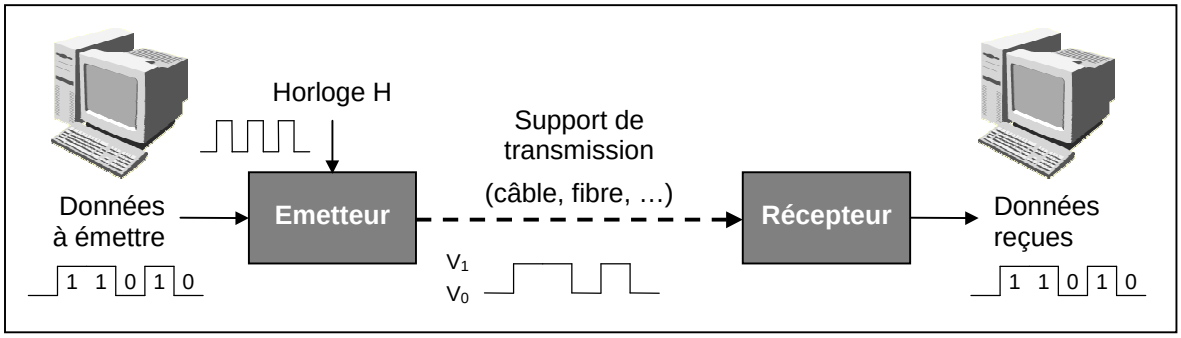
Chapitre 28

Transmissions numériques

1. Généralités

Les réseaux informatiques se fondent sur la numérisation des informations consistant à représenter les données par des suites de « **0** » et de « **1** ». La figure suivante représente une liaison numérique entre deux équipements informatiques. L'un des équipements joue le rôle d'**émetteur** tandis que l'autre équipement, celui de **récepteur**.



Le problème consiste à retrouver à l'autre extrémité de la chaîne de **transmission numérique** les mêmes données qu'à la source. Ces données sont transmises suivant un rythme régulier, fixé par un signal de référence appelé l'**horloge H**.

Pour transmettre ces données sur le **support de transmission**, il est nécessaire de les transformer au préalable en signal électrique. La méthode la plus simple consiste à représenter les éléments binaires « **0** » et « **1** » par deux tensions $\mathbf{V_0}$ et $\mathbf{V_1}$ (voir chapitre 27 - page 324).

2. Modes de transmission

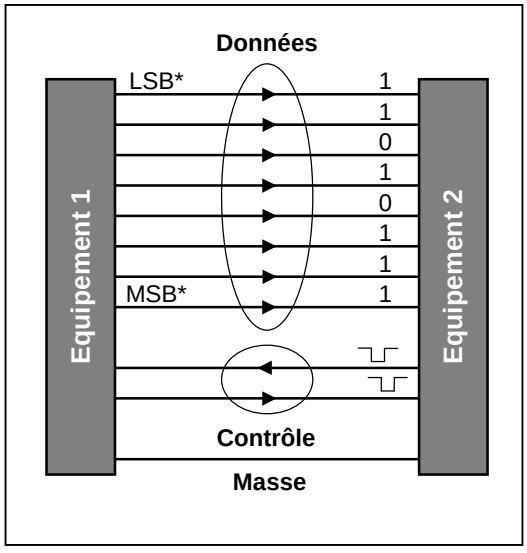
2.1. Transmission parallèle

Lorsque les équipements informatiques sont séparés par une **courte distance**, on peut envisager une transmission **parallèle** (octet par octet généralement), ce qui a pour effet d'accroître le **débit**.

Les bits de la donnée sont transmis **simultanément**.

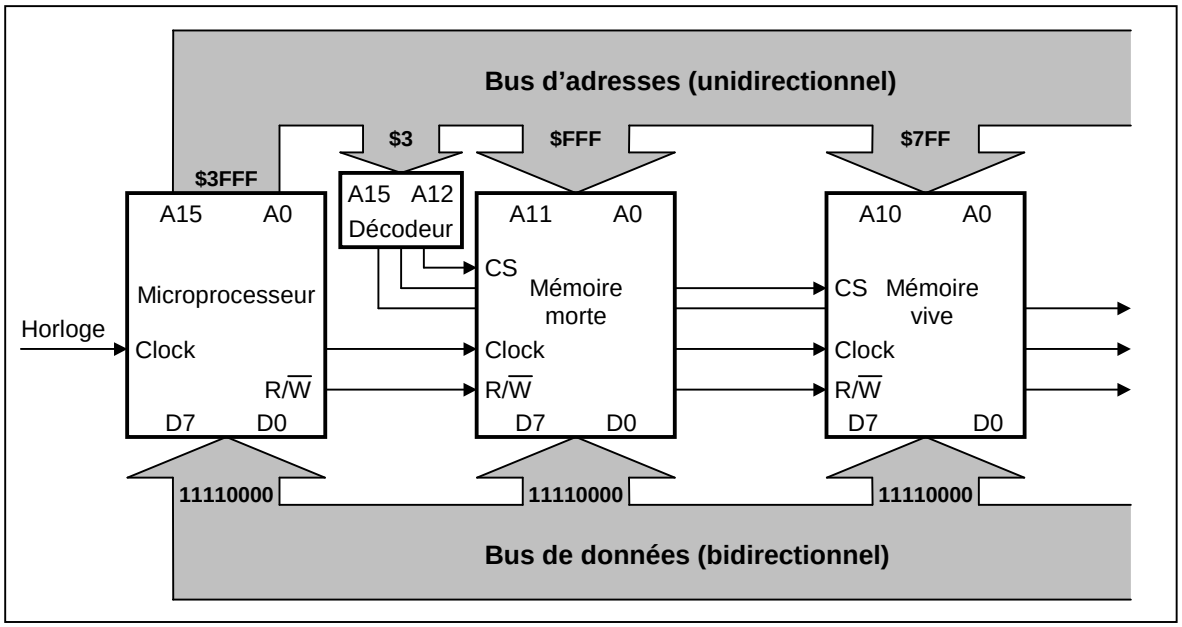
Les équipements à relier comportent autant de fils de données que de bits à transmettre, un ou plusieurs fils de **contrôle** cadencent la transmission.

Cette transmission est **rapide** mais inadaptée aux transferts sur de longues distances en raison de son **coût**, de son **encombrement** (un fil par bit) et de sa **sensibilité** à l'environnement électrique.



* ***LSB*** *: **L**east **S**ignificant **B**it (bit de poids le plus faible)*
MSB *: **M**ost **S**ignificant **B**it (bit de poids le plus fort)*

L'exemple classique de **transmission parallèle** est celui des liaisons entre un microprocesseur et les mémoires ou autres gestionnaires de périphériques (voir page suivante).



On utilise dans ce cas un « **bus** » (ensemble de lignes transportant en **parallèle** des informations de même type) pour les adresses et un autre pour les données. Il en existe également un pour les bits de contrôle (Read / Write, etc.) La seule remarque à faire concernant ce mode de transport est la nécessité d'utiliser un **décodeur** qui, en fonction des valeurs des bits de poids fort (A15 … A12) du bus d'adresses, sélectionnera tel ou tel **circuit mémoire** grâce aux broches **C**hip **S**elect.

Ainsi, pour une même valeur des bits de poids faible du bus d'adresses (A11 … A0), seul le circuit sélectionné, grâce aux bits de poids fort (A15 … A12), sera actif.

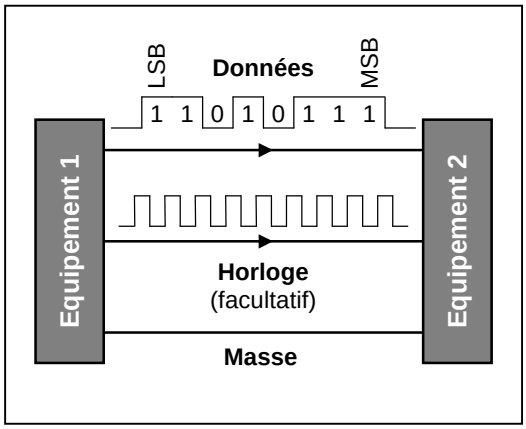
2.2. Transmission série

Lorsque les équipements informatiques sont séparés de plus de quelques mètres, on utilise la transmission **série** où les bits sont envoyés **les uns derrière les autres** sur un unique support de transmission.

Les équipements à relier ne comportent qu'un seul fil pour la transmission des données.

L'horloge peut être câblée ou non entre l'émetteur et le récepteur. Dans ce cas, la transmission est dite **synchrone** et **asynchrone** dans le cas contraire.

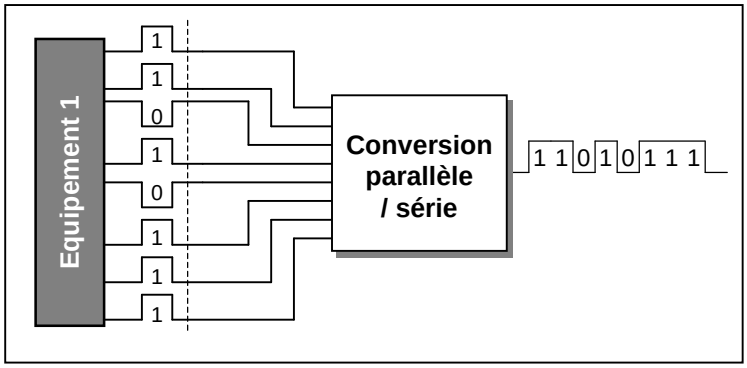
Cette transmission est **lente** mais permet de couvrir des distances importantes (réseaux informatiques).



Les liaisons effectuées par l'intermédiaire d'ondes infrarouges ou lumineuses, d'ondes radio, de fibre optique, etc. sont de type **série** car le support ne peut transmettre qu'une information à la fois.

Le traitement de l'information au sein d'un système numérique s'effectue sur des mots binaires parallèles. Or, lors du transport, ils sont envoyés en série. Il faut donc avant l'émission une **conversion parallèle / série** et à la réception, la **conversion inverse** pour retrouver l'information.

Ceci est réalisé au moyen de deux **registres à décalage** (cf. p 287).

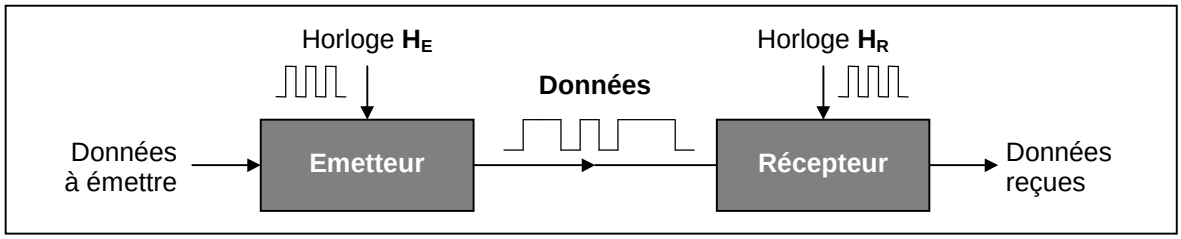


3. Synchronisation

La transmission d'informations sous forme numérique exige en général un synchronisme précis entre l'émetteur et le récepteur. Cette synchronisation est essentielle pour la reconstitution des données et peut être mise en œuvre par différents procédés.

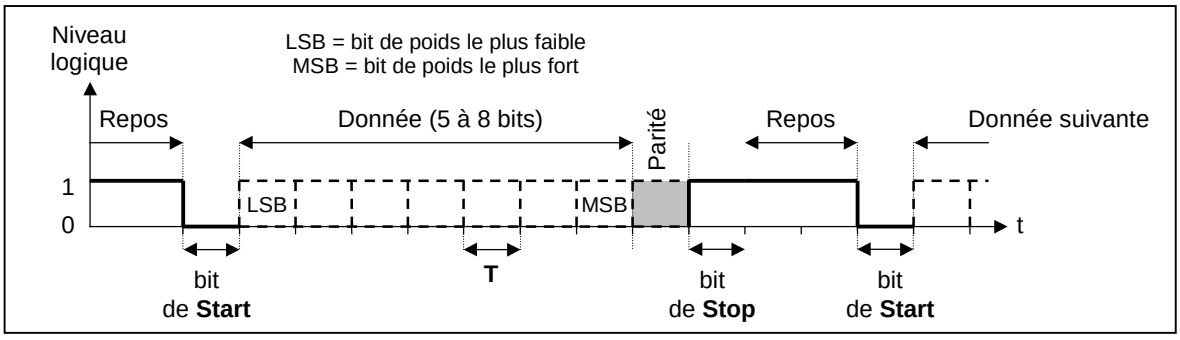
3.1. Transmission asynchrone

La transmission est **asynchrone** car l'horloge d'émission $\mathbf{H_E}$ n'est pas transmise vers le récepteur ni physiquement, ni implicitement dans le signal lui-même : les horloges $\mathbf{H_E}$ et $\mathbf{H_R}$ sont indépendantes.



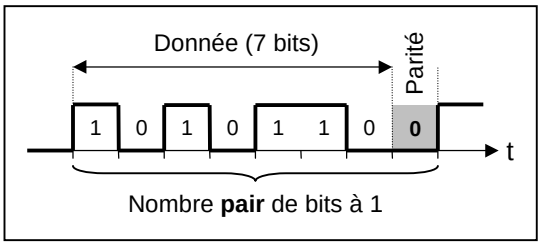
Les données sont émises de façon **irrégulière**, au moment où celles-ci sont disponibles, sans tenir compte des données précédentes ou suivantes. Il est donc nécessaire d'introduire des éléments de repérage (bits de **Start** et de **Stop**) permettant non seulement de reconnaître le début et la fin de chaque donnée, mais aussi d'assurer la synchronisation entre l'émetteur et le récepteur.

Dans le récepteur, l'horloge locale $\mathbf{H_R}$ de même période **T** que celle de l'émetteur $\mathbf{H_E}$ est activée au début de chaque donnée émise par le signal **Start** et désactivée par le signal **Stop**.



Un **bit de parité** facultatif permet le contrôle de la transmission. Entre deux données successives, la ligne peut être inactive pendant une durée variable (Repos).

Dans le cas d'une parité **paire**, on ajoute aux **n** bits de la donnée un bit supplémentaire, calculé pour que le mot à (**n + 1**) bits émis comporte un nombre **pair** de bits à « **1** ». Le récepteur vérifie la parité du mot reçu. Une erreur sur un bit sera détectée, mais deux non, car la parité restera inchangée.



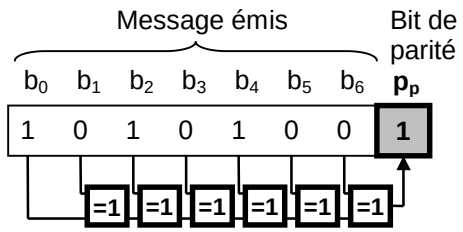
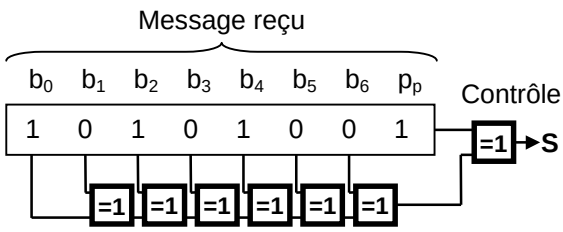
- **Paramètres de la transmission** :

- Durée du **Start** :	1 temps élémentaire d'horloge (T)
- Durée du **Stop** :	1T - 1,5T ou 2T (on utilise couramment 1T)
- Protection contre les erreurs :	Parité **paire** (Even parity) ou parité **impaire** (Odd parity)
- Vitesse de transmission :	**300**, **600**, **1200**, **2400**, **9600** ou **19200 bauds** (1 baud = 1 bit transmis par seconde)

La configuration de l'émetteur et du récepteur doit être semblable. L'exemple le plus connu de transmission série asynchrone est sans conteste la liaison série **RS232**, qui a fait les beaux jours de l'informatique industrielle naissante avec le **RS485** des milieux industriels (cf. page suivante).

3.1.1. Générateur du bit de parité

Dans le cas d'une **parité paire** avec **n = 7**, on a :

Emission	Réception
	
Le bit $\mathbf{p_p}$ de parité paire s'obtient par la relation : $$\mathbf{p_p = b_0 \oplus b_1 \oplus \cdots \oplus b_{n-2} \oplus b_{n-1}}$$ Le code obtenu à **(n + 1) bits (p_p, b_{n-1}, ... , b_0)** comporte un nombre **pair** de bits à **1**. Une parité impaire correspond au système inverse : $\mathbf{p_i = \overline{p_p}}$	Il y a une erreur si **S = 1** et aucune si **S = 0**. Remarques : - Si un bit à **0** est modifié à **1**, et un autre bit passe de **1** à **0**, on ne détecte pas d'erreur. - Si une erreur survient sur le bit de parité, la vérification est faussée.

La méthode du bit de parité, si elle a l'avantage de limiter la quantité de données insérées (un seul bit), n'est pas très fiable. En effet, le contrôle devient difficile voire impossible lorsque plusieurs erreurs surviennent dans le message transmis.

3.1.2. Tableau comparatif des normes de transmission série asynchrone

Paramètres	RS232	RS423	RS422A	RS485 (1)
Structure	Asymétrique (2)	Asymétrique (2)	Symétrique (3)	Symétrique (3)
Longueur maxi.	15 m	1200 m	1200 m	1200 m
Débit maxi.	20 kbauds	100 kbauds	10 Mbauds	10 Mbauds
Niveau haut (« 1 »)	de - 3 V à - 25 V	de - 0,2 V à - 6 V	de - 0,2 V à - 6 V	de - 0,2 V à - 7 V
Niveau bas (« 0 »)	de + 3 V à + 25 V	de + 0,2 V à + 6 V	de + 0,2 V à + 6 V	de + 0,2 V à + 7 V
Nombre d'émetteurs maxi.	1	1	1	32
Nombre de récepteurs maxi.	1	10	10	32

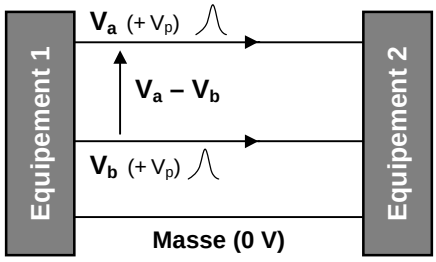
(1) : Plusieurs émetteurs et récepteurs peuvent dialoguer en même temps ⇒ **Bus multipoint** (par opposition à une **liaison point à point** : 1 émetteur – 1 récepteur)

(2) : Un **seul fil** est utilisé pour la transmission des données (V_a), ainsi qu'un **fil de masse** (0 V). La valeur binaire est déduite de la différence de potentiel $\mathbf{V_a - 0}$.

(3) : Les données transitent sur une **paire de fils** distincts de la masse, de façon **différentielle** : On ne lit plus $\mathbf{V_a - 0}$ mais $\mathbf{V_a - V_b}$.

⇒ **Insensibilité aux bruits environnants** car les tensions parasites $\mathbf{V_p}$ s'annulent dans le calcul de la différence de potentiel.

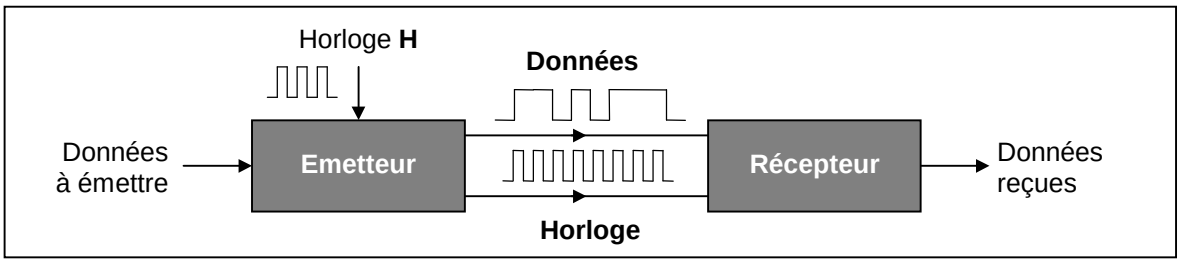
⇒ **Débits plus importants** que la structure asymétrique. Un seul fil de masse est utilisé comme pour la structure asymétrique.



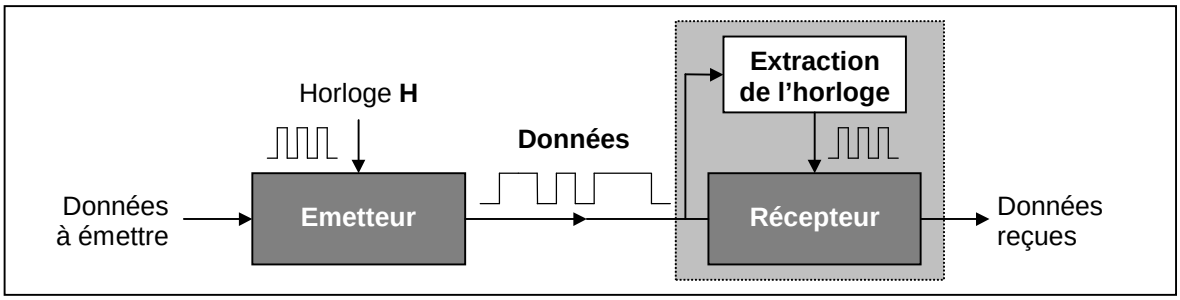
3.2. Transmission synchrone

En transmission synchrone, les données sont émises de façon **régulière**, sans séparation entre les différentes données. Pour cela, un signal d'horloge de période **T** fonctionne pendant toute la durée de l'émission. Pour assurer la synchronisation, le récepteur doit reconstituer, à un décalage près, le rythme du signal d'horloge qui a servi à l'émission. On dispose de deux moyens pour y parvenir :

- **Transporter** le signal d'horloge sur un support séparé reliant l'émetteur et le récepteur. Cette technique est utilisée sur des courtes distances (liaison USB).



- **Reconstituer** le signal d'horloge à partir des **changements d'état** du signal reçu.



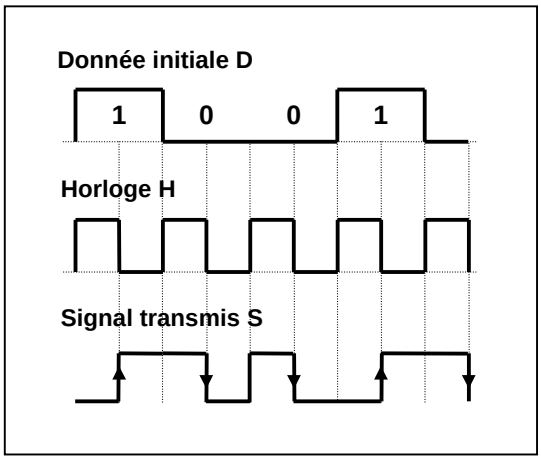
Les transitions du signal de données doivent être suffisamment nombreuses pour éviter la perte de la synchronisation. Cette deuxième alternative est extrêmement utilisée car elle permet de réaliser des transmissions à **très haut débit**.

Pour mélanger l'horloge avec les données, on utilise des **transcodeurs** qui élaborent, de manière logique, un signal à partir de l'horloge et de la donnée à transmettre. Un des codes les plus utilisés est le **code Manchester**.

On code le bit **« 1 »** par une transition du niveau bas au niveau haut et le bit **« 0 »** par la transition inverse (mais cela peut être l'inverse – cf. page 325).

Le signal transmis **S** est issu du **OU** - **EXCLUSIF** entre l'horloge **H** et la donnée initiale **D** :

$$S = D \oplus H = D\overline{H} + \overline{D}H$$



Ce système permet de faire apparaître sur la donnée envoyée des fronts montants ↑ ou descendants ↓ pour toutes les périodes d'horloge. Du côté réception, il alors facile d'élaborer une horloge de réception en se **synchronisant sur la détection des fronts**.

Cette synchronisation qualifiée de « **synchronisation bit** », doit être constante, c'est à dire aussi bien lors des périodes d'émission que pendant les moments de « silence ». Elle implique donc un autre niveau de synchronisation pour déterminer le début et la fin d'une trame.

C'est le rôle du **code de synchronisation** (rôle joué par exemple par le champ « Préambule » dans les trames Ethernet) situé au début des trames transmises et présentant des successions de « 0 » et de « 1 », afin de recaler l'oscillateur local avant chaque message.

❑ **Structure générale d'une trame synchrone**

Code de synchronisation	*Champ de service*	*Champ de données*	*Champ de contrôle*

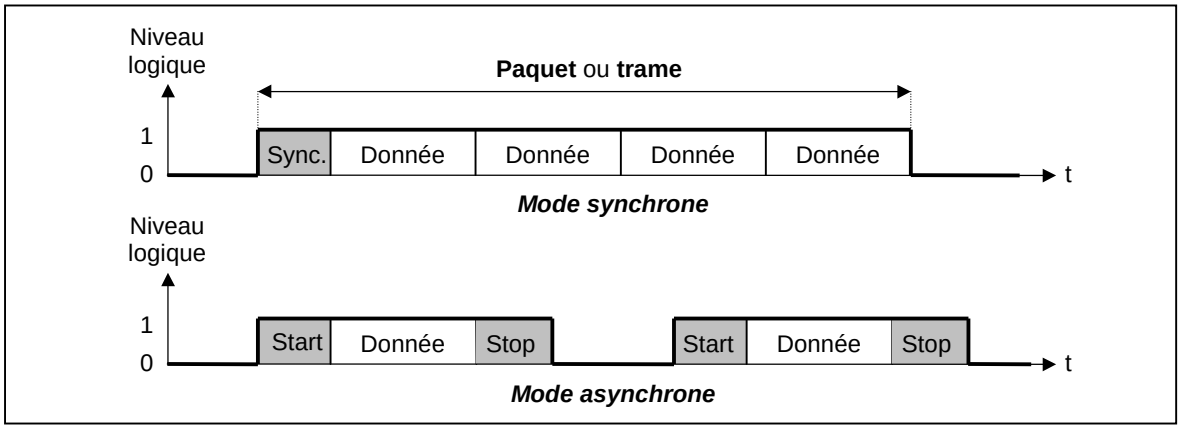
Suivent ensuite un champ de service pouvant contenir l'adresse de l'émetteur et du récepteur ou d'autres informations sur le type de trame ou la structure du message (début de fichier, début ou longueur de bloc, …), un champ de données correspondant au message et un champ de contrôle permettant la détection des erreurs de transmission.

Le message transmis peut représenter **quelques milliers d'octets** se suivant sans temps mort.

Bien que le message soit techniquement transmis de manière synchrone (synchronisation bit), l'intervalle de temps entre deux messages ne donne pas à lieu à synchronisation.

On parle parfois de **transmission asynchrone - synchronisée**.

3.3. Comparaison entre transmissions synchrone et asynchrone



On suppose qu'on a un bloc de données de **1500 octets** à transmettre.

L'efficacité d'un mode de transmission est mesurée par le nombre de bits utiles transmis sur le nombre de bits réellement émis. L'efficacité notée $\mathbf{E_{ff}}$ est donnée par la relation suivante :

$$\mathbf{E_{ff} = \frac{Nombre\ de\ bits\ de\ données}{Nombre\ de\ bits\ transmis}}$$

➢ Dans le **mode synchrone**, en raisonnant sur la trame suivante :

Code de synchronisation	*Adresse de destination*	*Adresse de source*	*Type*	*Données applicatives*	*Code de contrôle*
8 octets	**6 octets**	**6 octets**	**2 octets**	**1500 octets**	**4 octets**

$$E_{ff} \approx \frac{1500 \cdot 8}{(8+6+6+2+1500+4) \cdot 8} = \boxed{\mathbf{98{,}3\,\%}}$$

➢ Dans le **mode asynchrone**, il faut ajouter à chaque octet de données **1 bit** de **Start**, **2 bits** de **Stop** (par exemple) et **1 bit** de **parité**, soit **12 bits** pour **8 bits utiles**. Dans ces conditions :

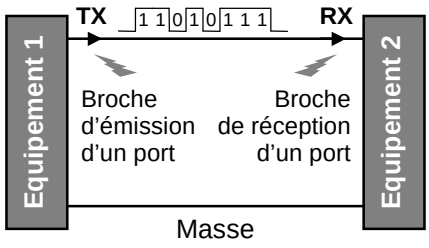
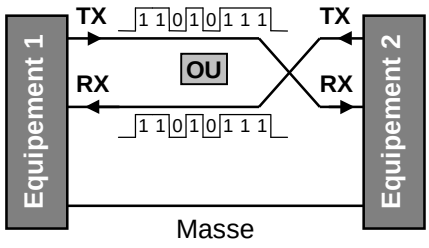
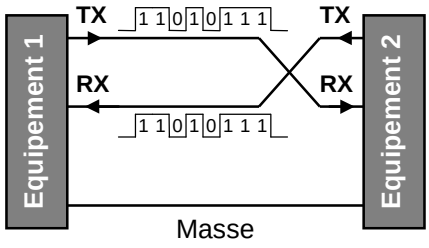
$$E_{ff} = \frac{1500 \cdot 8}{1500 \cdot 12} = \boxed{\mathbf{67\,\%}}$$

❑ **Conclusion**

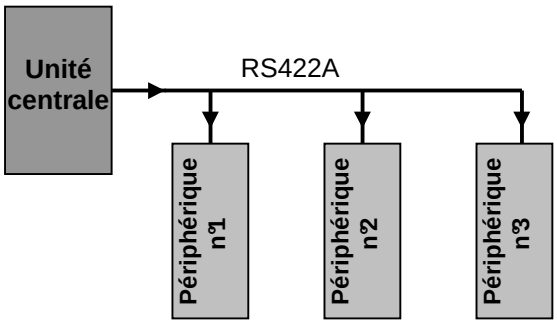
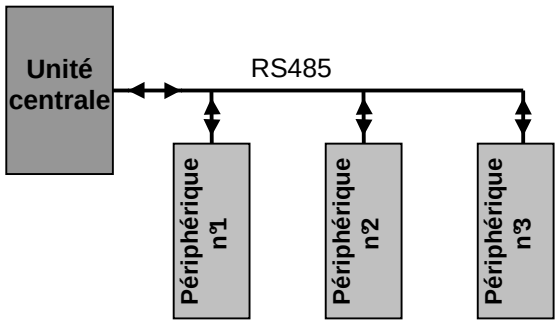
La redondance due aux bits de **Start**, de **Stop** et de **parité** ajoutés à chaque octet dans la transmission asynchrone, ne permet pas d'atteindre des débits élevés (plus de 25% du temps est perdu). De ce fait, son utilisation est limitée aux systèmes de transmission bas débit (< 200 kbits/s). Par contre, le mode synchrone permet des débits plus importants. Les réseaux informatiques dépendent principalement de la **transmission série synchrone**.

4. Sens des transmissions

Lors d'une transmission entre deux points, il faut traiter un dialogue et non un monologue. Il faut donc une convention pour fixer le sens de la transmission. On rencontrera 3 cas :

Liaison SIMPLEX Cette liaison **unidirectionnelle** n'autorise le passage de données que dans un seul sens. L'un des équipements est émetteur tandis que l'autre est récepteur.	
Liaison HALF DUPLEX Cette liaison est **bidirectionnelle** mais les équipements ne peuvent émettre simultanément. A chaque instant, le rôle d'émetteur revient à un équipement et celui de récepteur à l'autre.	
Liaison FULL DUPLEX Cette liaison **bidirectionnelle** requiert un fil pour l'émission et un pour la réception des données. La communication entre équipements peut se faire **simultanément**.	

❑ **Exemples** : Transmission série asynchrone

Liaison RS422A	**Liaison RS485**
	
Simplex Liaison **multipoint** Liaison **symétrique** (différentielle)	**Half duplex** ou **Full duplex** (ligne doublée) Liaison **multipoint** Liaison **symétrique** (différentielle)