Influence d'une architecture de type maître-esclave dans les problématiques de sécurité de l'Internet des Objets

Philippe Pittoli

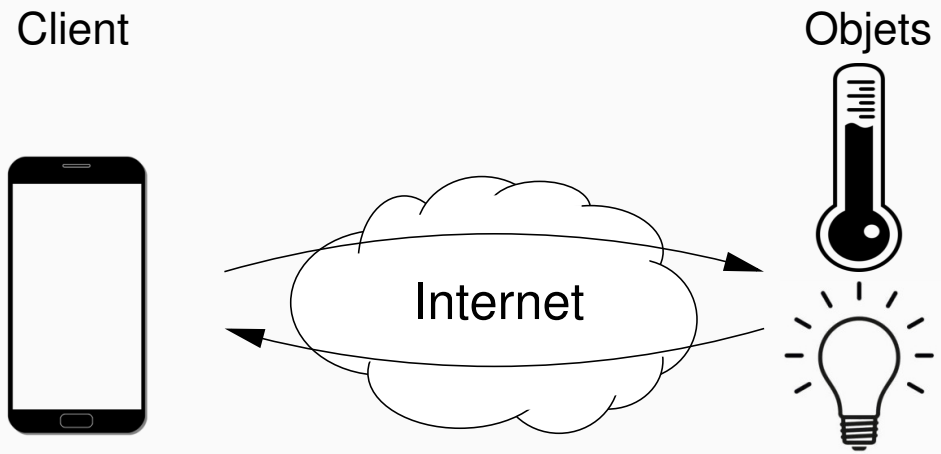
Encadrants	Thomas Noël, Pierre David
Rapportrices	Isabelle Chrisment, Nathalie Mitton
Examinateurs	André-Luc Beylot, Frank Rousseau

20 mai 2019

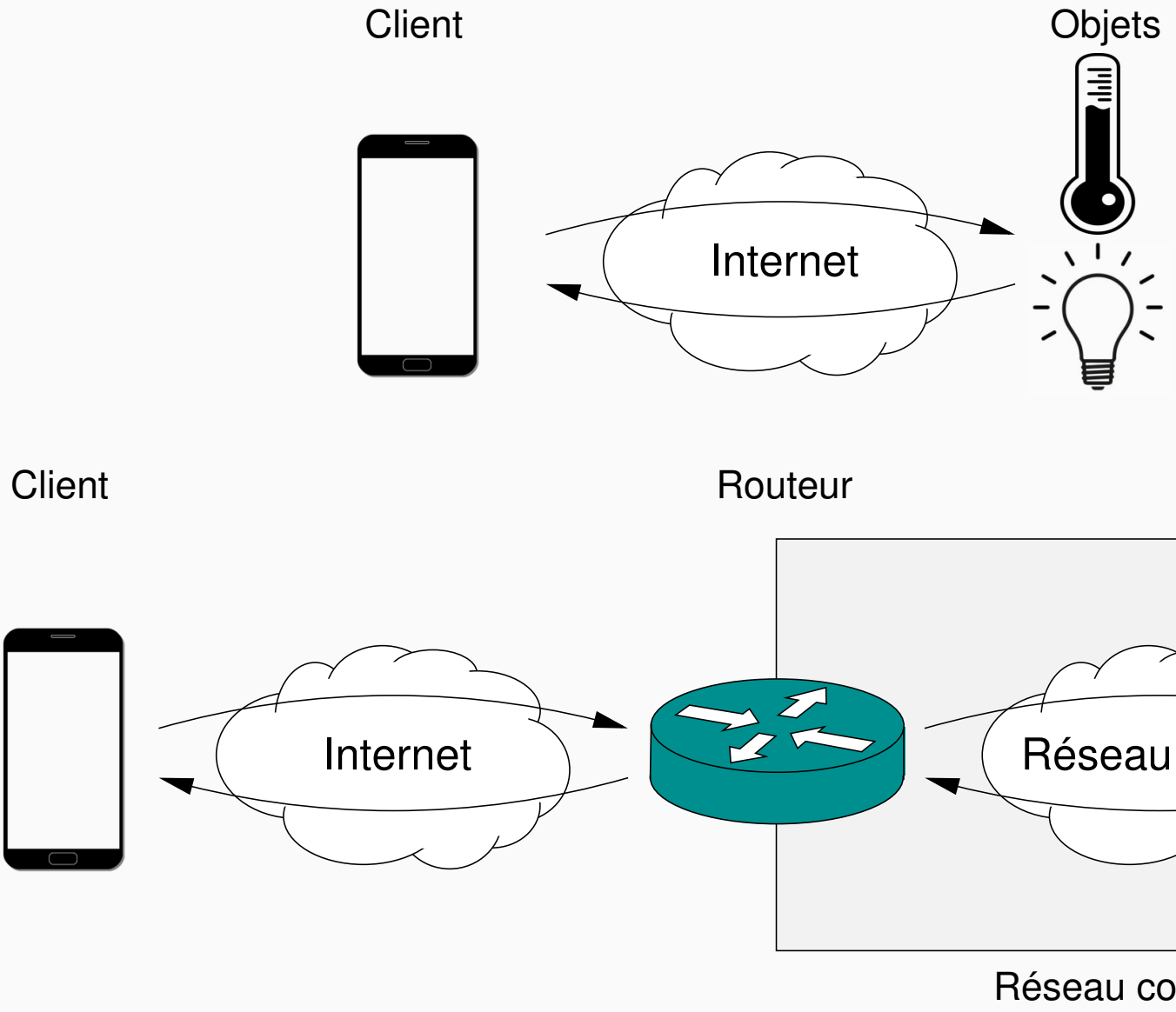
Plan

1. **Introduction de la problématique**
2. Identifier puis comparer les architectures de sécurité
3. Comprendre les différences entre les types d'architectures
4. Mesurer les différences
5. Conclusion

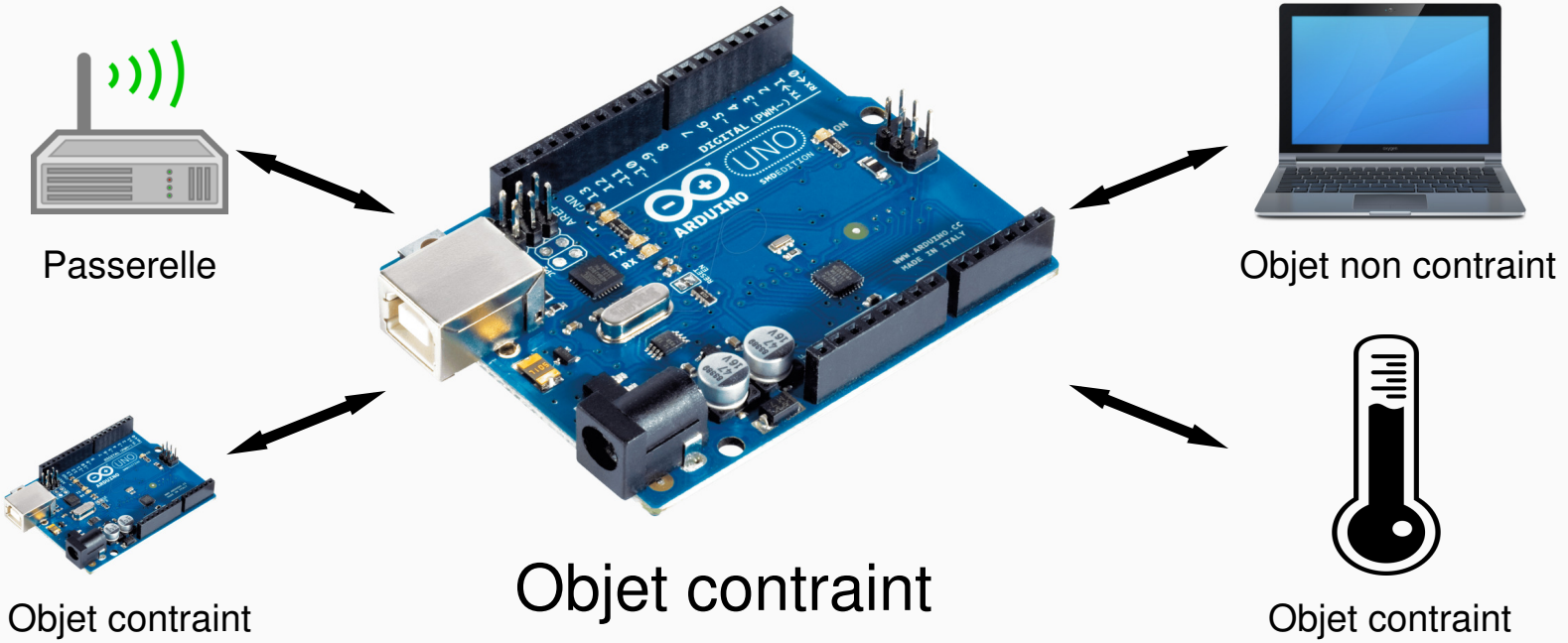
Internet des Objets



Internet des Objets



Capteurs et actionneurs



16 MHz, 16 KB RAM, 128 KB Flash

Communication sécurisée avec des objets très contraints

Contraintes sur le réseau

Cible : IEEE 802.15.4

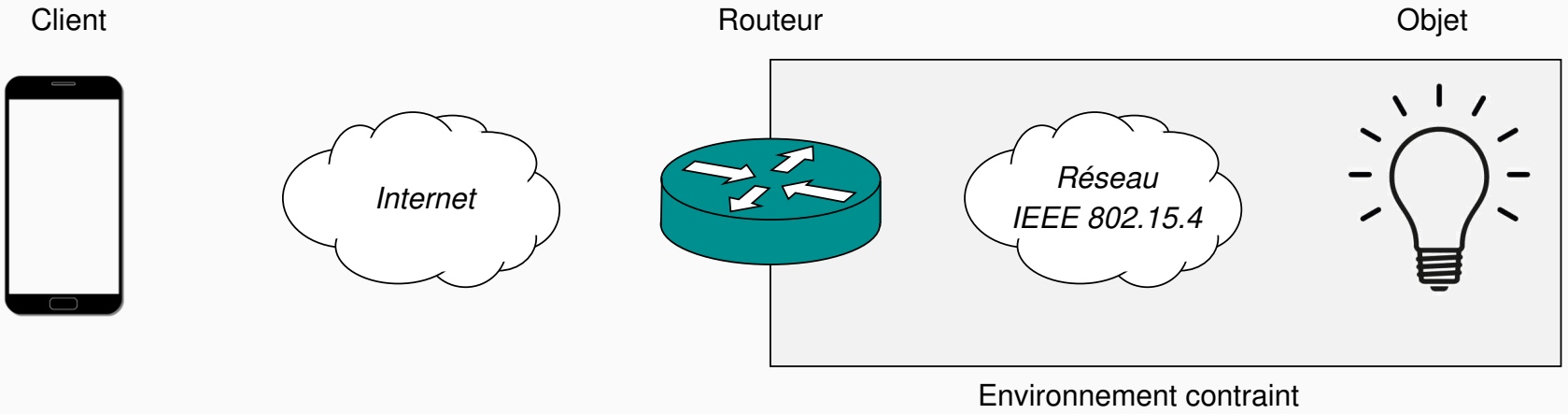
- standard ouvert
- adapté aux capteurs et actionneurs
    - courts temps d'accès (par ex. pour ouvrir une porte)
- seulement couches physique et liaison : échanger, adresser
- a déjà fait l'objet de recherches académiques
- fortement déployé : 500 millions de puces vendues

Contrainte : charge utile 127 octets

Énergie : pas toujours une contrainte (exemple : industrie 4.0)

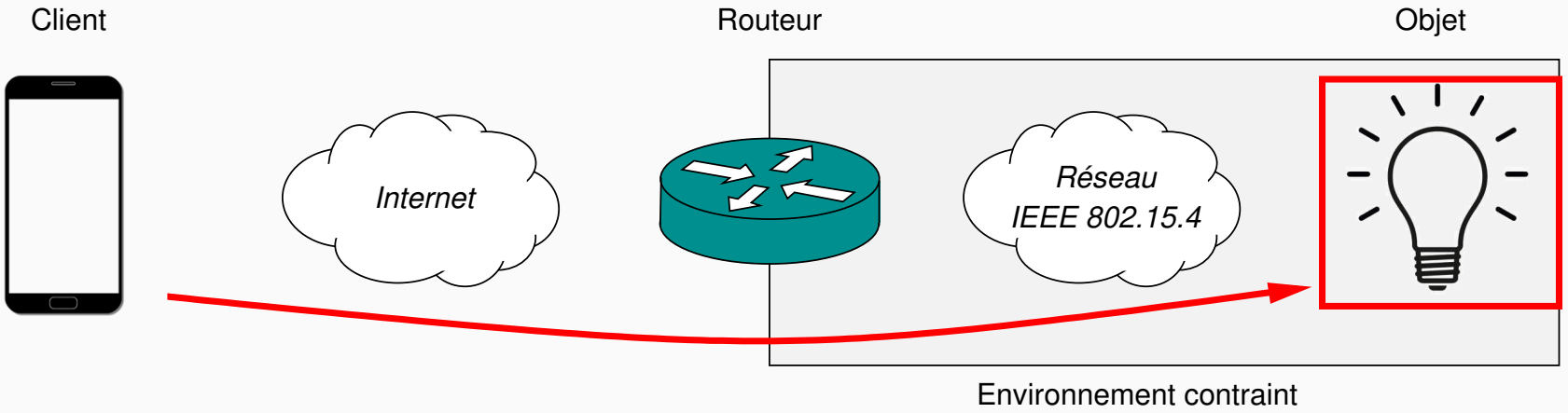
Problématique de sécurité

Architecture de référence



Problématique de sécurité

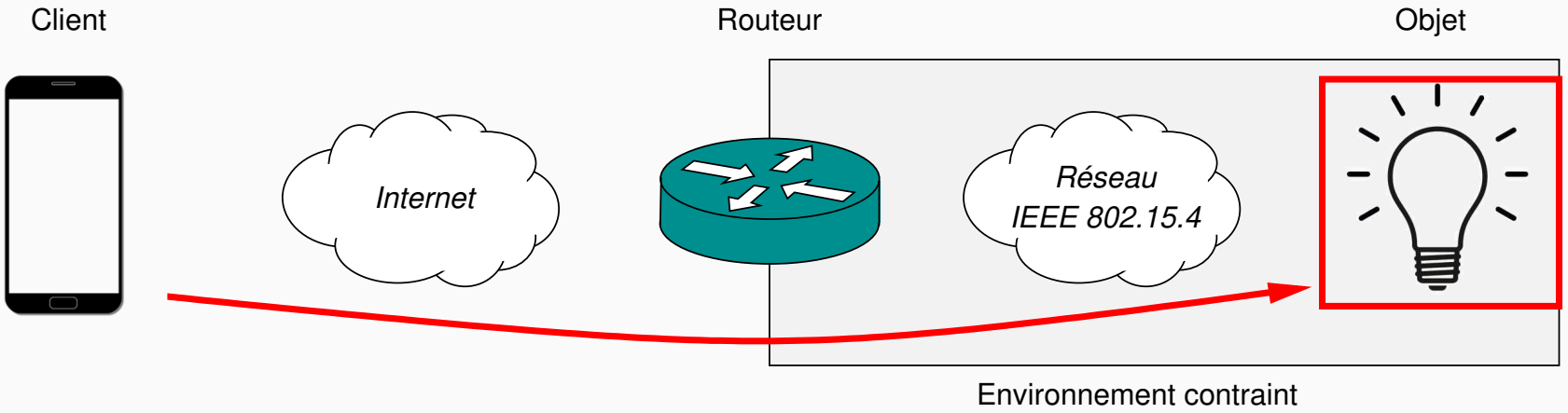
Architecture de référence



Point de terminaison

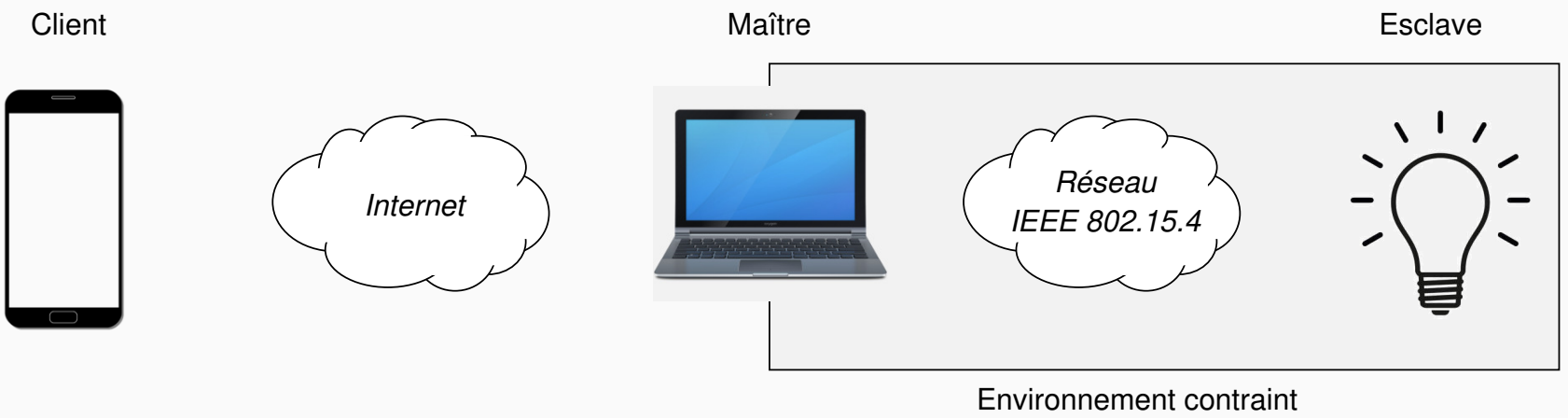
Problématique de sécurité

Architecture de référence



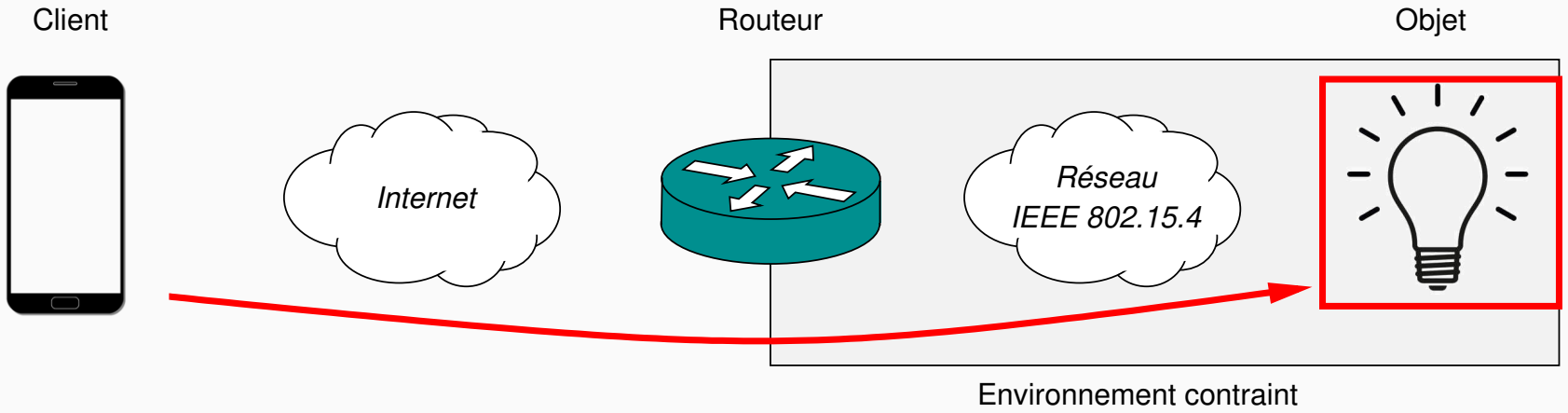
Point de terminaison

Architecture Maître–Esclave



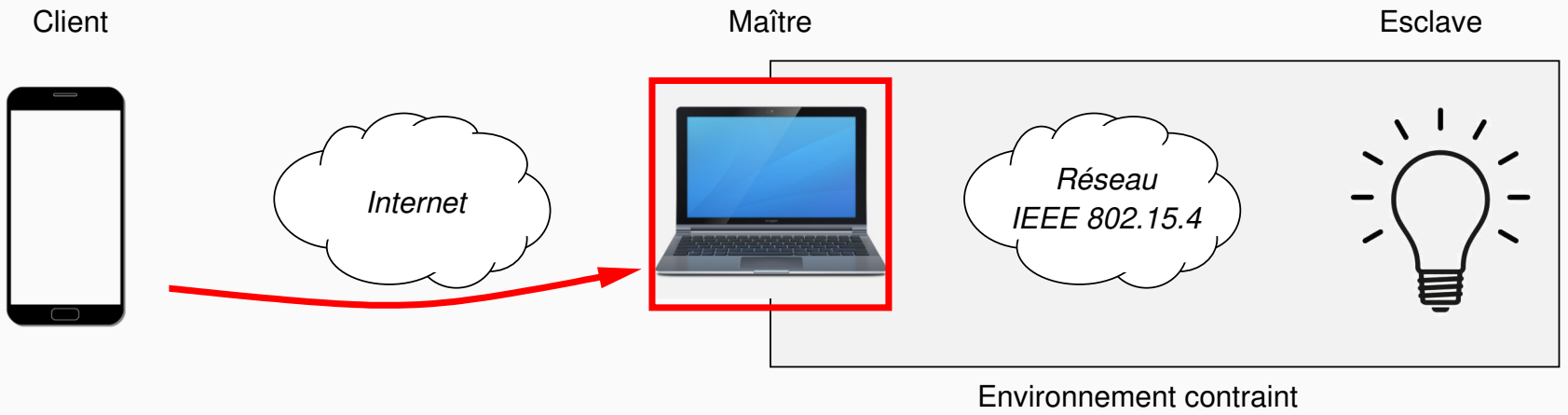
Problématique de sécurité

Architecture de référence



Point de terminaison

Architecture Maître–Esclave



Point de terminaison

Plan

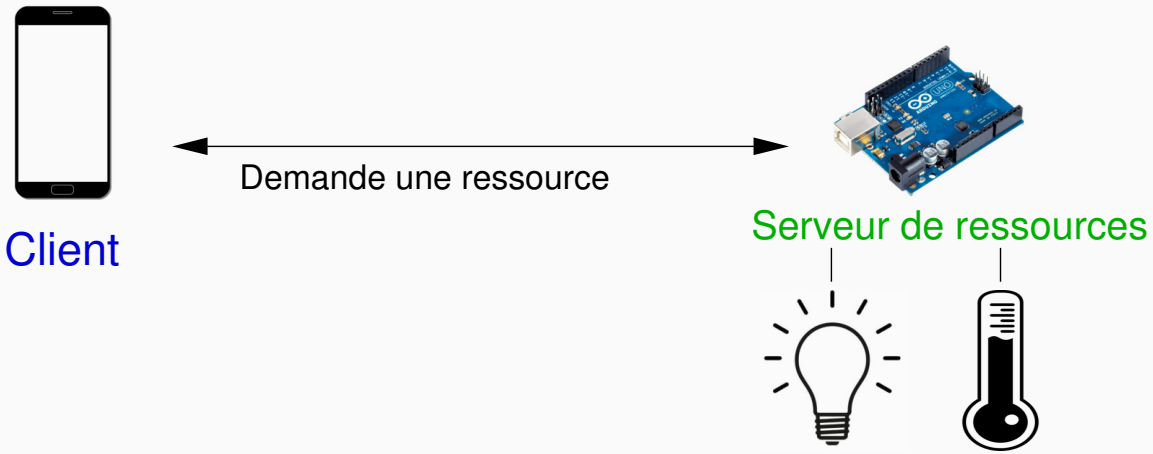
1. Introduction de la problématique
2. **Identifier puis comparer les architectures**
3. Comprendre les différences entre les types d'architectures
4. Mesurer les différences
5. Conclusion

Architecture de sécurité ?

À notre connaissance, pas de définition formelle de ce qu'est une architecture de sécurité

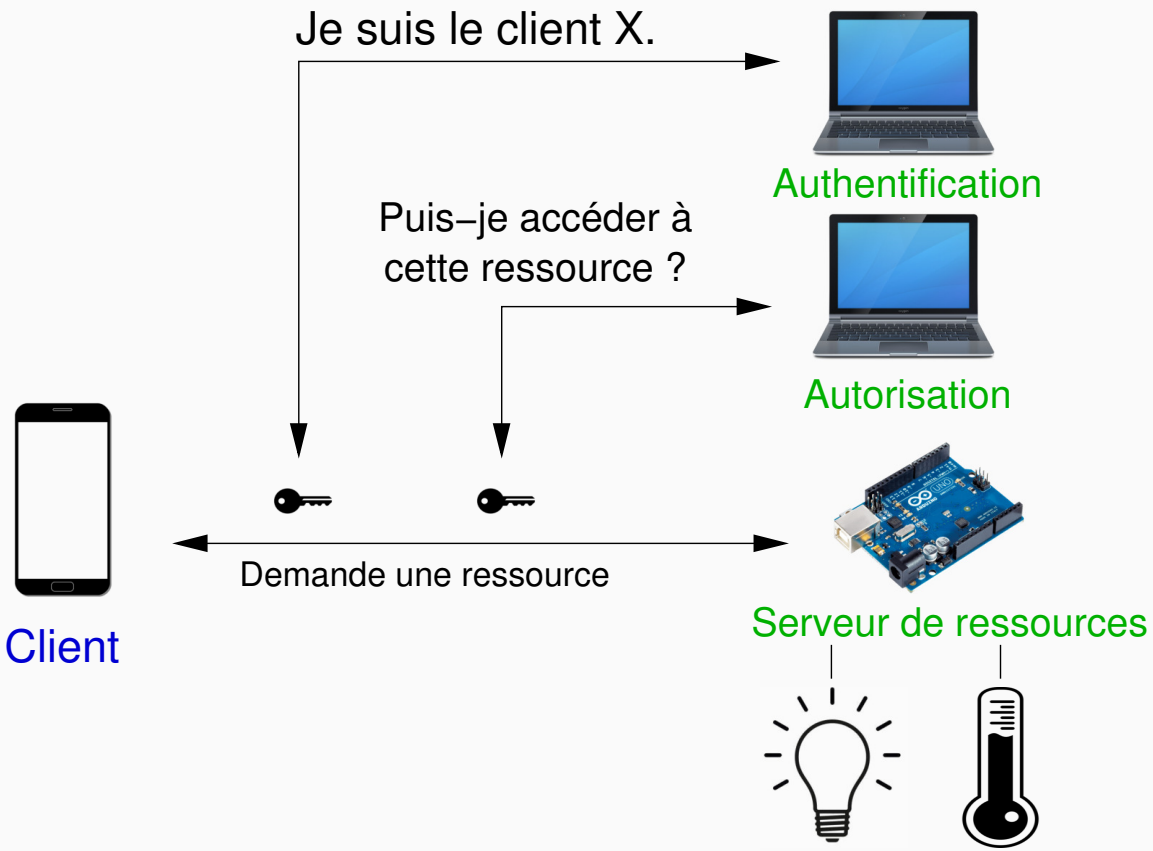
1. Définir formellement ce qu'est une architecture
2. Cataloguer les architectures
3. Définir des scénarios
4. Déterminer des points de comparaison

Étape 1 : définir une architecture de sécurité



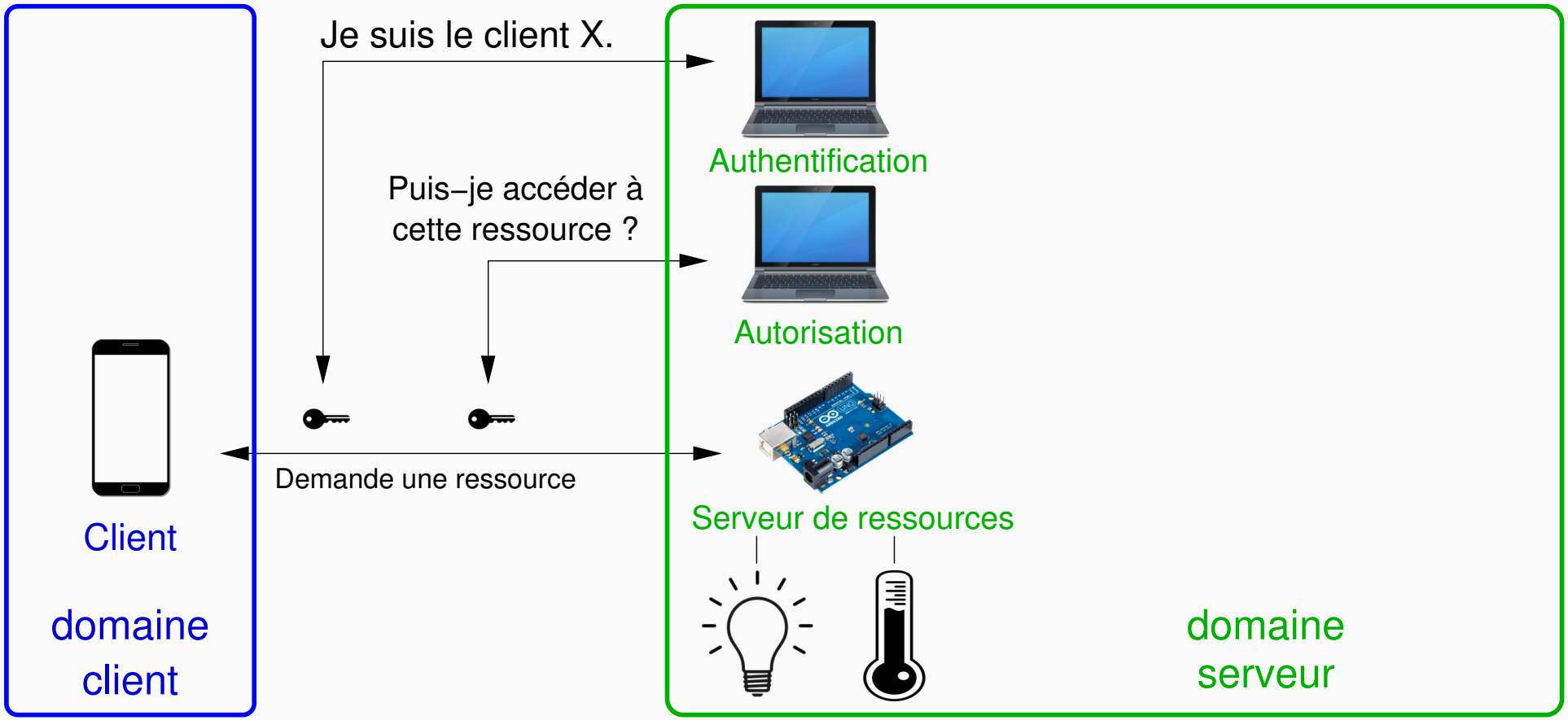
- **client** et **serveur de ressources**

Étape 1 : définir une architecture de sécurité



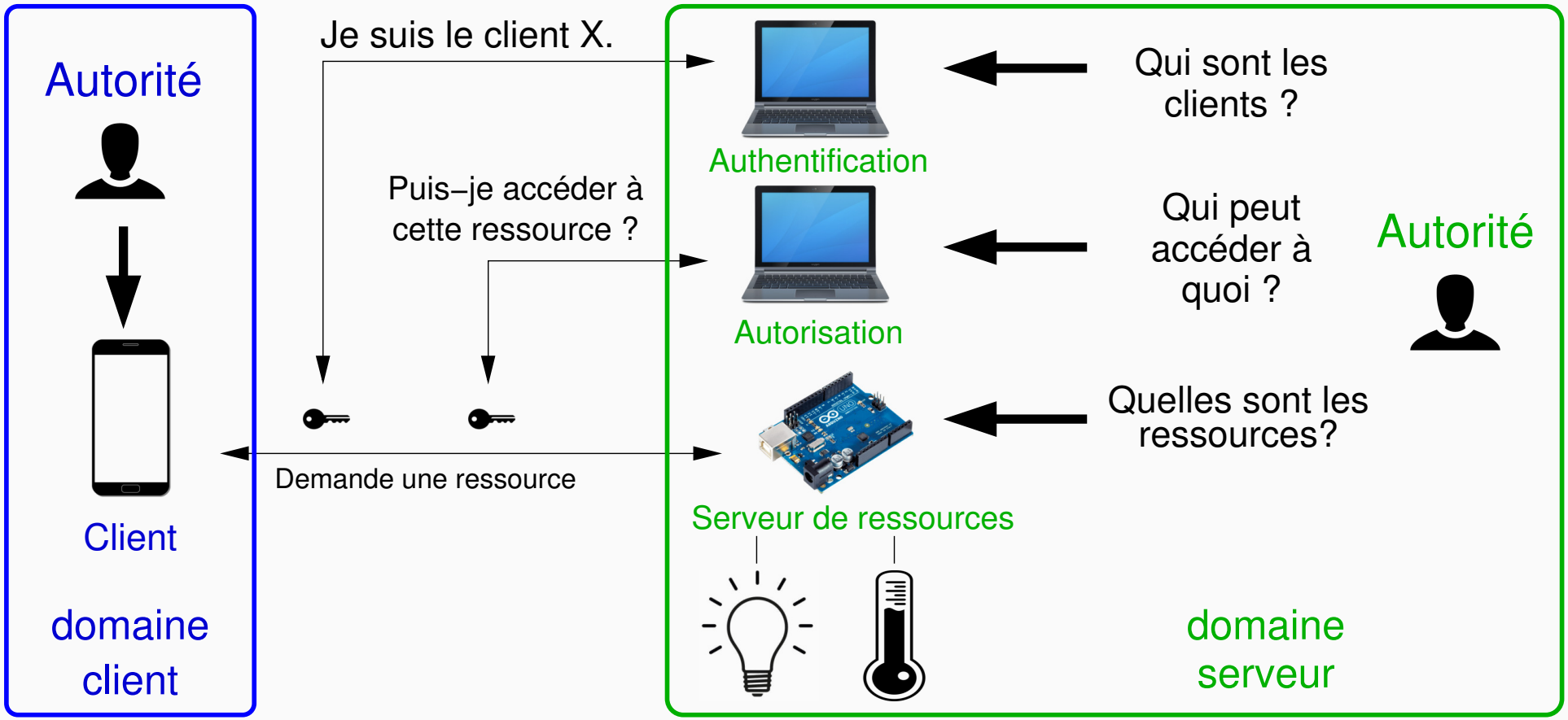
- **client** et **serveur de ressources**
- **authentification** et **autorisation**

Étape 1 : définir une architecture de sécurité



- **client** et **serveur de ressources**
- **authentification** et **autorisation**
- **domaines**

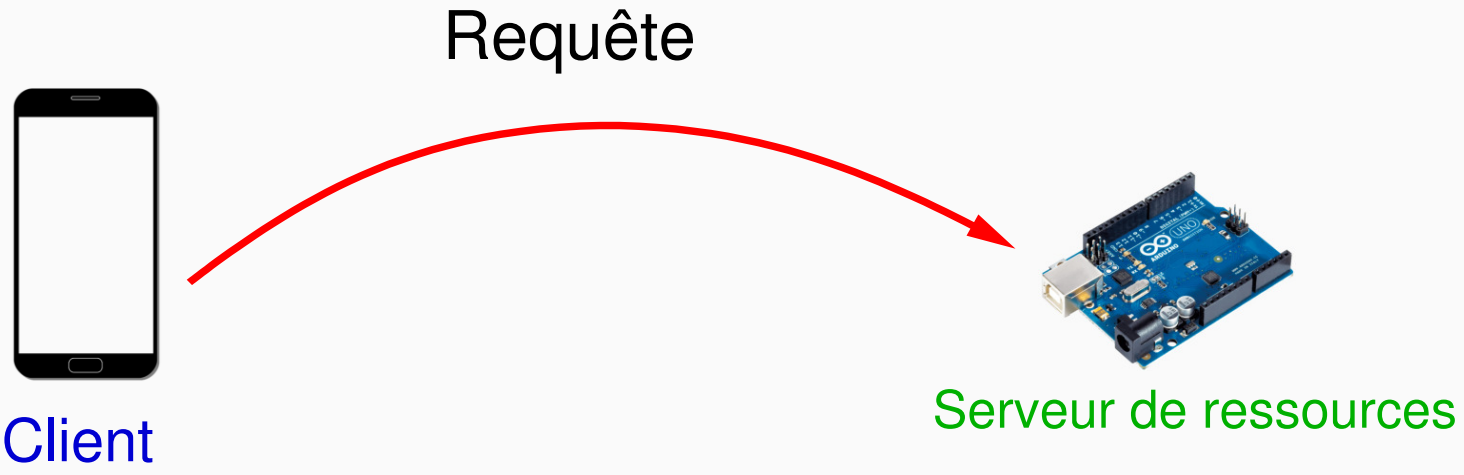
Étape 1 : définir une architecture de sécurité



- **client** et **serveur de ressources**
- **authentification** et **autorisation**
- **domaines**
- **autorités**

Étape 1 : définir une architecture de sécurité

Relations : interactions entre les entités



Exemple :

Entités :	Client $\rightarrow$ Serveur de ressources
Protocoles :	DTLS, CoAP
Propriétés :	chiffrement, intégrité, authentification
Informations :	Requête

Relation = entités, protocoles, propriétés, informations

Étape 1 : définir une architecture de sécurité

Formalisme des relations

$$\begin{aligned} R = \quad & \{< e_1 \ \mathbf{op} \ e_2, \mathbf{p}, \mathbf{o}, \mathbf{i} > | e_1, e_2 \in E, \\ & \mathbf{op} \in \{\rightarrow, \leftrightarrow\}, \mathbf{p} \subseteq P, \mathbf{o} \subseteq O, \mathbf{i} \subseteq I\} \end{aligned}$$

- op = communication directionnelle / bidirectionnelle
- P = protocoles employés
- O = propriétés de la communication (fiabilité, ordre, confidentialité. . .)
- I = informations partagées

Formalisme d'une architecture

<E, A, D, R >

- E = entités (serveur de ressources, client. . .)
- A = autorités (exemple : l'administrateur de l'infrastructure)
- D = domaines, $\{e$ contrôlée par $a | e \in E, a \in A\}$
- R = relations entre les entités

Étape 1 : définir une architecture de sécurité

Une architecture de sécurité est un ensemble d'**entités**, contrôlées par des **autorités** (personnes ou organisations) qui définissent des **domaines de confiance**

Une interaction entre entités est nommée **relation** : elle indique la direction du message, les informations, les protocoles et les propriétés de l'échange

Étape 2 : cataloguer les architectures

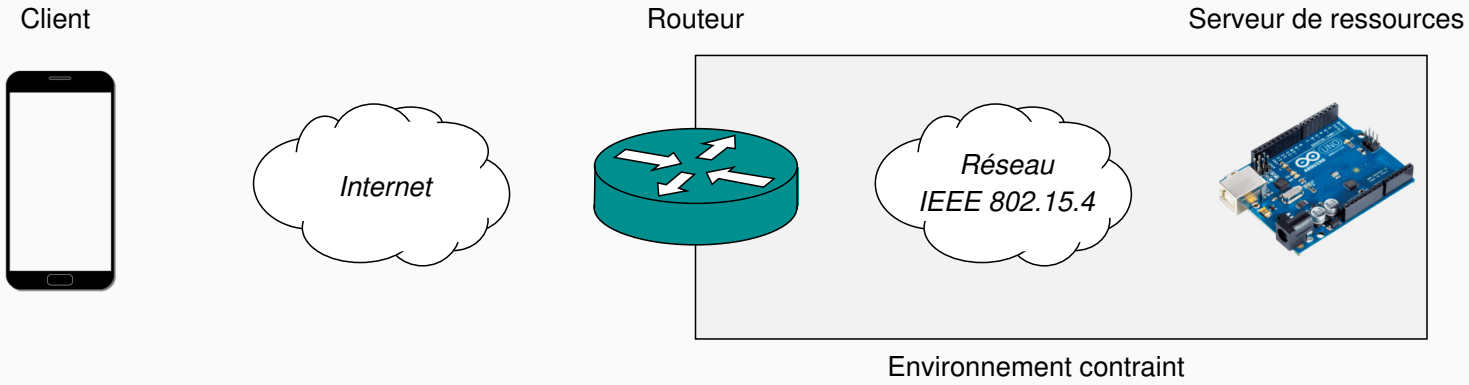
Ces architectures diffèrent sur

- objectifs
- entités (ex. : avec et sans cache, proxy, . . .)
- relations
- fonctionnalités

Étape 3 : définir des scénarios pour la comparaison

Nous voulons couvrir entre autres les méthodes courantes d'**authentification** et de **partage de clés**

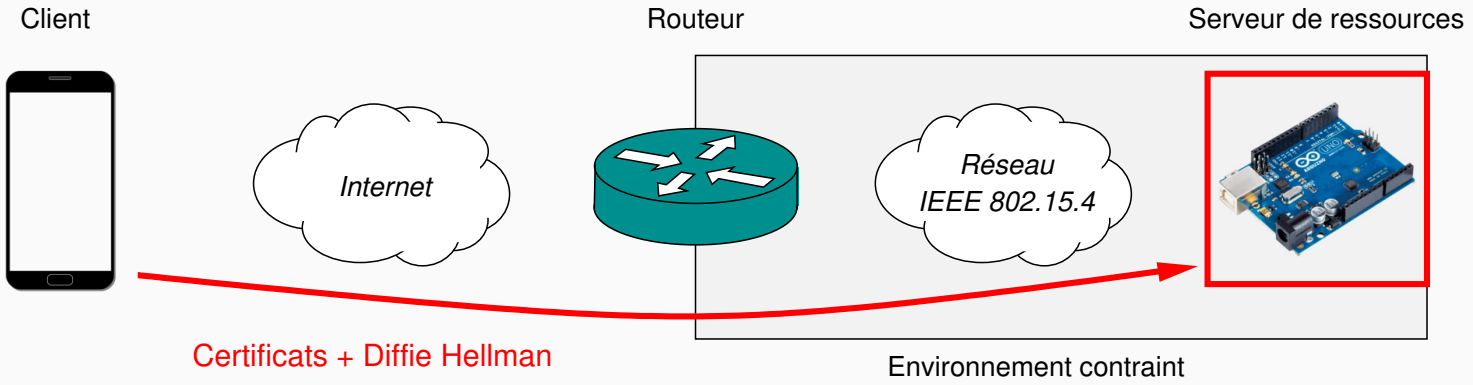
Client externe



Étape 3 : définir des scénarios pour la comparaison

Nous voulons couvrir entre autres les méthodes courantes d'**authentification** et de **partage de clés**

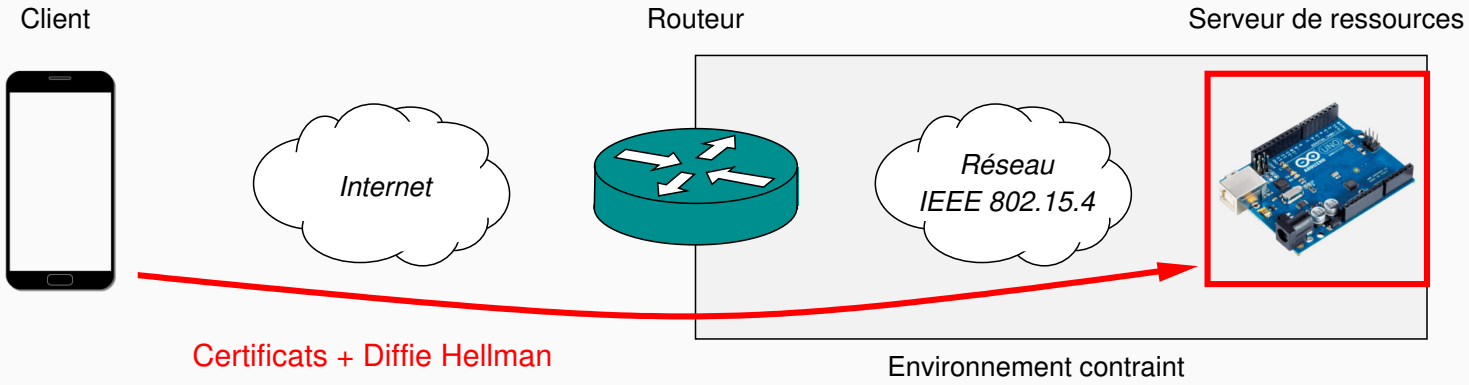
Client externe



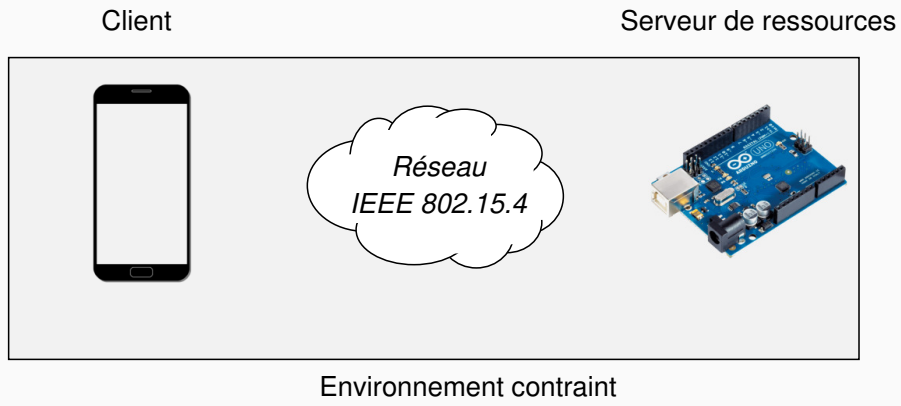
Étape 3 : définir des scénarios pour la comparaison

Nous voulons couvrir entre autres les méthodes courantes d'**authentification** et de **partage de clés**

Client externe



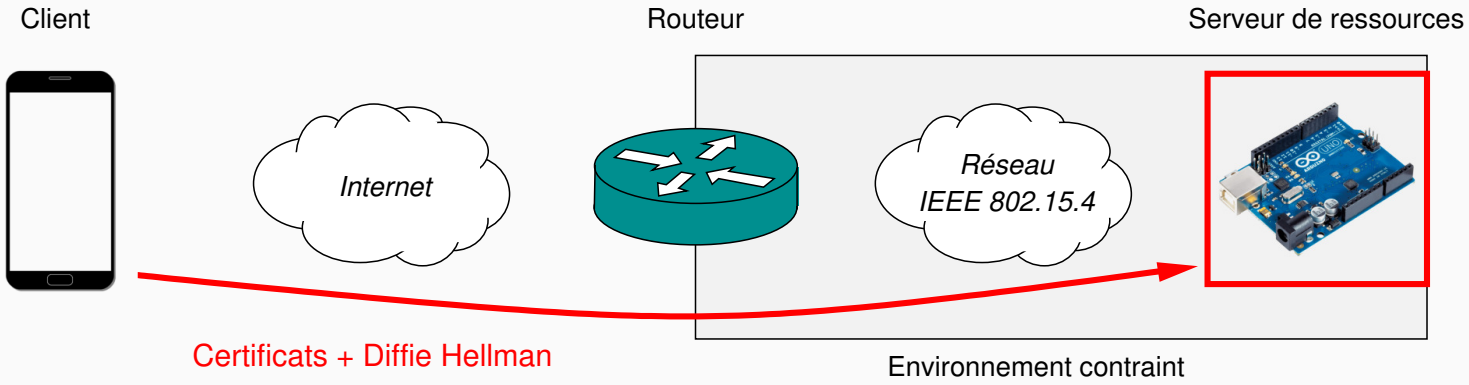
Client interne



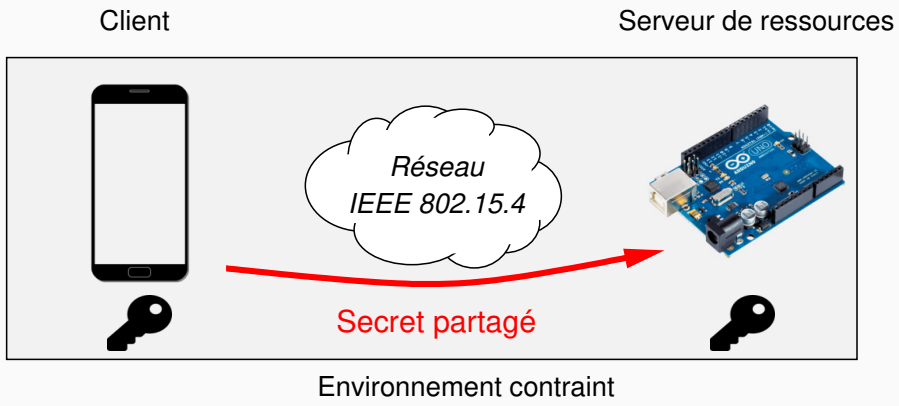
Étape 3 : définir des scénarios pour la comparaison

Nous voulons couvrir entre autres les méthodes courantes d'**authentification** et de **partage de clés**

Client externe



Client interne



Étape 4 : déterminer des points de comparaison

Exemples de critères :

Architecture

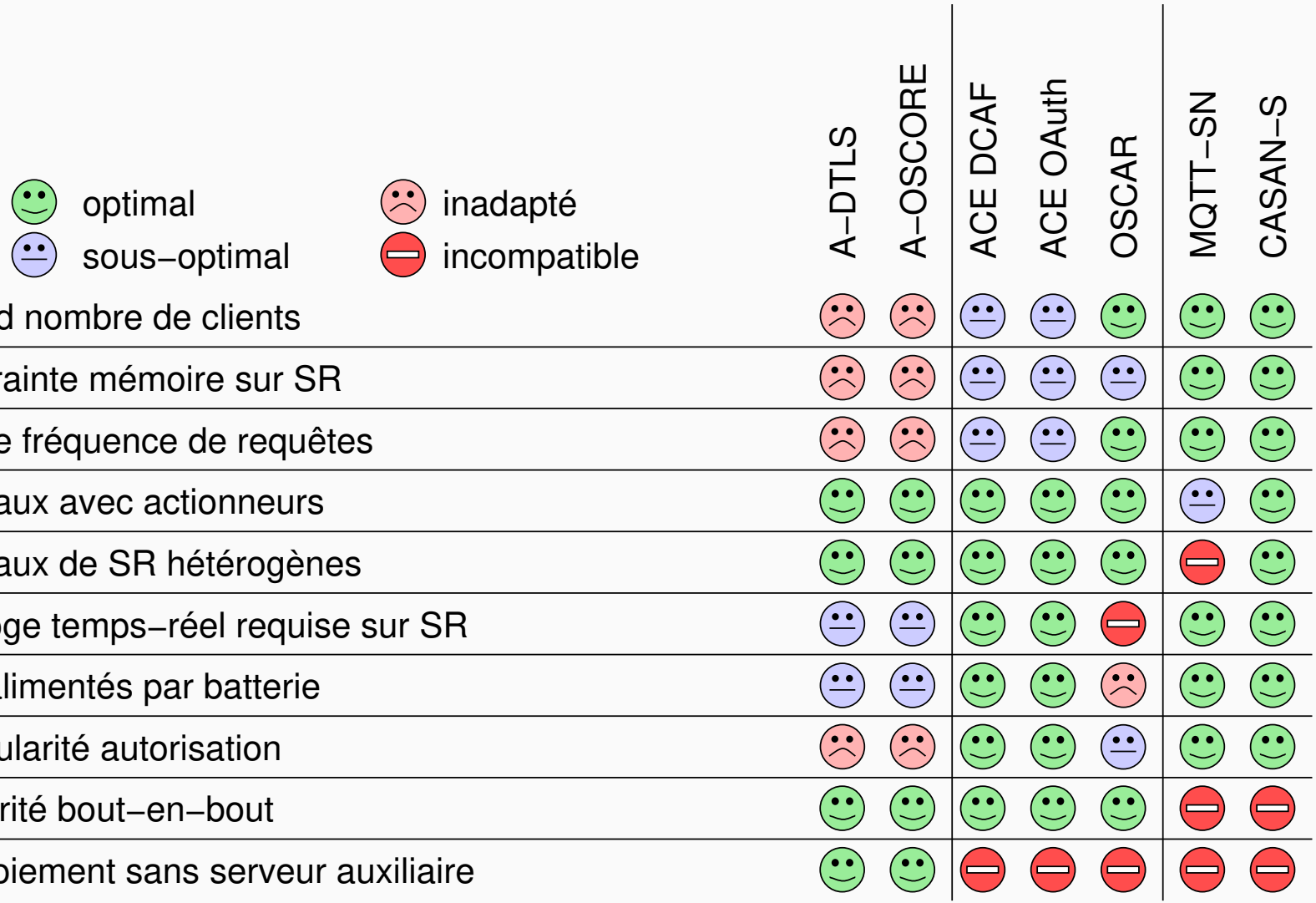
- contexte par client sur SR
- cryptographie asymétrique obligatoire
- granularité de l'autorisation
- surcharge par message

Sécurité des communications

- protection contre le déni de service
- protection contre le rejeu
- protection données de connexion
- agilité cryptographique

Comparaison : 10 architectures, 2 scénarios, 22 critères

Recommandations suite à la comparaison



Pas de solution couvrant toutes les contraintes

Recommandations suite à la comparaison

architectures de référence sans serveur auxiliaire

optimal
sous–optimal
inadapté
incompatible

	A–DTLS	A–OSCORE	ACE DCAF	ACE OAuth	OSCAR	MQTT–SN	CASAN–S
grand nombre de clients	☹	☹	😐	😐	🙂	🙂	🙂
contrainte mémoire sur SR	☹	☹	😐	😐	😐	🙂	🙂
haute fréquence de requêtes	☹	☹	😐	😐	🙂	🙂	🙂
réseaux avec actionneurs	🙂	🙂	🙂	🙂	🙂	😐	🙂
réseaux de SR hétérogènes	🙂	🙂	🙂	🙂	🙂	⛔	🙂
horloge temps–réel requise sur SR	😐	😐	🙂	🙂	⛔	🙂	🙂
SR alimentés par batterie	😐	😐	🙂	🙂	☹	🙂	🙂
granularité autorisation	☹	☹	🙂	🙂	😐	🙂	🙂
sécurité bout–en–bout	🙂	🙂	🙂	🙂	🙂	⛔	⛔
déploiement sans serveur auxiliaire	🙂	🙂	⛔	⛔	⛔	⛔	⛔

Pas de solution couvrant toutes les contraintes

Recommandations suite à la comparaison

architectures de référence sans serveur auxiliaire

architectures de référence avec serveur auxiliaire

optimal — sous–optimal — inadapté — incompatible

	A–DTLS	A–OSCORE	ACE DCAF	ACE OAuth	OSCAR	MQTT–SN	CASAN–S
grand nombre de clients	☹	☹	😐	😐	🙂	🙂	🙂
contrainte mémoire sur SR	☹	☹	😐	😐	😐	🙂	🙂
haute fréquence de requêtes	☹	☹	😐	😐	🙂	🙂	🙂
réseaux avec actionneurs	🙂	🙂	🙂	🙂	🙂	😐	🙂
réseaux de SR hétérogènes	🙂	🙂	🙂	🙂	🙂	⛔	🙂
horloge temps–réel requise sur SR	😐	😐	🙂	🙂	⛔	🙂	🙂
SR alimentés par batterie	😐	😐	🙂	🙂	☹	🙂	🙂
granularité autorisation	☹	☹	🙂	🙂	😐	🙂	🙂
sécurité bout–en–bout	🙂	🙂	🙂	🙂	🙂	⛔	⛔
déploiement sans serveur auxiliaire	🙂	🙂	⛔	⛔	⛔	⛔	⛔

Pas de solution couvrant toutes les contraintes

Recommandations suite à la comparaison

architectures de référence sans serveur auxiliaire

architectures de référence avec serveur auxiliaire

architectures maître–esclave

optimal — sous–optimal — inadapté — incompatible

	A–DTLS	A–OSCORE	ACE DCAF	ACE OAuth	OSCAR	MQTT–SN	CASAN–S
grand nombre de clients	☹	☹	😐	😐	☺	☺	☺
contrainte mémoire sur SR	☹	☹	😐	😐	😐	☺	☺
haute fréquence de requêtes	☹	☹	😐	😐	☺	☺	☺
réseaux avec actionneurs	☺	☺	☺	☺	☺	😐	☺
réseaux de SR hétérogènes	☺	☺	☺	☺	☺	⛔	☺
horloge temps–réel requise sur SR	😐	😐	☺	☺	⛔	☺	☺
SR alimentés par batterie	😐	😐	☺	☺	☹	☺	☺
granularité autorisation	☹	☹	☺	☺	😐	☺	☺
sécurité bout–en–bout	☺	☺	☺	☺	☺	⛔	⛔
déploiement sans serveur auxiliaire	☺	☺	⛔	⛔	⛔	⛔	⛔

Pas de solution couvrant toutes les contraintes

Plan

1. Introduction de la problématique
2. Identifier puis comparer les architectures
3. **Comprendre les différences entre les types d'architectures**
4. Mesurer les différences
5. Conclusion

Comprendre les différences

	Architecture de référence	**Architecture maître-esclave**
algorithmes d'authentification	limités clé préconfigurée	aucune limitation TLS et gestion de certificats
mémoire nécessaire sur SR	liée au nb clients	sans lien avec les clients
connexion client - SR	à chaque client	aucune
authentification et autorisation	sur chaque nœud	centralisée(s)
découverte de ressources	non spécifié	au démarrage du nœud
cache de données	impossible	intégré
taille des messages	routeur 6LoWPAN	minimale

Pistes d'amélioration d'une architecture de référence

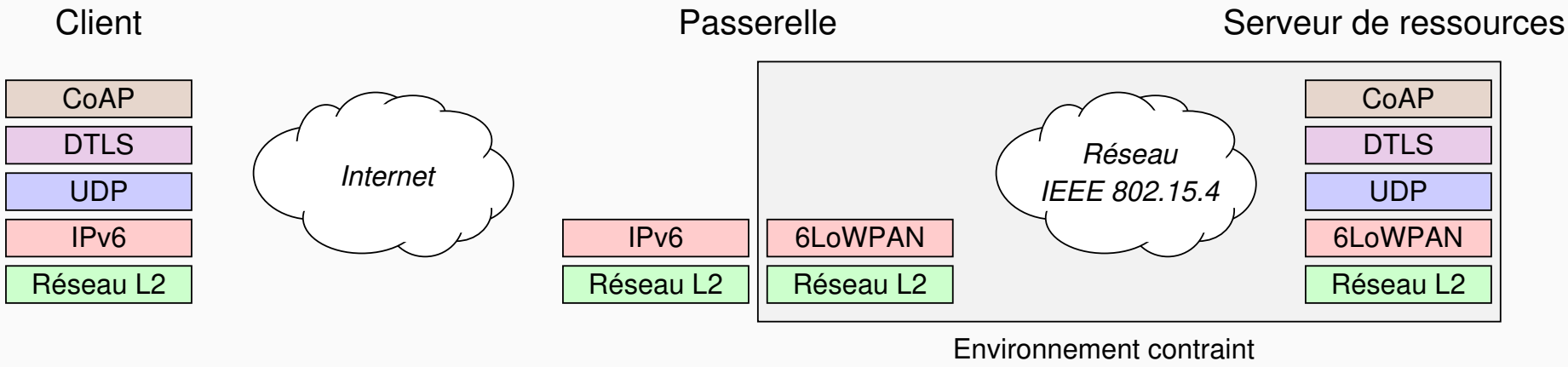
Problème	Piste d'amélioration
algorithmes limités	serveur d'authentification
mémoire sur SR	serveur d'authentification
chaque client se connecte au SR	serveur d'authentification
auth. + autorisation sur SR	serveur centralisant l'authentification et l'autorisation
pas de découverte de ressources	serveur de catalogue de ressources
cache de données	serveur de cache
taille des messages	serveur initiant les requêtes côté réseau contraint

La délégation des tâches à un matériel dédié est une piste récurrente

Plan

1. Introduction de la problématique
2. Identifier puis comparer les architectures
3. Comprendre les différences entre les types d'architectures
4. **Mesurer les différences**
    - **Présentation des architectures**
    - Mesure de l'influence d'une architecture maître-esclave
5. Conclusion

Architecture de référence : A-DTLS

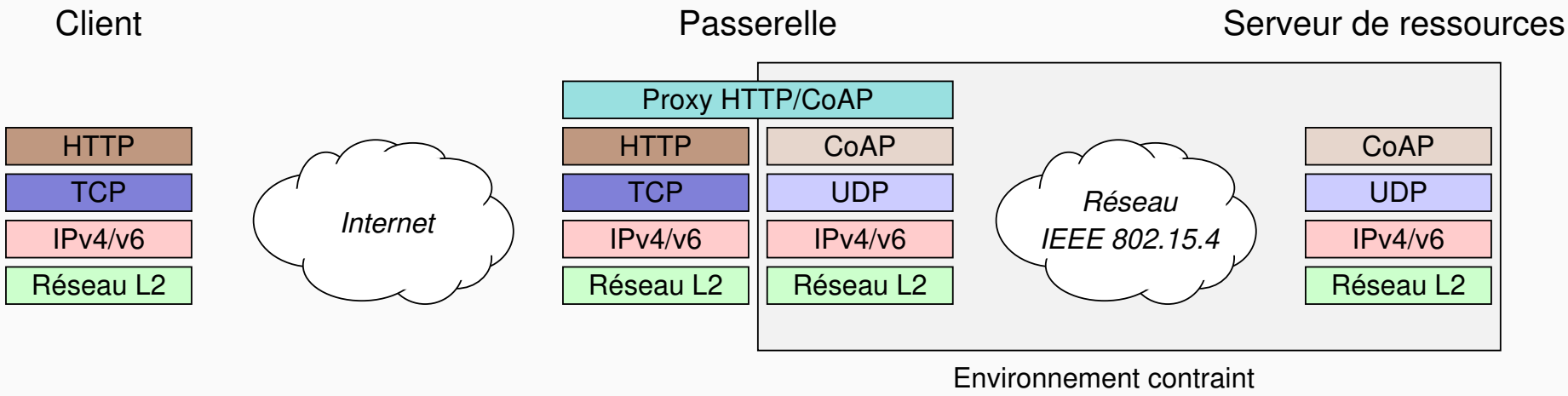


- communication de bout-en-bout
- même pile de protocoles (client, serveur de ressources)
- routeur 6LoWPAN pour réduire la taille des en-têtes

Architecture de référence utilisée pour les mesures

Exemple d'architecture déployée

Contexte : des capteurs à un saut de la passerelle



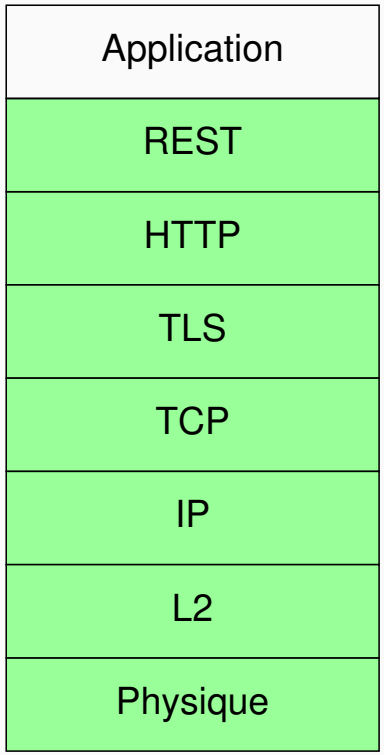
Problème sur les capteurs dans ce contexte

- fonctionnalités inutiles (couches réseau et transport)
- surcharge protocolaire importante

Constat ?

Constat

Pile de communication pour un service web



. . . et d'autres protocoles pour :

- la découverte de ressources (Bonjour)
- le multicast (pour Bonjour)
- les noms (DNS et DNS-SD)
- et bien d'autres. . .

3 décennies de protocoles réunies

Le tout dans une ampoule ?

Les solutions actuelles simplifient chaque couche, une autre approche est possible

CASAN (1/3)

Common **A**rchitecture for **S**ensor and **A**ctuator **N**etworks [1]

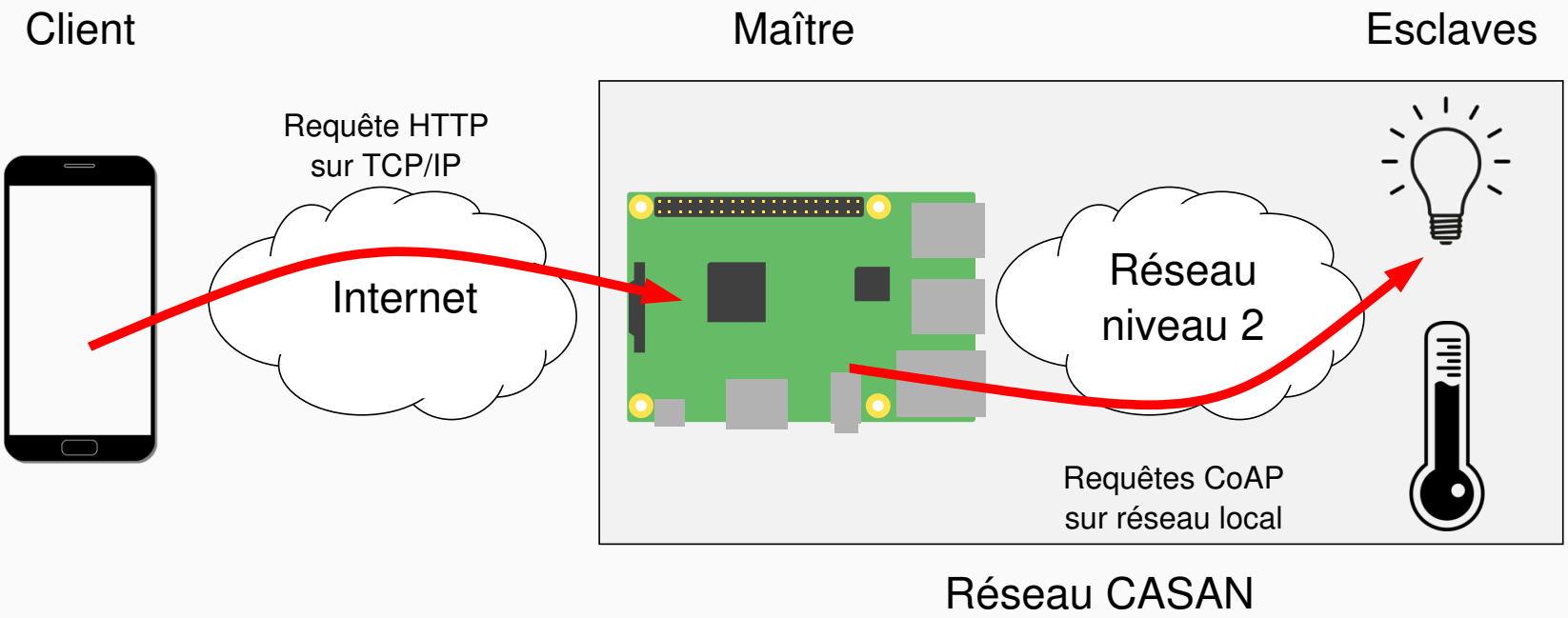
« Ne pas voir un objet contraint comme un ordinateur, mais comme un périphérique »

Idées générales

- réseaux contraints = appareils spécialisés
- complexité à déplacer sur du matériel adapté (maître)
- capteur = périphérique du maître
  mais passant par le réseau, qu'il faut donc sécuriser

1. David, P. and Noël, T., ***CASAN : A New Communication Architecture for Sensors Based on CoAP***, 2015, 12th IEEE International Conference on Networking, Sensing and Control

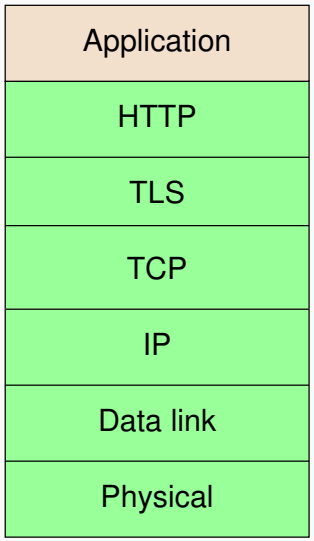
CASAN (2/3)



Fonctionnalités principales

- confidentialité des communications
  sur Internet et sur le réseau CASAN
- découverte de ressources simple
  le client n'a pas à connaître les serveurs de ressources
- faible complexité sur les capteurs

CASAN (3/3)

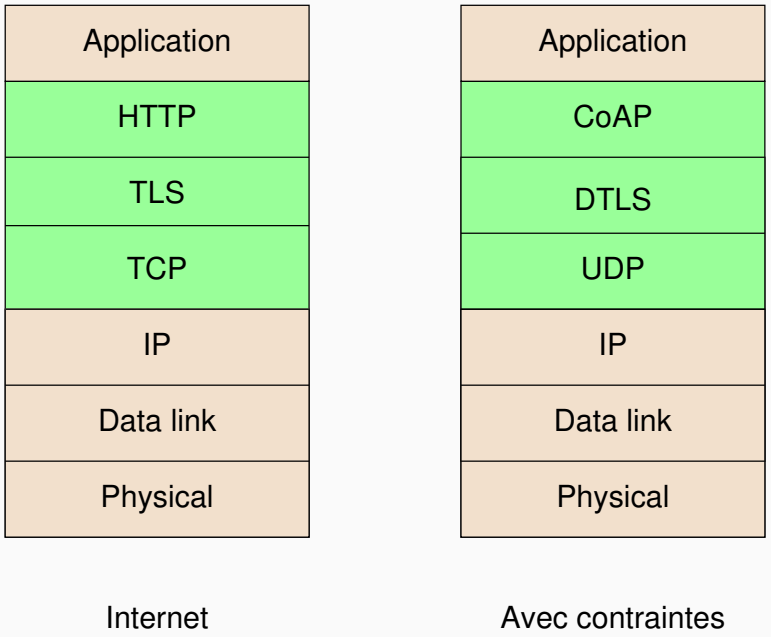


Internet

- Pile de communication classique sur Internet

2. Z. Shelby and K. Hartke and C. Bormann, ***The Constrained Application Protocol (CoAP)***, RFC 7252, Juin 2014

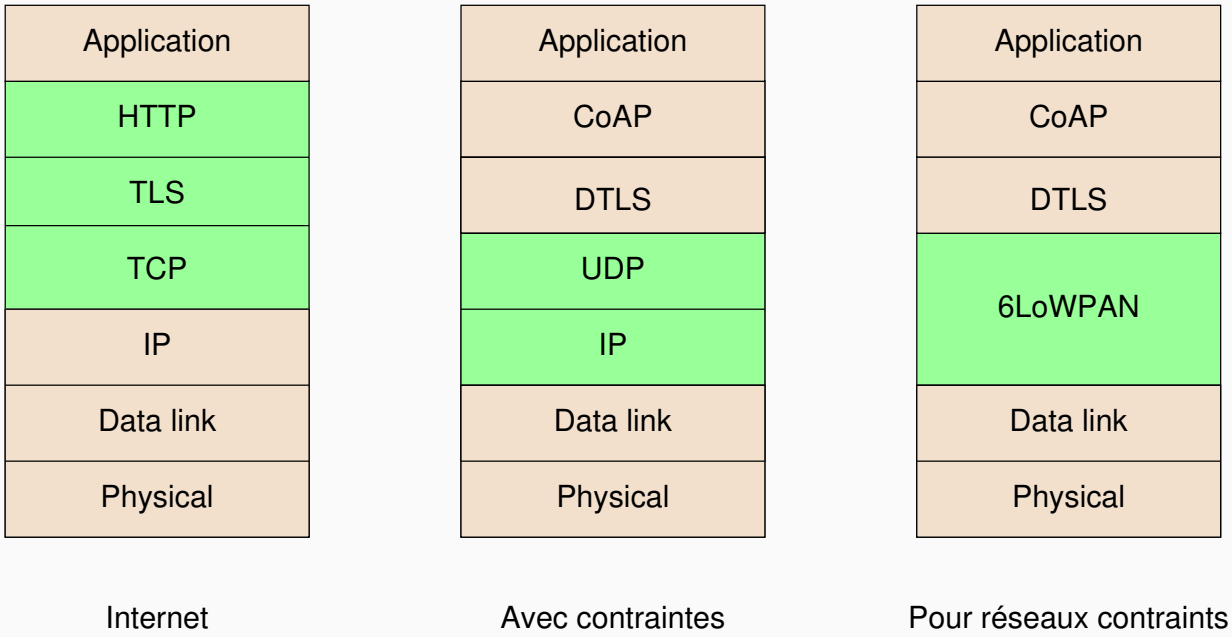
CASAN (3/3)



- Pile de communication classique sur Internet
- Couches transport, sécurité et application [2] plus simples

2. Z. Shelby and K. Hartke and C. Bormann, ***The Constrained Application Protocol (CoAP)***, RFC 7252, Juin 2014

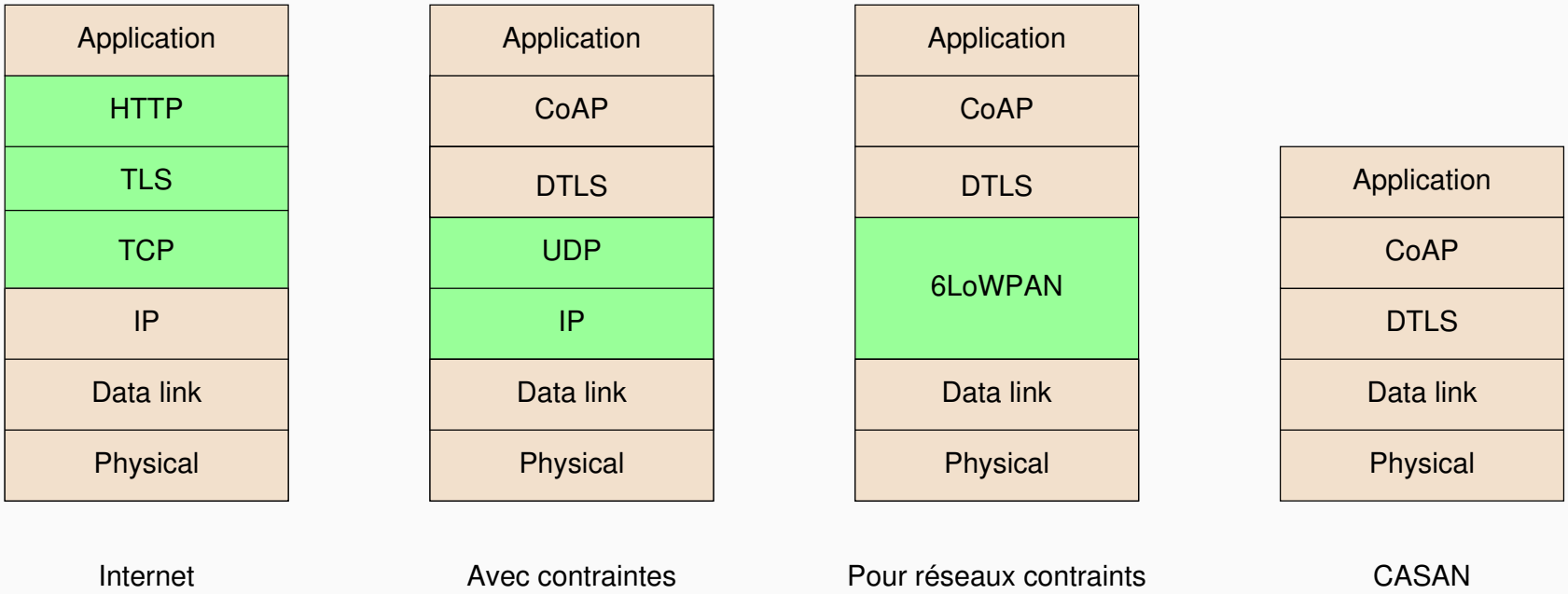
CASAN (3/3)



- Pile de communication classique sur Internet
- Couches transport, sécurité et application [2] plus simples
- Nouvelle couche « réseau et transport » spécialisée (vision IETF)

2. Z. Shelby and K. Hartke and C. Bormann, ***The Constrained Application Protocol (CoAP)***, RFC 7252, Juin 2014

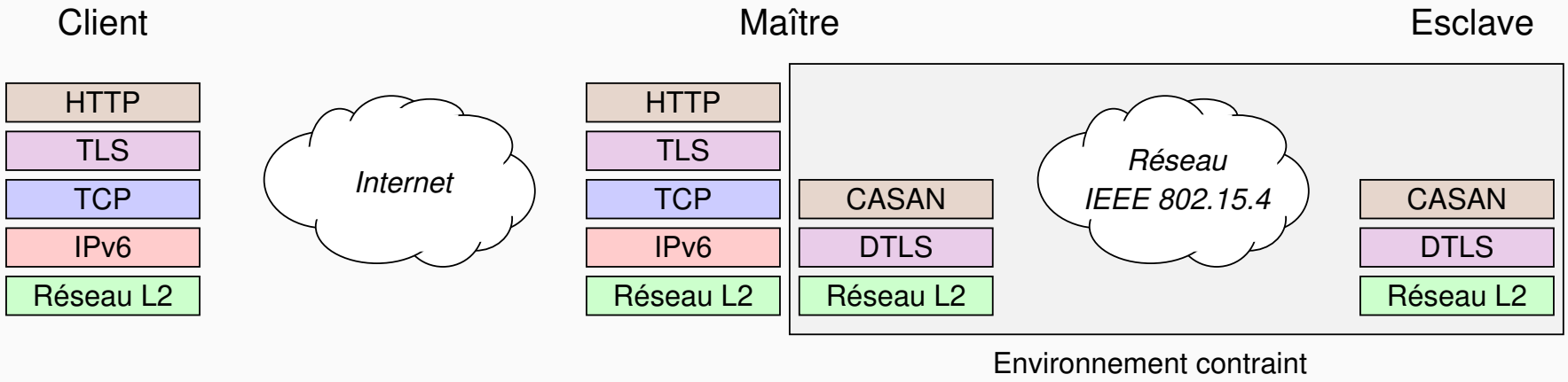
CASAN (3/3)



- Pile de communication classique sur Internet
- Couches transport, sécurité et application [2] plus simples
- Nouvelle couche « réseau et transport » spécialisée (vision IETF)
- Suppression des couches réseau et transport

2. Z. Shelby and K. Hartke and C. Bormann, ***The Constrained Application Protocol (CoAP)***, RFC 7252, Juin 2014

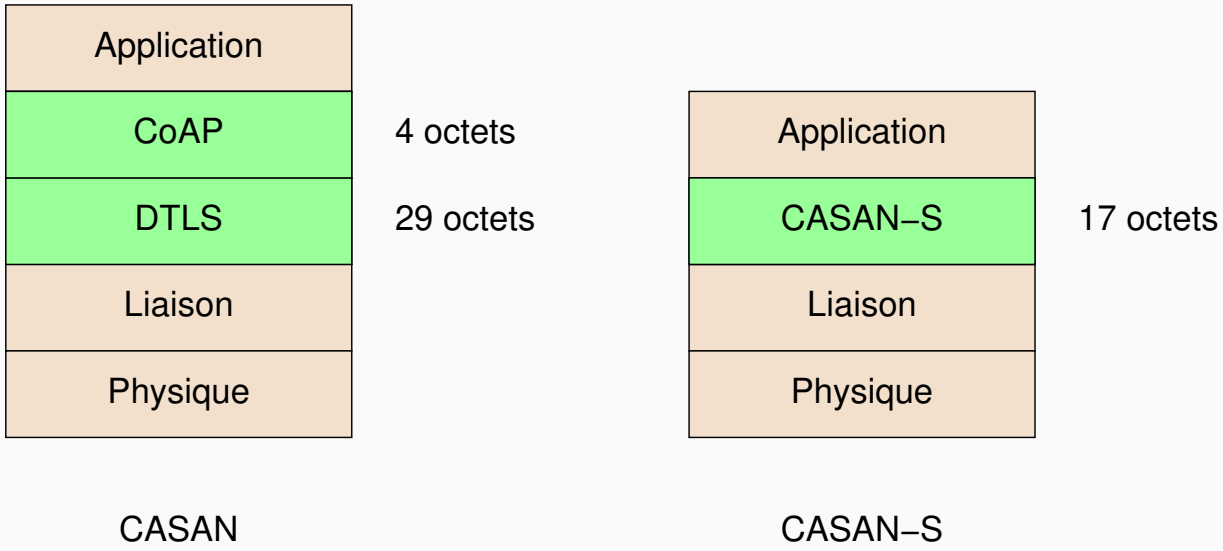
Architecture maître-esclave CASAN



- protocoles classiques sur Internet
- pile réseau réduite dans l'environnement contraint

Nous pouvons encore simplifier

Contribution : communication maître-esclave



Couche DTLS… qui n'est pas adaptée

- médium partagé : nombreux messages à la connexion
- mauvaise prise en charge des contraintes de taille des paquets

L'architecture maître-esclave permet une communication très simple entre le maître et l'esclave

Contribution : communication maître-esclave

Conception du protocole CASAN-S

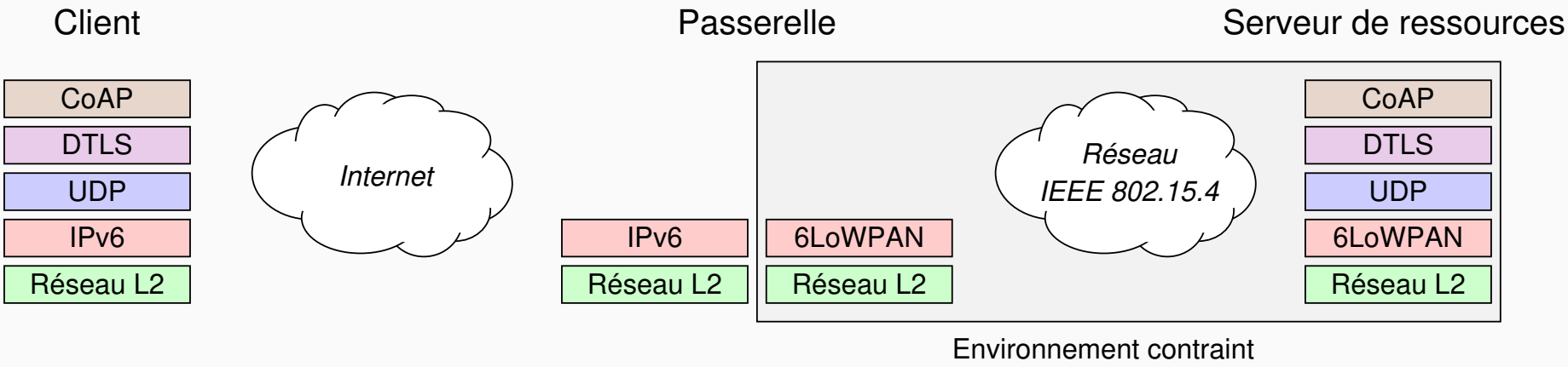
intégration $\rightarrow$ données publiques échangées pendant la négociation

cohérence $\rightarrow$ seulement CoAP

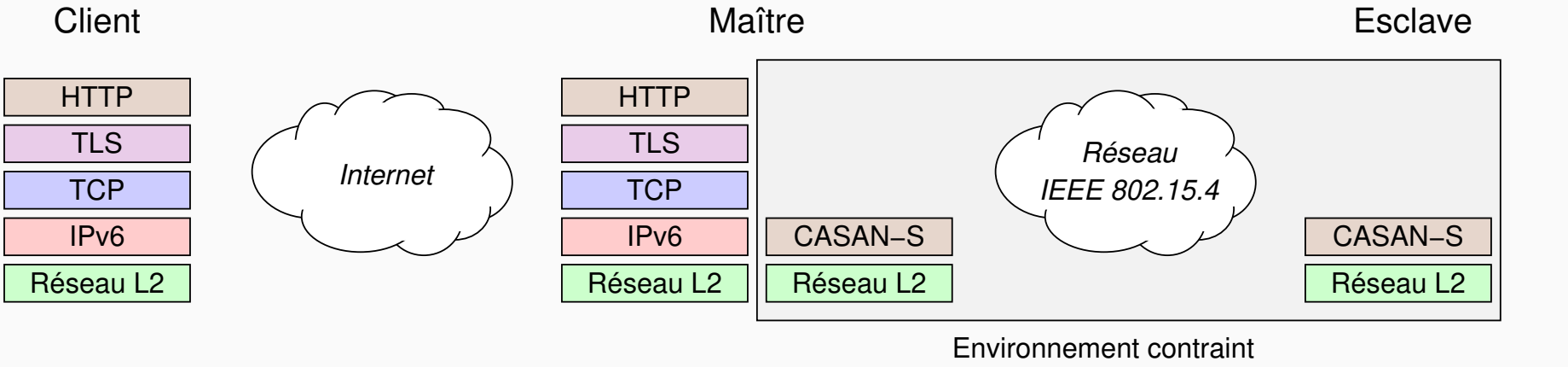
médium partagé $\rightarrow$ peu d'échanges

	CASAN + DTLS	CASAN-S
messages à la connexion	nombreux (13)	**modérés (5)**
aller-retours à la connexion	5	**3**
taille d'en-tête	33	**17**
charge utile 802.15.4 (octets)	85	**101**

Architectures mesurées



A-DTLS



CASAN-S

Plan

1. Introduction de la problématique
2. Identifier puis comparer les architectures
3. Comprendre les différences entre les types d'architectures
4. **Mesurer les différences**
    - Présentation des architectures
    - **Mesure de l'influence d'une architecture maître-esclave**
5. Conclusion

Plateforme expérimentale

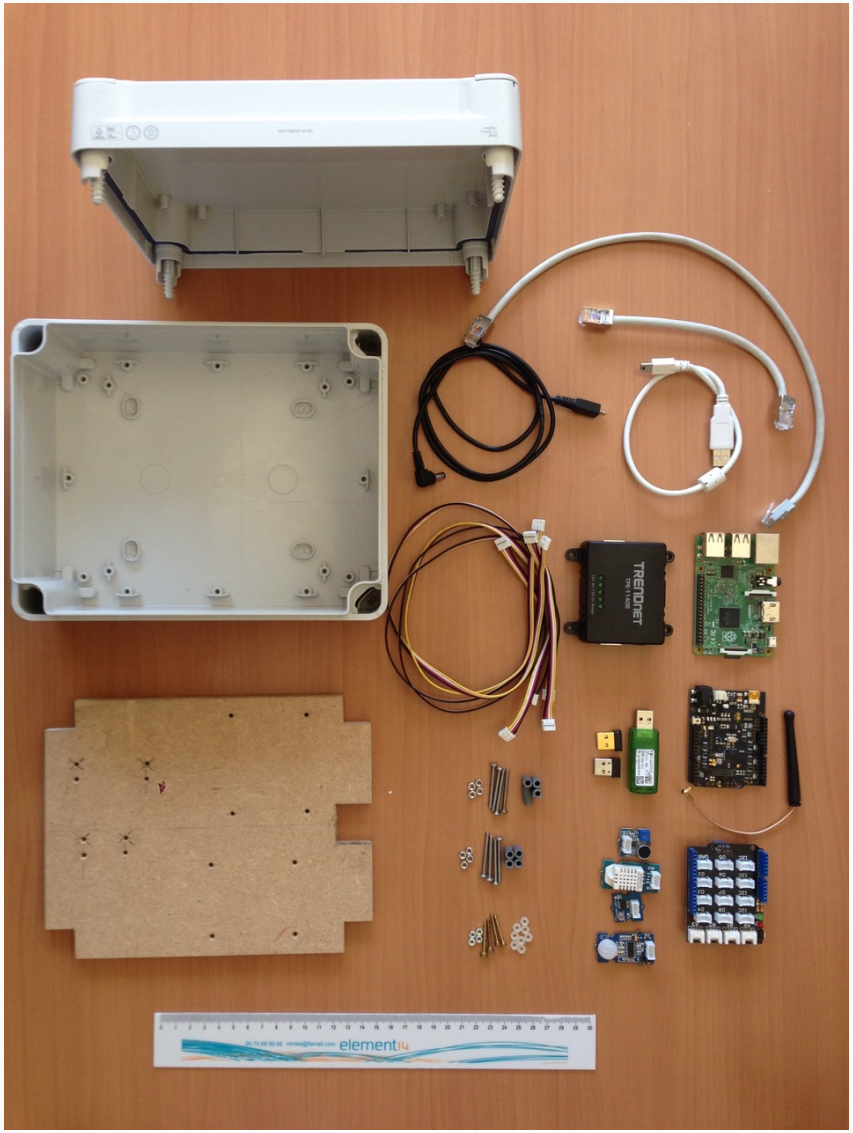
Objectif : comparer architecture de référence et maître-esclave

Plateforme expérimentale

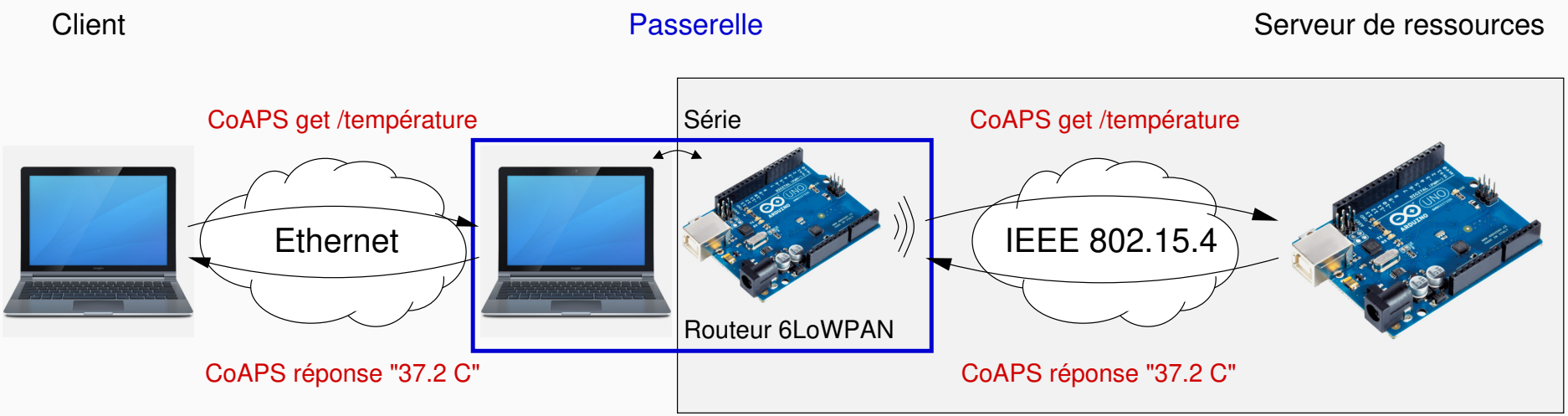
Objectif : comparer architecture de référence et maître-esclave

Plateforme expérimentale

Objectif : comparer architecture de référence et maître-esclave

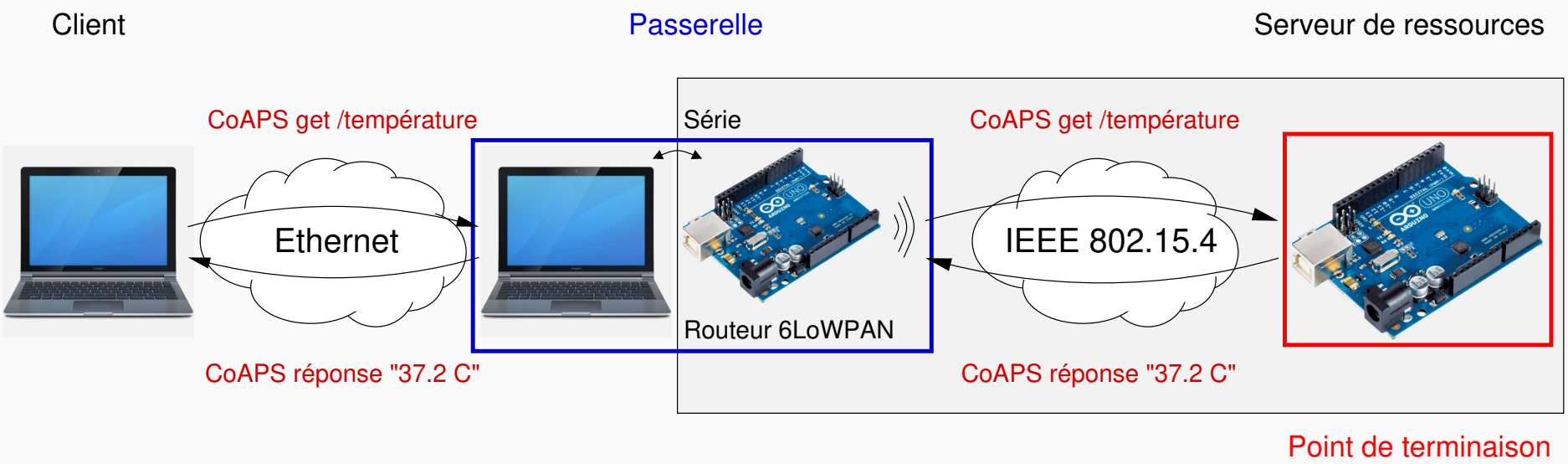


Dispositif expérimental : architecture de référence (1/2)



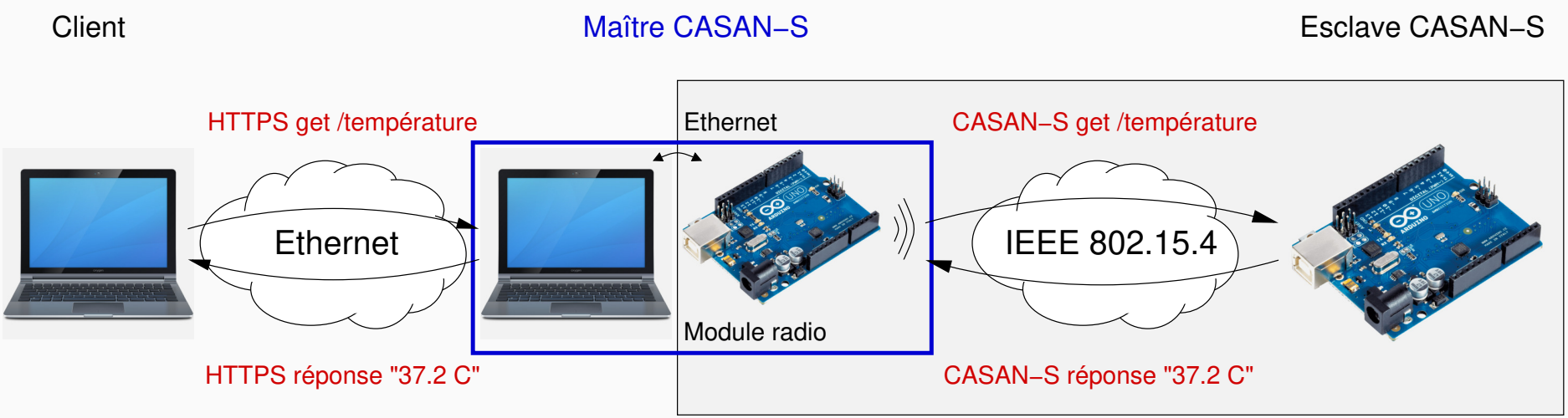
- protocoles : CoAP, DTLS, 6LoWPAN (et RPL)
- démarrage du réseau : récupération d'une adresse IP
- point de terminaison sur le serveur de ressources

Dispositif expérimental : architecture de référence (1/2)



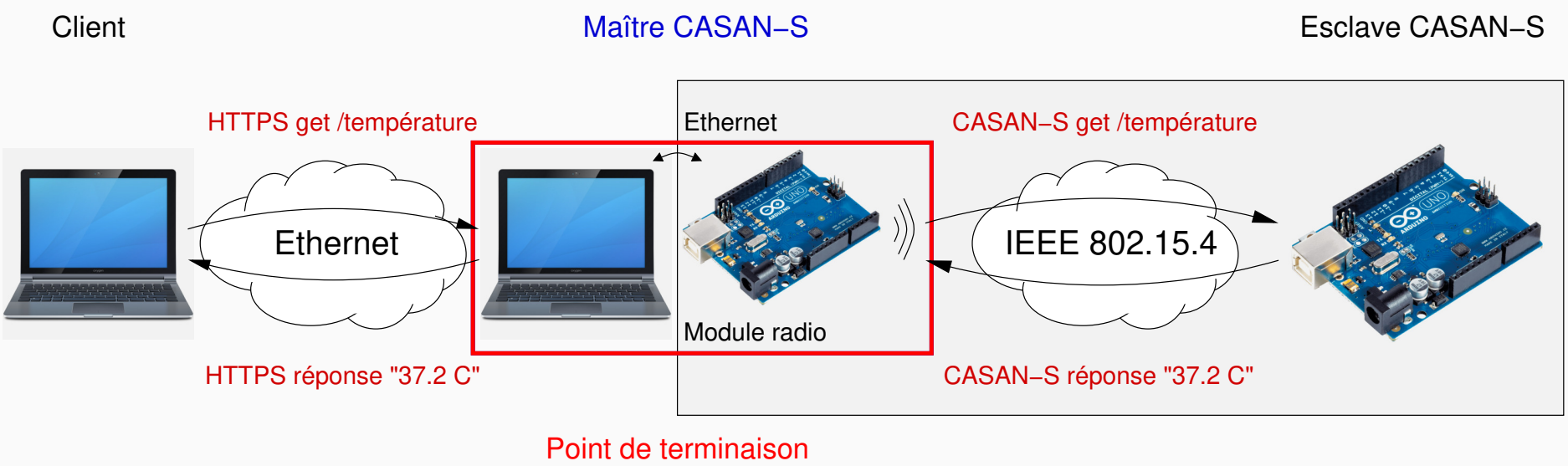
- protocoles : CoAP, DTLS, 6LoWPAN (et RPL)
- démarrage du réseau : récupération d'une adresse IP
- point de terminaison sur le serveur de ressources

Dispositif expérimental : architecture maître-esclave (2/2)



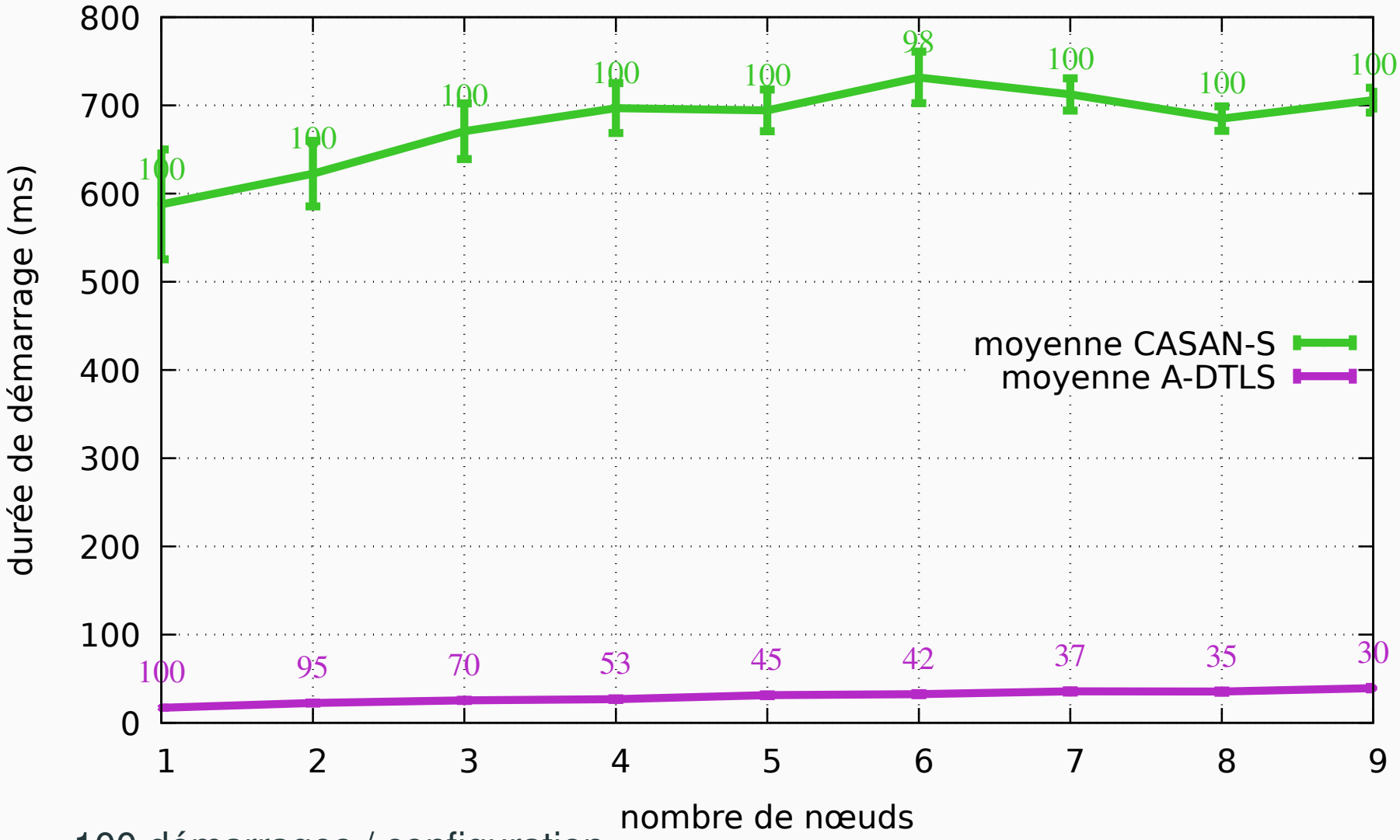
- protocole : CASAN-S
- démarrage du réseau : connexion maître-esclave
- point de terminaison sur le maître

Dispositif expérimental : architecture maître-esclave (2/2)



- protocole : CASAN-S
- démarrage du réseau : connexion maître-esclave
- point de terminaison sur le maître

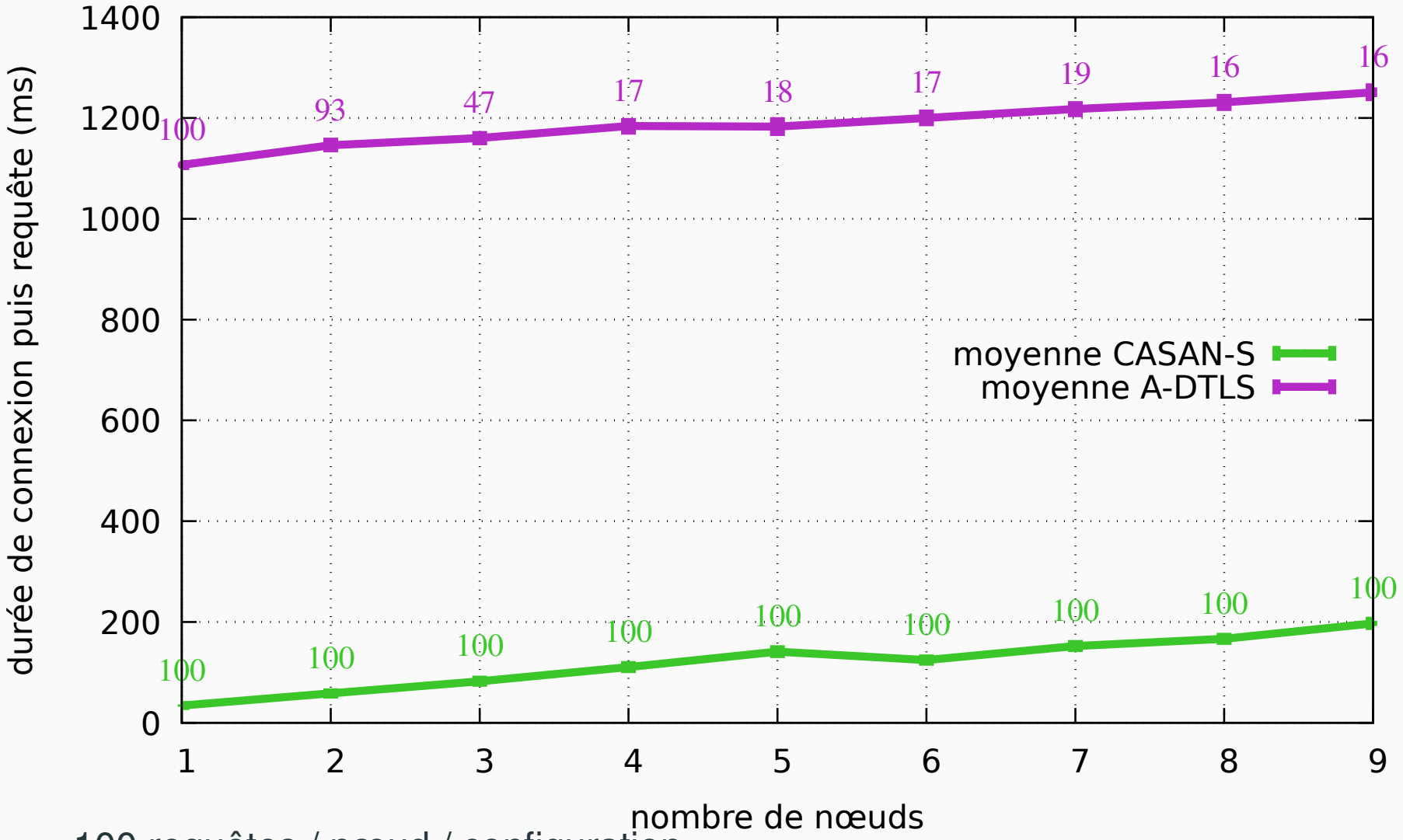
Résultats (1/2) : démarrage du réseau



100 démarrages / configuration

A-DTLS plus rapide : aucune opération cryptographique sur SR

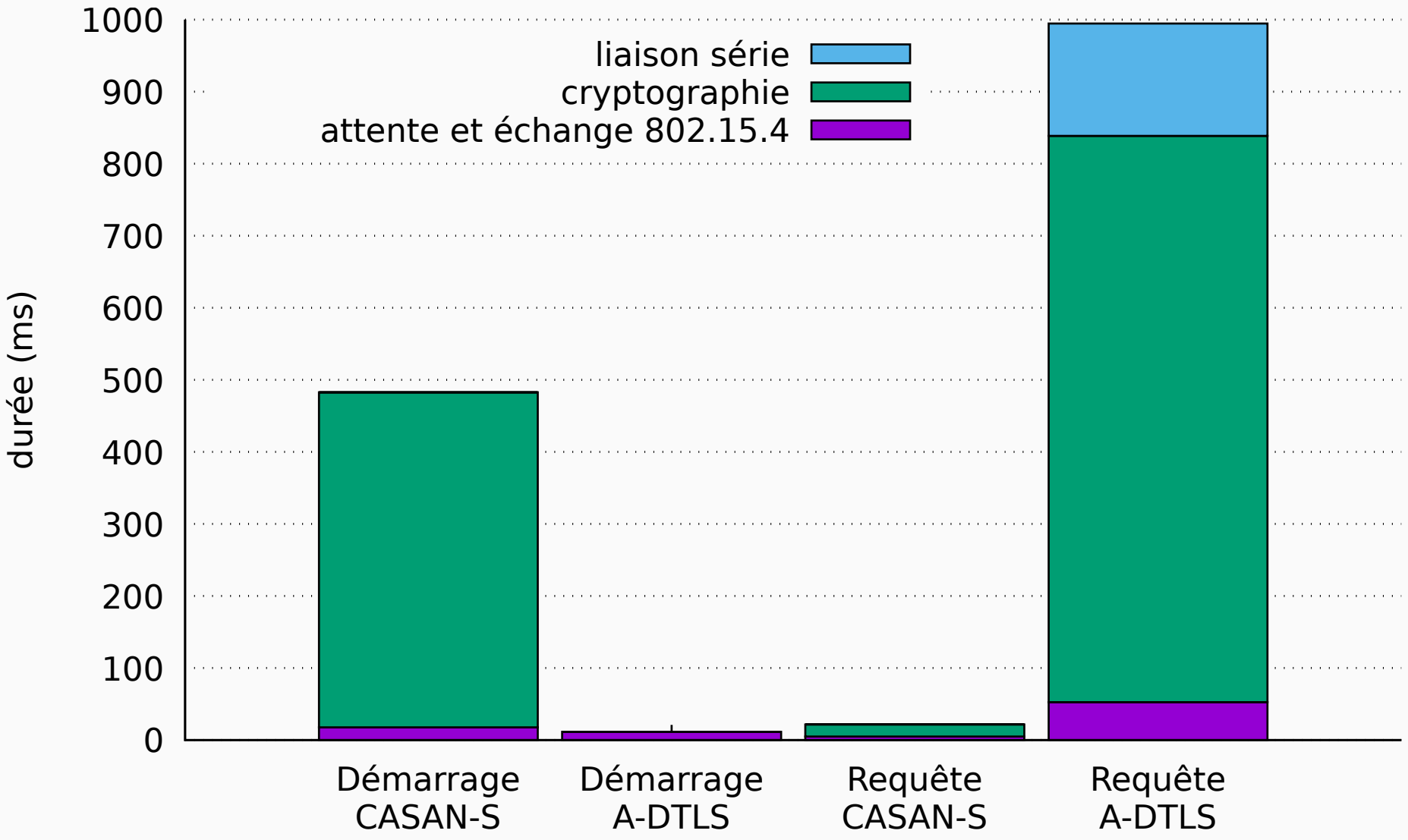
Résultats (2/2) : première requête d'un client



100 requêtes / nœud / configuration

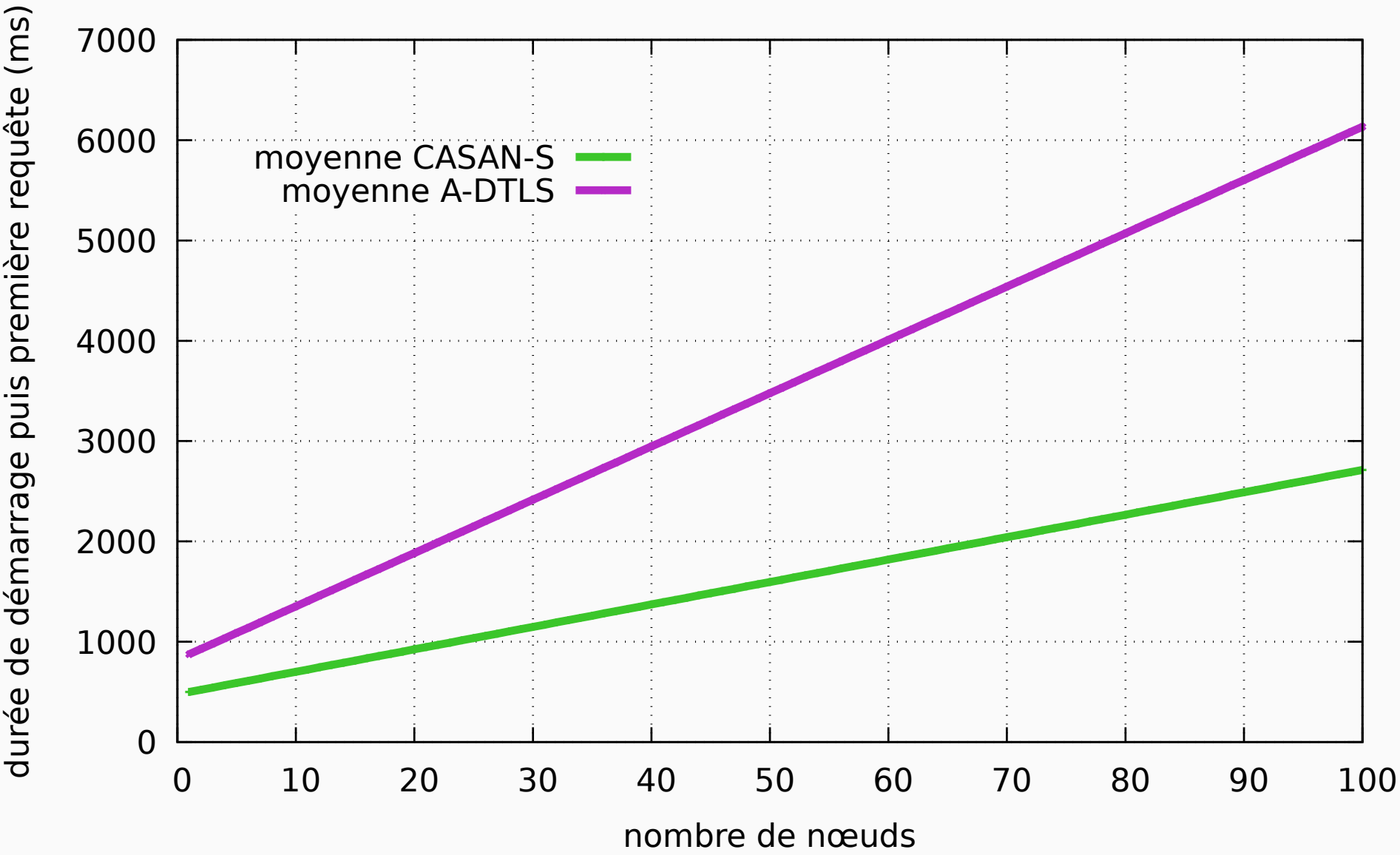
CASAN-S plus rapide : peu d'opérations cryptographiques sur SR

Durées d'échange



Temps réseau $\ll$ Temps cryptographie

Projection : démarrage puis requête (calculé)



CASAN-S plus rapide dans tous les cas

Plan

1. Introduction de la problématique
2. Identifier puis comparer les architectures
3. Comprendre les différences entre les types d'architectures
4. Mesurer les différences
5. **Conclusion**

Conclusion sur CASAN-S

Comparaison avec A-DTLS

	CASAN-S
Au démarrage d'un nœud	**~470 ms plus lent**
Pour chaque connexion d'un client	**~800 ms plus rapide**
Pour chaque requête (calculé)	**66 % plus rapide**

CASAN-S est une solution adaptée aux réseaux IoT

Conclusion sur les architectures maître-esclave

- authentification centralisée
- autorisation centralisée
- confiance centralisée
- cryptographie non limitée
- catalogue de ressources
- cache de données
- client et SR indépendants
- en-têtes minimales

Fonctionnalités offertes par les architectures maître-esclave étudiées

Bilan de nos contributions

- Définition des architectures de sécurité
- Catalogue puis comparaison des architectures
- Recommandations selon des critères de déploiement
- Amélioration du protocole pour l'architecture CASAN
- Mesure de l'influence d'une architecture maître-esclave
- Optimisation d'un protocole de communication sécurisée
  - n'atteint pas les avantages d'une architecture maître-esclave

Leçon tirée

Une architecture maître-esclave permet l'IoT sans les problèmes associés

- temps de connexion fortement réduits
- nouvelles fonctionnalités

Les nœuds sont toujours accessibles depuis Internet

Perspectives : étendre notre travail

Description d'une architecture exécutable

- Propriétés calculées automatiquement

Puis selon un usage prévu

- Calcul d'un score de mise à l'échelle
- Calcul de l'énergie consommée

Contributions

Contributions acceptées

- conférence AdHoc Now (Springer), 2016, *DTLS Improvements for Fast Handshake and Bigger Payload in Constrained Environments*
- conférence CoRes, 2016, *Améliorations de DTLS : connexion rapide et augmentation de la charge utile pour les réseaux contraints*
- journal AdHoc Networks (Elsevier), 2017, *Security Architectures in Constrained Environments : a Survey*

Contributions en cours

- journal AdHoc Networks (Elsevier), *Comparison of end-to-end and master-slave architectures for the IoT and its security*

Plan

1. Introduction de la problématique
2. Identifier puis comparer les architectures
3. Comprendre les différences entre les types d'architectures
4. Mesurer les différences
5. Conclusion
6. **Annexe : optimisation d'un protocole de bout-en-bout**

Optimisation d'un protocole de sécurité bout-en-bout

Objectif : diminuer le temps de connexion et la taille des en-têtes dans un environnement contraint pour une architecture de référence

	DTLS	DTLS optimisé
messages à la connexion	10	6
en-têtes après connexion	29	21

Avantages

- gain en mémoire et temps de connexion
- fragmentation réduite

Suite à l'amélioration du protocole

Conclusion

- Pas de nouvelle fonctionnalité
- Clients spécialisés (protocole non standard)
- Gains intéressants mais minimes en comparaison

Gain par rapport à A-DTLS	**DTLS optimisé**	**CASAN-S**
Durée de connexion	20 %	96,6 %
Taille des en-têtes (35 - 53 octets pour A-DTLS)	15 à 22 %	51 à 68 %

Les améliorations du protocole de sécurité n'offrent pas les avantages d'une architecture maître-esclave

Exemple d'architecture : A-DTLS

Entités	Autorités	Domaines	Relations
C, AN, AZ, SR	Client, Admin.	Client, Serveur	connexion puis requête

Domaine	Contrôlé par l'autorité	Inclut le(s) entité(s)
Client	Client	C
Serveur	Admin.	SR, AN, AZ

Relation	Qui parle à qui	Protocoles	Propriétés	Informations
R_1	C $\rightarrow$ SR	DTLS		id_C, version, id session, aléa, algorithmes
R_2	SR $\rightarrow$ C	DTLS		cookie
R_3	C $\rightarrow$ SR	DTLS		id_C, version, id session, aléa, algorithmes ...
R_4	SR $\rightarrow$ C	DTLS	I MA R AC	version, algo, aléa, ...
R_5	C $\rightarrow$ SR	DTLS	I MA R AC	id_C, preuve id_C, ...
R_6	SR $\rightarrow$ AN	$\perp$	$\perp$	$\perp$
R_7	AN $\rightarrow$ SR	$\perp$	$\perp$	$\perp$
R_8	SR $\rightarrow$ C	DTLS	C I MA R AC	[data] chiffré comme négocié
R_9	C $\rightarrow$ SR	DTLS, CoAP	C I MA R	requête
R_{10}	SR $\rightarrow$ AZ	$\perp$	$\perp$	$\perp$
R_{11}	AZ $\rightarrow$ SR	$\perp$	$\perp$	autorisation
R_{12}	SR $\rightarrow$ C	DTLS, CoAP	C I MA R	réponse

Extrait de la comparaison

	MQTT-SN	A-DTLS
Éléments de sécurité		
cryptographie asymétrique (SR)	non	oui (si client non local)
autorisation explicite	oui	non
granularité d'autorisation	ressource	SR
gestion automatique des clés	oui	oui
contexte par client (SR)	non	oui

Comparaison sur la sécurité des architectures (protection des communications, autorisation. . .)

Extrait de la comparaison

	MQTT-SN	A-DTLS
Aspects contraintes		
nb relations entre domaines	très nombreuses	nombreuses
horloge requise (SR)	non	oui (si client non local)
data cache service	oui	non
surcharge taille de messages	grande, DTLS (29 octets)	grande, DTLS (29 octets)
nb msg 1ère connexion (SR)	nombreux	nombreux
nb msg 1ère connexion (C)	nombreux	nombreux
nb msg à chaque requête (SR)	aucun (asynchrone)	plusieurs
nb msg à chaque requête (C)	1 à 2 / mise à jour	2
connexions maintenues (SR)	1 (avec un proxy)	0
surcharge de messages (SR)	dépend de la fréquence de màj	dépend du nb de clients et fréquence des requêtes
Aspects généraux		
protocoles utilisés L	DTLS, MQTT-SN	DTLS, CoAP
arch avec app spécifique	oui	non
découverte de ressources	non	non

. . . et sur les contraintes du matériel et des réseaux