Protocole xAAL - Version 0.1-json

Corinne Le Moan

Mars 2012

1 Transport

Bus multicast IP (adresse et numéro de port TBD)

2 Device

Un device possède :

- Un type (integer) composé de :
  - Un numéro de méta-type (short) désignant la catégorie de types (e.g. éclairage, chauffage, ...)
  - Un numéro (short) de sous-type qui renvoie à un schéma device (variable, action/response, event).

Le type 0xFFFFFFFF est réservé et signifie "any", désigne tous les types. On réserve une plage particulière "expérimentale" (i.e. lorsque quelqu'un fait une implem et n'a pas encore obtenu un numéro de notre bureau d'attribution). Par exemple "0xFF89" pour le méta type et idem pour le sous-type.

- Un identifiant (ou adresse), numéro unique sur le bus
  - codé sur 1 mot de 32bits
  - Génération du numéro :
    - codé en dur en usine,
    - auto généré (random),
  - Il est vérifié par un genre de "arp gratuit" par des requêtes "IsAlive"
  - Il n'est pas attribué par un superviseur
- Un équipement peut être composé de plusieurs sous-équipements (embedded) et aura ainsi plusieurs types et plusieurs IDs (i.e. thermomètre + hydromètre)
- Dans le même ordre d'idée, une gateway aura dans ses "embedded" tous les équipements administrés par elle.

3 Schéma de type de device

Chaque équipement est typé, i.e. correspond à un schéma. Le schéma est fortement inspiré de UPnP (cf [[http:*upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf*]], chapitre 2).

Un schéma définit :

- Une liste d'actions
  - Chaque action possède :
    - un nom
    - une liste d'arguments, chaque argument étant défini par :
      - un nom
      - une direction (in ou out)
      - un type (?)
- Une liste de variables (leur changement de valeur génère par défaut un évènement sur le bus), chaque variable étant définie par :
  - un nom,
  - un type
  - une valeur par défaut (?)

4 Composants particuliers du bus

5 Gateway

Une passerelle ou gateway est un composant masquant un protocole plus bas niveau sur le bus (xAP, X10, X2D, X3D, ...)

6 Annuaire

Le service d'annuaire est facultatif. Il permet d'administrer la complexité du réseau d'équipements en leur attachant des informations de localisations, de présentations voires de fonctionnalités. Concrêtement, par soucis d'ouverture et d'extensibité (au sens souplesse d'utilsilisation et non au sens monté en charge), ces informations sont matérialisés par des tags.

7 Message

Tous les messages transmis sur le bus respectent la syntaxe suivante :

```
{
    "msg" :
       <!--2 elements array -->
       [   <!-- first element is the header of message -->
         {
           "msgtype":<message type id>,
           "source":<device ID source>,
           "target":<device ID destination>,
           "devtype":<device type>
         }
         ,
         <!-- second element is the body of message -->
         {             ...          }
       ]
}
```

8 Description du header

```
{
  "source":<ID source>,
  "target":<ID destination>,
  "devtype":<Type device>,
  "msgtype":<Type message>
}
```

Un ID est codé sur 32 bits. Un ID de "broadcast" = 0xFFFF. Le type de message est un entier sur 32 bits (cf détails plus bas) et le type de device est celui de la destination si "Type Message" ¡256, la source si "Type Message" ¿255.

9 Type de message

- Le type de message est codé par un entier 32bits non signé
- L'octet de poids fort indique la version du protocole (e.g. ici on parle du protocole 0x1)
- Le bit de poids faible indique si c'est un message de type requête (0x1) ou de type réponse/notification (0x0). Ce bit permet donc de déterminer la portée du DeviceType de l'entête (respectivement celui de la destination, respectivement celui de la source).
- Un message de type requête:
  - DeviceType = ClassID, TypeID

    - soit 0xFFFFFFFF (*any*): on s'adresse à tous les types de devices sur le bus
    - soit un ClassID précis, et un TypeID=0xFFFF: on s'adresse à tous les types d'une classe donnée
    - soit un ClassID précis et un TypeID précis: on s'adresse à un type bien déterminé
  - SrcID: le DeviceID de l'émetteur du message (peut être l'adresse 0x0 dans certains cas, par exemple pour gérer le genre d'arp gratuit, lorsque l'on n'a pas encore choisi un ID pour soi-même)
  - DstID: le DeviceID d'un device particulier à qui l'on adresse une requête, ou bien l'adresse de broadcast 0xFFFFFFFF

- Un message de type réponse ou notification
  - DeviceType: celui du device qui émet la requête (et pas de any d'aucune sorte)
  - SrcID: le DeviceID de l'émetteur du message
  - DstID: le DeviceID du device qui a fait la requête et à qui l'on répond, ou bien l'adresse de broadcast 0xFFFFFFFF, notamment pour les messages de notification

10 Les différents types des messages

10.1 Requête de type "Who is alive?" (0x01000101)

- Pas de body
- Sollicite des annonces de type "Alive" pour découvrir qui est actif sur le bus
- De manière globale: DstID broadcast, et DeviceType any
- Spécifiquement à un ClassID précis
- Spécifiquement à un ClassID+TypeID précis
- Spécifiquement à un DeviceID précis

10.2 Annonce (ou réponse) de type "Alive" (0x01000100)

- Dans le body :
  - la liste des équipements "embedded", rien si aucun,
  - le device ID de l'"embedder", rien si aucun
- Le device s'annonce spontanément sur le bus lorsqu'il est branché

- Le device s'annonce régulièrement (selon une périodicité de son choix, voir jamais)
- Le device s'annonce suite à une requête "Who is alive?"
- Le device peut choisir de ne pas s'annoncer s'il est en mode économie d'énergie, ou s'il ne sait pas traiter des requêtes " *Who is alive?*"

10.3 Requête de notification d'état (0x01000201)

- Pas de body
- Sollicite un message de notification d'état
- De manière globale, spécifique, ou entre les deux (cf plus haut)

10.4 Message de notification d'état (0x01000200)

- Body :
  - L'intégralité des variables d'état du device tel que définit dans son type
  - Le nom de la ou des variables qui ont changé de valeur
- Le device émet ce message :
  - spontanément, c'est-à-dire lorsque son état a changé,
  - en réponse à une requête de status du device, le device peut choisir de ne pas répondre à la requête correspondante s'il ne sait pas écouter le bus par exemple.

10.5 Requête de type action (0x01000301)

- Toujours de manière spécifique (pas de broadcast ou de any)
- Body:
  - Un RequestID: un entier de 64bits non signé (aléatoire)
  - Un ActionID: le numéro de la requête tel que définit dans le schéma
  - Une table de paramètres: les paramètres requis pour l'action en question, tel que définit dans le schéma (Absent si l'action ne requiert pas de paramêtre)

10.6 Réponse à une action (0x01000300)

- Répond spécifiquement à l'émetteur de la requête (i.e. sur son DeviceID à lui).
- Body:
  - RequestID: recopie celui de la requête
  - ActionID: rappel le numéro d'action de la requête
  - ReturnCode: un entier 32bits signé. Toujours présent.
  - Une table de variables de retours tel que définit dans le schéma (Absent si l'action ne fournit pas de variables en retour)
- Note: la spécification ne dit rien quant à l'absence de réponse. D'une part, puisque l'on est en UDP, la requête comme la réponse peut se perdre. D'autre part, il y a des device qui ne savent qu'émettre et pas écouter sur le bus. La spécification n'impose pas de mécanisme de timeout: certains device n'ont pas nécessairement d'horloge sous la main, et peuvent gérer les réponses aux requêtes dans un buffer circulaire (et si une réponse arrive tellement tard que l'on a oublié la question, tant pis)
- Autre argument concernant la non-réponse à une requête: dans le cas d'une requête non unicast (soit réellement broadcast, soit sur une classe de type), il parraît naturel que les équippements qui ne peuvent honorer la requête puissent choisir de se taire plutôt que de polluer le bus avec des messages d'erreur.

11 Meta-type d'équipements

Valeur num	Valeur texte	Description
FFFF	any	any
FF89	tbd	TBD
0001	soft	software
0002	light	light
0003	hvac	HVAC (heating, ventilation, and air conditioning)
0004	shutter	Shutter
0005	sensor	Sensor
0006	security	Security
0007	command	Command

12 software

0001	soft.x2dgateway	X2DGATEWAY
0002	soft.monitoring	Command and monitoring application

13 light

FF89	light.tbd	TBD
0001	light.switch	simple switch light
0002	light.dimmable	dimmable light

14 HVAC

FF89	hvac.tbd	TBD
0001	hvac.heater	heater

15 Shutter

FF89	shutter.tbd	TBD
0001	shutter.deltadore.4630	Deltadore roller shutter 4630

16 Sensor

17 Security

18 Command

FF89	command.tbd	TBD
0001	command.switch.onoff	on/off switch
0002	command.remotecontrol.deltadore.tyxia1601	1 shutter, 2 lights, remote control