Architecture et fonctionnalités du serveur de données

Ref : TV080114_TU_SM

Rédacteur : Serge Morvan

Présentation

Le module d'acquisition de données de NaVisu est distribué. Un serveur reçoit les données des différents capteurs les analyse, les traite puis les distribue au format XML [ou JSON (option)] aux différents clients. Dans la version la plus compacte serveur et client se trouvent sur une même machine. Le protocole de communication utilisé sont les WebSockets. [1] ce protocole permet les actions bidirectionnelles, *push* et *pull*, avec les clients. Actuellement seules les requêtes *pull* venant des clients sont prises en compte, mais il est simple d'introduire des actions *push* de la part du serveur pour des remontées d'alarmes par exemple. Le protocole WebSocket est implémenté dans la plupart des langages actuellement. L'implémentation du serveur repose sur le *framework* Vert.x [2] [3], ce *framework* assure la communication asynchrone entre les entrées et les sorties du serveur, à l'aide de bus événementiels : `EventBus`. Il est simple à utiliser, ne nécessite pas d'installation particulière. L'intérêt de la distribution est la possibilité des développer des clients ayant différentes technologies d'interfaces : JavaFX, HTML5 et différents OS : Windows7/Windows8, Mac OS, iOS, Linux ou Android. les clients peuvent être une application NaVisu complète ou de simple affichages de données capteurs : des instruments par exemple, ou des clients de communication type chat.

Les données en entrée

Les données sont envoyées au serveur via les ports série, USB ou Internet. Il est aussi possible d'avoir des données à partir de fichiers, pour un rejeu de parcours par exemple. L'acquisition de données est réalisée par des objets `Reader`,

- `SerialReader` pour la communication série
- `HTTPReader` pour la saisie sur internet
- `FileReader` pour la lecture de données sur disque

Le composant `DataServer` peut créer en dynamique les différents `Reader` et les paramétrer.

1. http ://fr.wikipedia.org/wiki/WebSocket
2. http ://vertx.io/
3. http ://fr.wikipedia.org/wiki/Vert.x

Exemples de débits sur les ports série :

- 4800 bauds (GPS, centrale de navigation, ..., NMEA0183 ASCII)
- 38400 bauds (AIS, NMEA0183 binaire)
- 250 Kbits/sec (GPS, centrale de navigation, moteurs, ...,N2K binaire)

Exemples de frames pour les entrées capteurs :

```
NMEA0183   : $GPRMC,093518.470,A,4826.2552,N,00429.8708,W,000.0,203.5,051113,,,A*7A
AIS        : !AIVDM,1,1,,B,13IMVT0000OcEt8KcIfW75F@0<0>,0*52
NMEA 2000 : <0x18eeff01> [8] 05 a0 be 1c 00 a0 a0 c0
```

L'architecture [4]

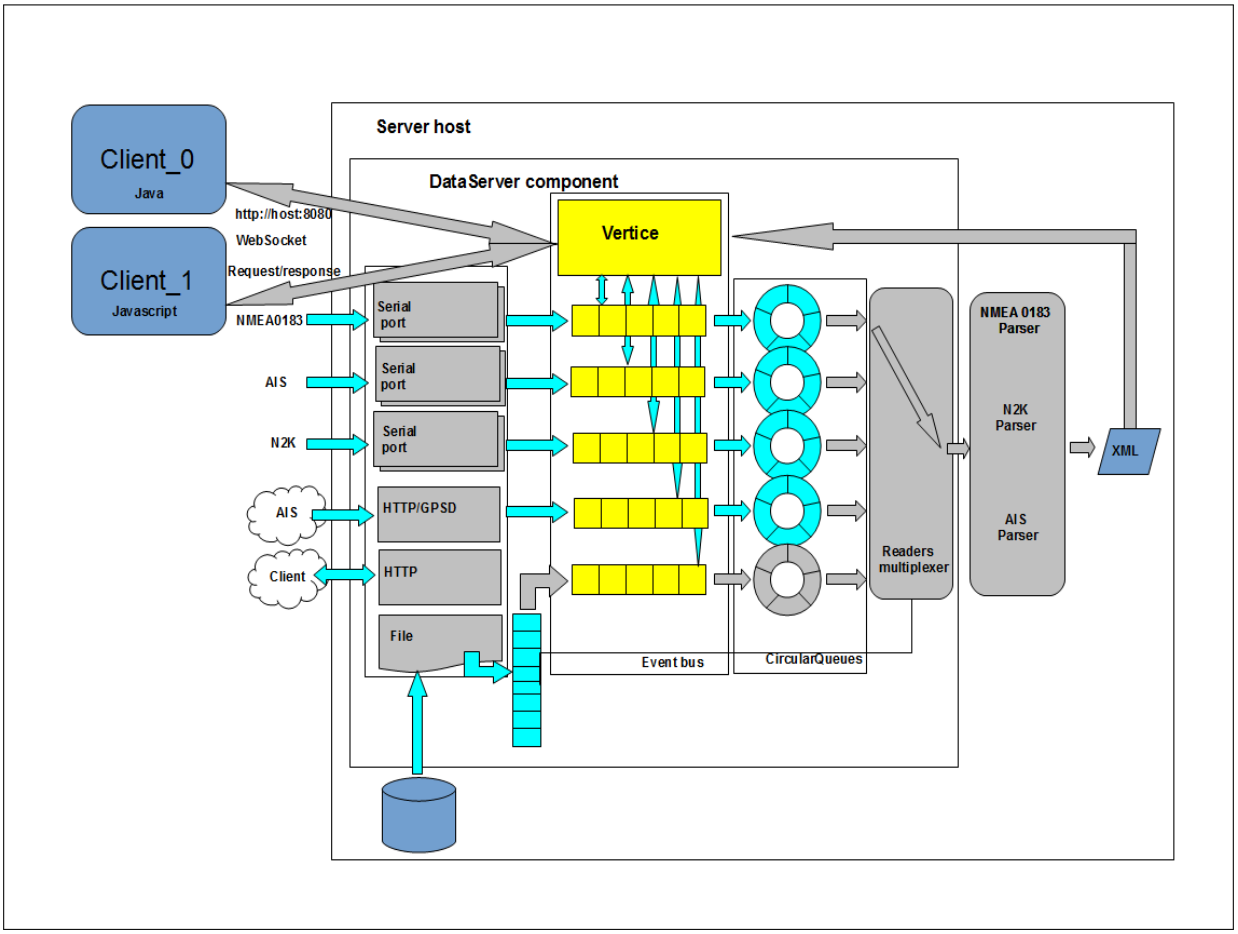

FIGURE 1 – *Aucune requête client, les données dynamiques venant des capteurs sont perdues*

Un `Reader` spécialisé est associé à chaque entrée, un bus d'événements Vert.x lui est attribué, ainsi qu'une file circulaire. En entrée continu les données sont envoyées, via leur bus, à leur file. En sortie le débit d'acquisition est piloté par chaque client.Si il n'y a pas de requêtes client, les nouvelles données écrasent les anciennes, sauf pour celles venant de fichiers qui sont transférées

4. Dans les figures qui suivent les flux de données dynamiques sont coloriés en turquoise, les canaux de communication en gris clair.

intégralement. Un sélecteur de `Reader` assure le multiplexage des données, actuellement la stratégie du choix est simple parcours circulaire, mais une stratégie avec file à priorité peut être facilement mise en œuvre. Sur requête d'un client la file circulaire active délivre ses données au sélecteur d'analyseur syntaxique : NMEA0183, AIS, N2K. Le parseur choisi traite les données, génère des objets NMEA, les transforme au format XML et les envoie à l'objet `Vertice` qui les envoie au client. Une nouvelle file est sélectionnée dans l'attente d'une autre requête client. Les données sont donc analysées (*parsées*) deux fois, une première fois par le parseur NMEA spécialisé, une deuxième fois par le client à partir de d'un format XML. Ceci se justifie par le fait que l'écriture du premier parseur est complexe du fait de la grande variabilité des formats NMEA, les connexions électriques avec les capteurs peuvent aussi engendrer un bruitage des données, ces données sont ensuite mises en forme suivant un schéma XML unique, les clients n'ont aucune difficultés à les transformer en objets à partir de l'API JAXB. Dès que le format JSON sera bien intégré à JEE, ce format sera préféré à XML. D'autres protocoles, tel que CoAP sont envisageables dans l'avenir.

La dynamique

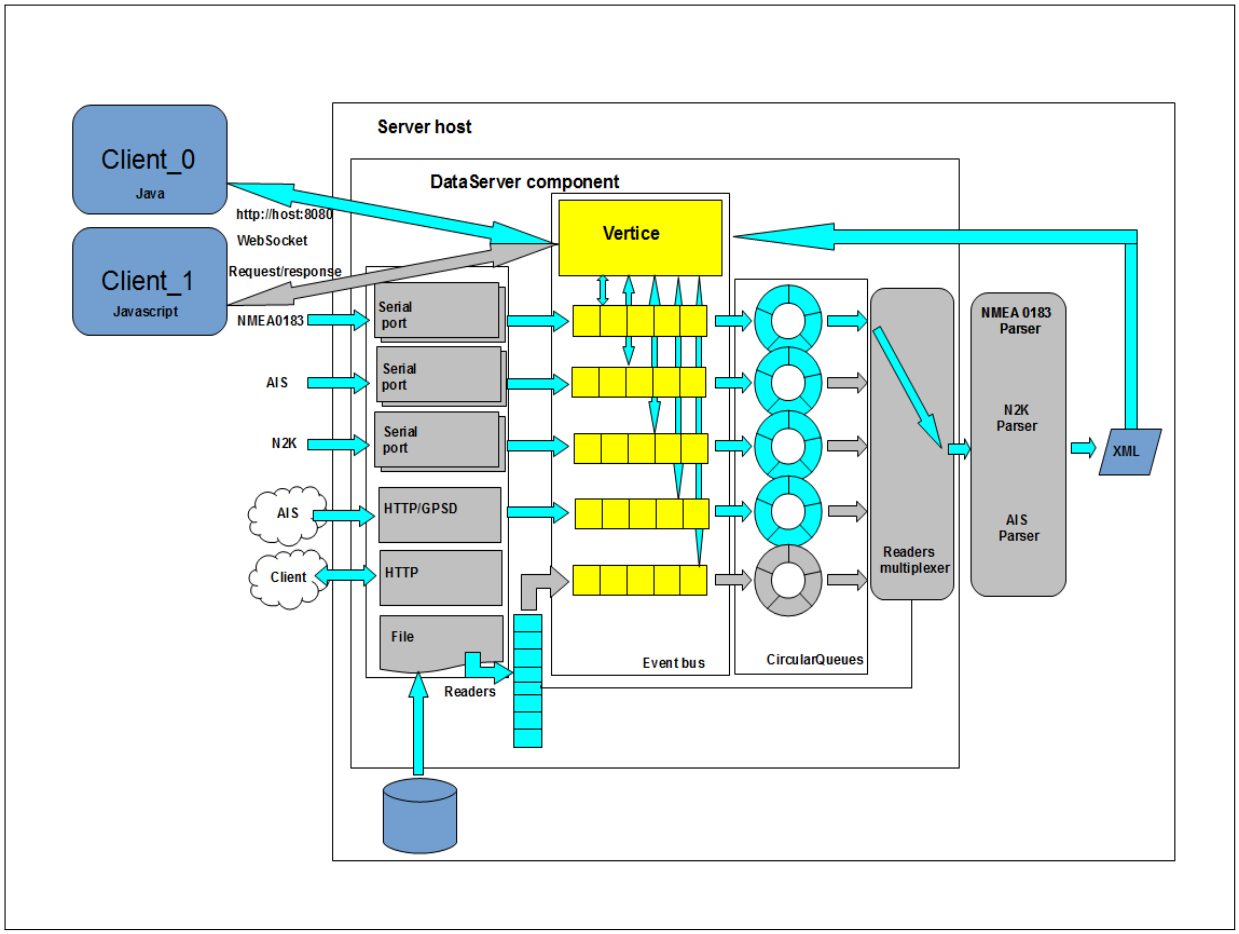

FIGURE 2 – *Exemple de requête client, suivie d'une réponse asynchrone à partir du premier port série enregistré.*

Performances

En test, le serveur supporte une requête client toutes les milisecondes, dans la pratique, un navire naviguant à 15 nœuds parcourt 7.5m en une seconde, un barreur va régler le débit d'affichage des données à 2 ou 3 secondes. Une période de 1 seconde pour les requêtes semble raisonnable.

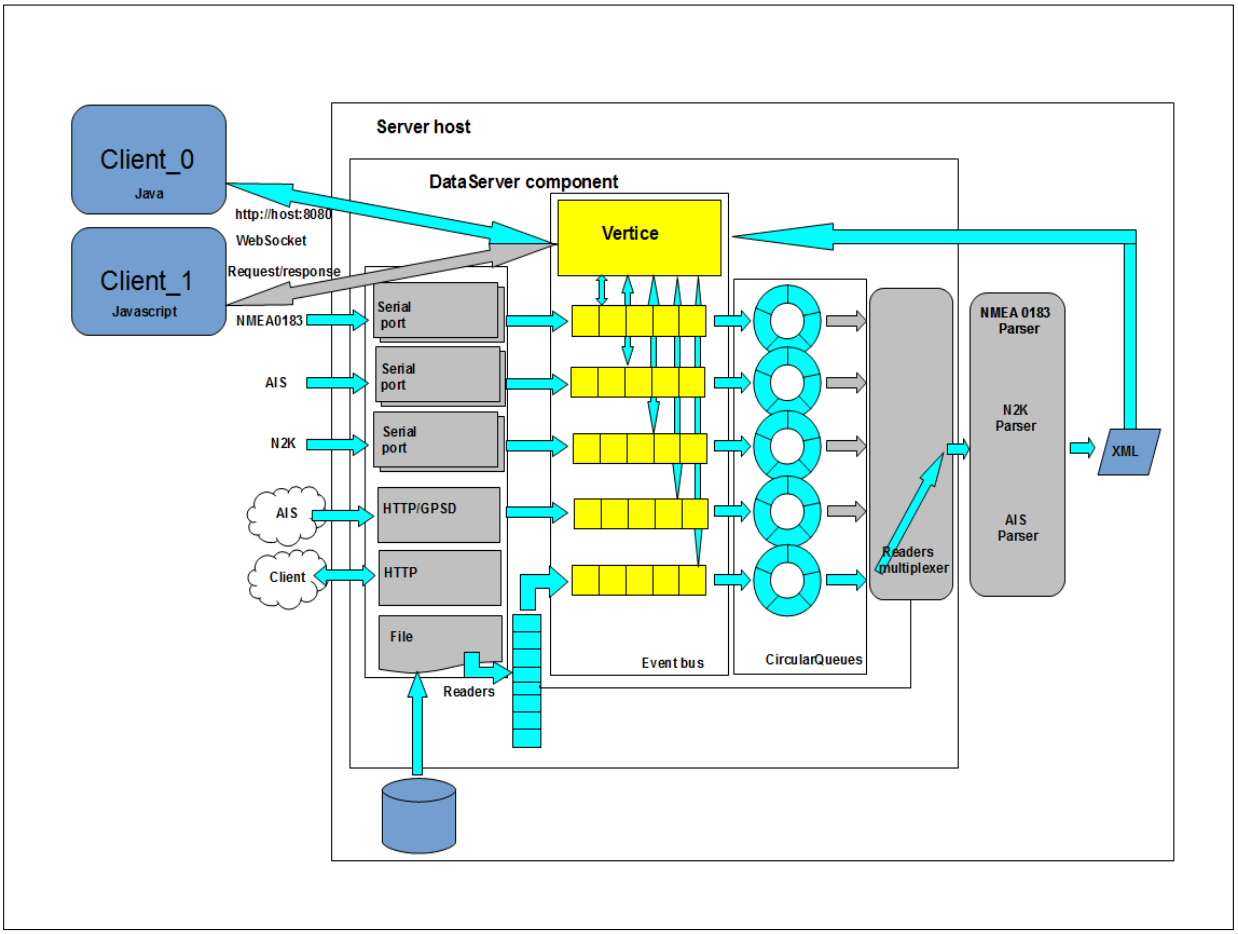

FIGURE 3 – *Exemple de requête client, suivie d'une réponse synchrone à partir de données provenant d'un fichier.*

Les données en sortie (extrait)

Extrait

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
-<sentences>
-<rmc>
  <device>GP</device>
  <sentence>
    $GPRMC,191137.304,A,4826.2620,N,00429.8665,W,000.0,225.3,050114,,,A*7C
  </sentence>
  <eastOrWestVariation/>
  <date>2014-01-05T19:11:37.972+01:00</date>
  <status>A</status>
```

```
  <latitude>48.437702</latitude>
  <longitude>-4.4977746</longitude>
  <variation>0.0</variation>
  <track>225.3</track>
  <sog>0.0</sog>
</rmc>
-<gga>
  <device>GP</device>
  <sentence>
    $GPGGA,191138.000,4826.2632,N,00429.8614,W,2,05,1.2,72.4,M,52.1,M,0.8,0000*50
  </sentence>
  <utc>2014-01-05T19:11:38.973+01:00</utc>
  <latitude>48.43772</latitude>
  <longitude>-4.4976897</longitude>
  <gpsQualityIndicator>2</gpsQualityIndicator>
  <numberOfSatellitesInView>5</numberOfSatellitesInView>
  <horizontalDilutionOfPrecision>1.2</horizontalDilutionOfPrecision>
  <antennaAltitude>72.4</antennaAltitude>
  <unitsOfAntennaAltitude>M</unitsOfAntennaAltitude>
  <geoidAltitude>52.1</geoidAltitude>
  <unitsOfGeoidAltitude>M</unitsOfGeoidAltitude>
  </gga>
</sentences>
```