame DeviceNet

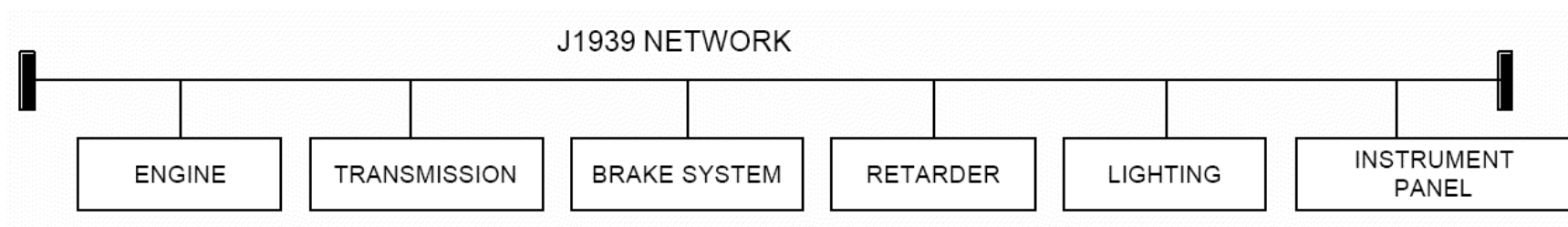


# Protocole J1939

# Protocole J1939

## Caractéristiques principales

- Protocole Applicatif basé sur le bus CAN, le J1708 et le J1587
- Utilisé pour la communication entre les ECU (Electronic Control Unit)
- Domaine d'application principale : Camion, engins lourds & spéciaux
- Peu de surcharge protocole car très orienté données
- Exemple de réseau :



# Protocole J1939

## Caractéristiques principales

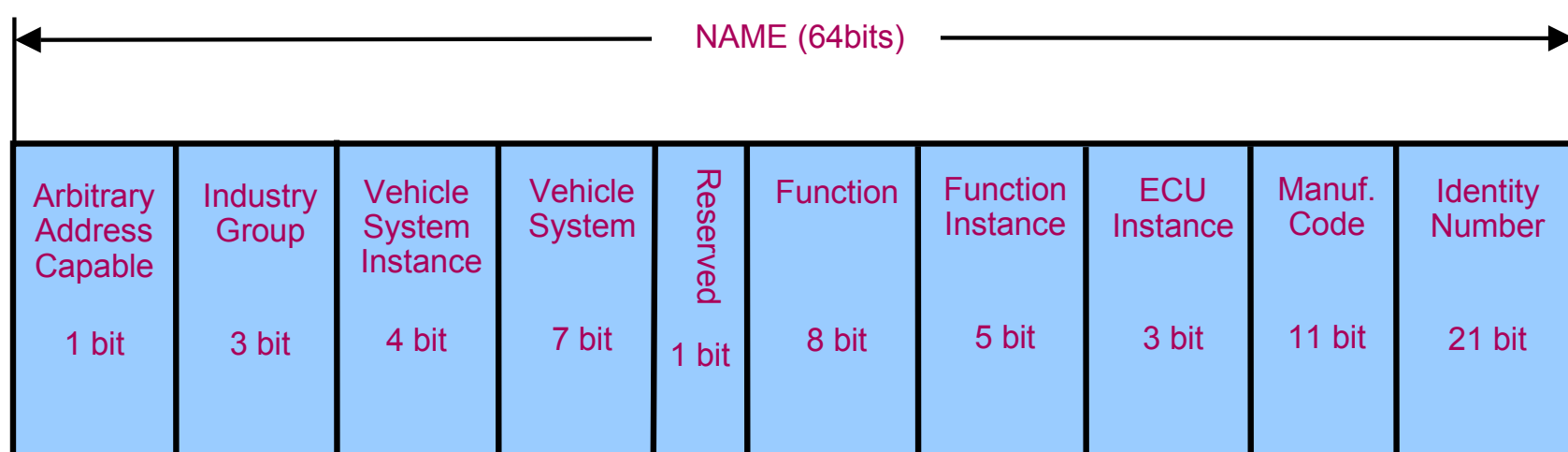
- Un réseau est composé d'ECUs (Electronic Control Units)
- Chaque ECU peut contenir un ou plusieurs CA (Contrôleur d'application)
- Chaque CA est caractérisé par :
  - ▶ une adresse
  - ▶ un "NAME"
  - ▶ Son propre programme
- Trois types de CA :
  - ▶ Standard CA
  - ▶ Diagnostic/Development tools CA
  - ▶ Network Interconnection CA



# Protocole J1939

## Caractéristiques principales

- Adresses des CA unique sur 8 bits :
  - ▶ Deux types de CA : Arbitrary Address Capable / Single Address Capable
  - ▶ Les adresses permettent :
    - L'utilisation d'ID CAN uniques
    - L'identification de la source du message
- Paramètre "Name" des CA :



# Protocole J1939

## Documents officiels

- **J1939/0X**

Document d'application du protocole où X fait référence à un domaine spécifique, par ex. :

- ▶ J1939/01 *Truck and Bus Control and Communications Network*
- ▶ J1939/02 (Draft) *Agricultural Equipment Control and Communications Network*

- **J1939/1X**

Document « Physical Layer » document, où X fait référence à une architecture physique spécifique :

- ▶ J1939/11 *Physical Layer, 250K Bits/sec, Shielded Twisted Pair*
- ▶ J1939/12 (Draft) *Physical Layer, 250K Bits/sec, Twisted Quad*
- ▶ J1939/13 *Physical Layer, Diagnostic Connector*
- ▶ J1939/15 (Draft) *Reduced Physical Layer, 250K bits/sec, Unshielded Twisted Pair (UTP)*

# Protocole J1939

## Documents officiel

- **J1939/21**  
Document de spécification de la couche Data Link Layer
- **J1939/3X**  
Document de spécification de la couche Réseau:
  - ▶ J1939/31 Network Layer
- **J1939/7X**  
Document de spécification de la couche Application où X fait référence à une application spécifique .
  - ▶ J1939/71 *Vehicle Application Layer*
  - ▶ J1939/73 *Diagnostic Application Layer*
  - ▶ J1939/73 *Configurable Messaging Application Layer*

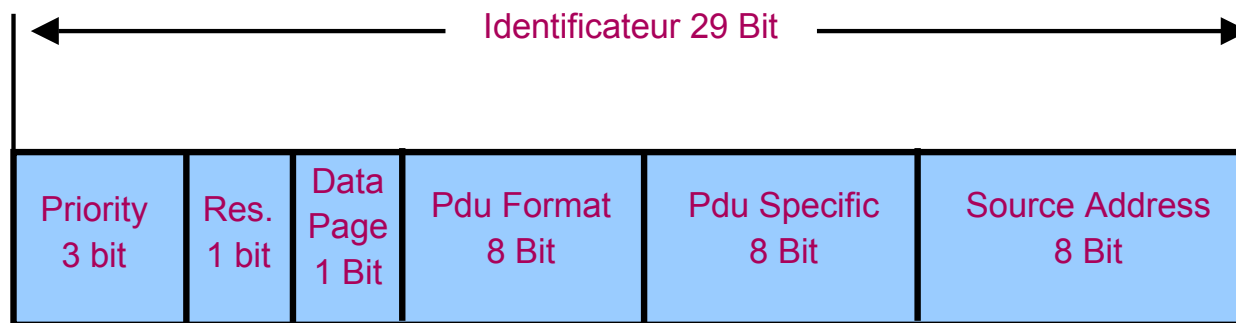
# Protocole J1939

## Caractéristiques techniques

- Bas niveau :
  - ▶ Basé sur CAN 2.0B :
    - Identifiant sur 29-bits (Extended-ID)
    - Vitesse bus : 250 kBits/s
  - ▶ Longueur max. : 40m
- Partie applicative
  - ▶ Communication Point-à-point et Broadcast
  - ▶ Protocoles de transport allant jusqu'à 1785 octets de données
  - ▶ Fonctionne sur le principe des groupes de paramètres
  - ▶ Gestion réseau spécifique
  - ▶ Découpage ID CAN spécifique

# Protocole J1939

## Interprétation de l'identificateur



<PDU Format><PDU Specific> = PGN (Parameter Group Number)

# Protocole J1939

## Groupes de paramètres

- Regroupent dans les messages des données similaires ou en relation
- Définis dans la norme SAE J1939-71 avec leur contenu
- Des paramètres « custom » peuvent également être créés et utilisés.
- Chaque groupe de paramètre est identifié de manière unique par une valeur :
  - ▶ Le PGN (Parameter Group Number)  
Codé sur 16 bits composé de deux informations :
    - PDU Format
    - PDU Specific

# Protocole J1939

## Groupes de paramètres

- Pour un PGN donné, les informations disponibles sont :
  - ▶ Nom : nom du paramètre
  - ▶ Transmission Repetition Rate : périodicité
    - Data Length : longueur des données de la trame CAN transmises
    - Extended Data Page : Pages de donnée étendue
    - Data Page : Page de donnée
    - Valeurs PDU Format et PDU Specific (déductibles du PGN)
    - Default Priority : priorité
    - Contenu : Liste, taille et emplacement des SPN dans le champ de donnée de la trame

# Protocole J1939

## Groupes de paramètres - Exemple

- Nom : **Brakes**

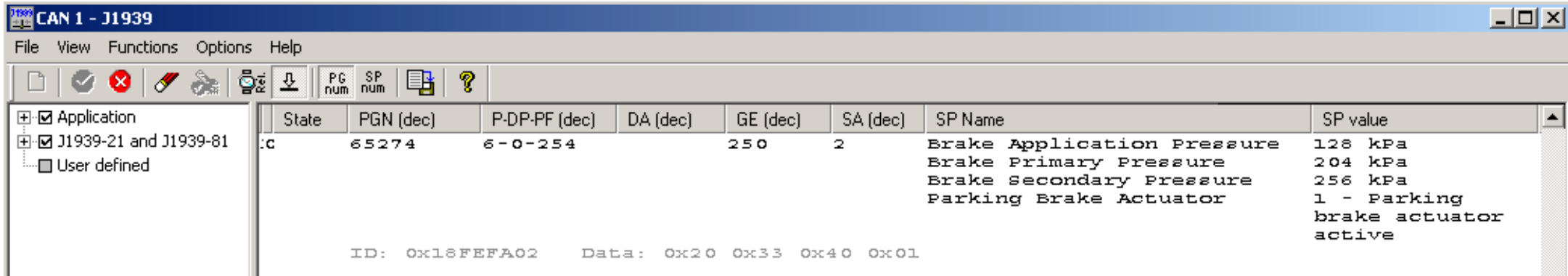
- ▶ Transmission Repetition Rate : 1 s
- ▶ Data Length : 8
- ▶ Extended Data Page : 0
- ▶ Data Page : 0
- ▶ PDU Format : 254
- ▶ PDU Specific : 250
- ▶ Default Priority : 6
- ▶ Parameter Group Number : 65274 (0xFEFA)
- ▶ Contenu :

Position	Taille	Nom du paramètre	SPN
1	1 byte	Brake Application Pressure	116
2	1 byte	Brake Primary Pressure	117
3	1byte	Brake Secondary Pressure	118
4.1	2 bits	Parking Brake Actuator	619



# Protocole J1939

## Exemple de message



State	PGN (dec)	P-DP-PF (dec)	DA (dec)	GE (dec)	SA (dec)	SP Name	SP value
	65274	6-0-254		250	2	Brake Application Pressure	128 kPa
						Brake Primary Pressure	204 kPa
						Brake Secondary Pressure	256 kPa
						Parking Brake Actuator	1 - Parking brake actuator active

ID: 0x18FEFA02    Data: 0x20 0x33 0x40 0x01

# Le CAN & ses protocoles

- Contacts ISIT :
  - ▶ Coordonnées :

7 rue André-Marie AMPERE  
31830 Plaisance du Touch  
Tél : 05 61 30 69 00 - Fax : 05 61 16 50 63  
<http://www.isit.fr>
  - ▶ Service Technique :
    - E-mail : [support@isit.fr](mailto:support@isit.fr)
    - CAN, CANopen, Expertise, Embarqué : Franck MONTAGNE - Yvelain NAUDE
  - ▶ Service Commercial :
    - E-mail : [contact@isit.fr](mailto:contact@isit.fr)
    - Bus de terrain : Walid CHELBAB