



**ALLSEEN
ALLIANCE**

Technical Steering Meeting

April 15, 2014

Antitrust Compliance Notice

- AllSeen Alliance meetings involve participation by industry competitors, and it is the intention of AllSeen Alliance to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of and not participate in any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.
- Examples of types of actions that are prohibited at AllSeen Alliance meetings and in connection with AllSeen Alliance activities are described in the AllSeen Alliance Antitrust Policy. If you have questions about these matters, please contact your company counsel, or if you are a member of AllSeen Alliance, feel free to contact Lee Gesmer or Andrew Updegrove, of the firm of Gesmer Updegrove LLP, which provides legal counsel to AllSeen Alliance.

Reminder:

This call is being recorded

Agenda

- Approve minutes from last call
- Infrastructure Repository
 - For information only
- Data-driven API Proposal
 - Presentation from Technicolor
- Wireshark protocol dissector for the AllJoyn framework
 - Brief description and pointers for download

Infrastructure Repository

Infrastructure Repository

- The Tools and Infrastructure team needs to store scripts and configuration files used to implement build automation
- These files need to be shared openly, but are not distributed as part of AllSeen official releases and do not fall under a working group.
- Name: `infra/automation.git`

Data-driven API Proposal

Dominique Chanet

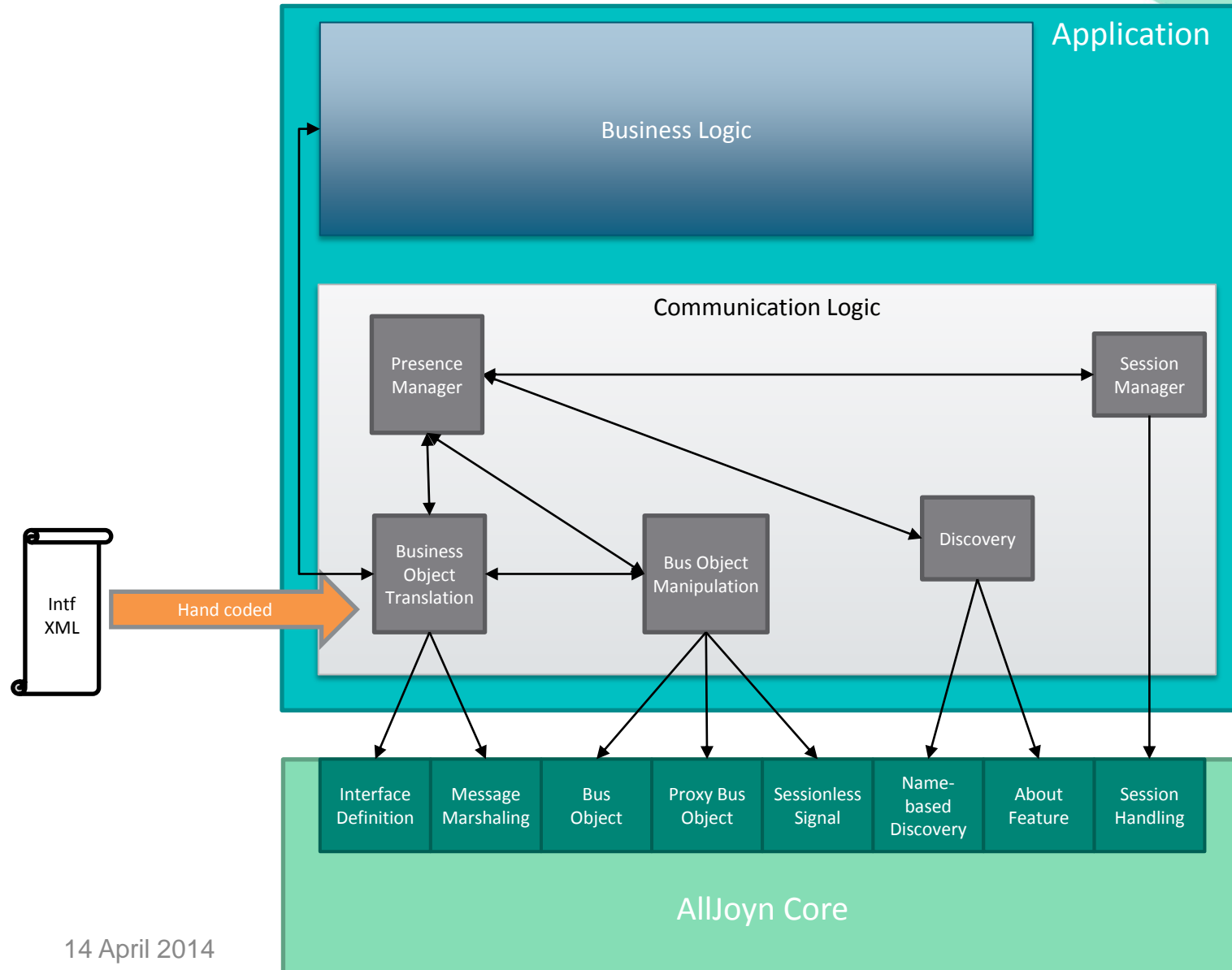
Raise conceptual level of the API

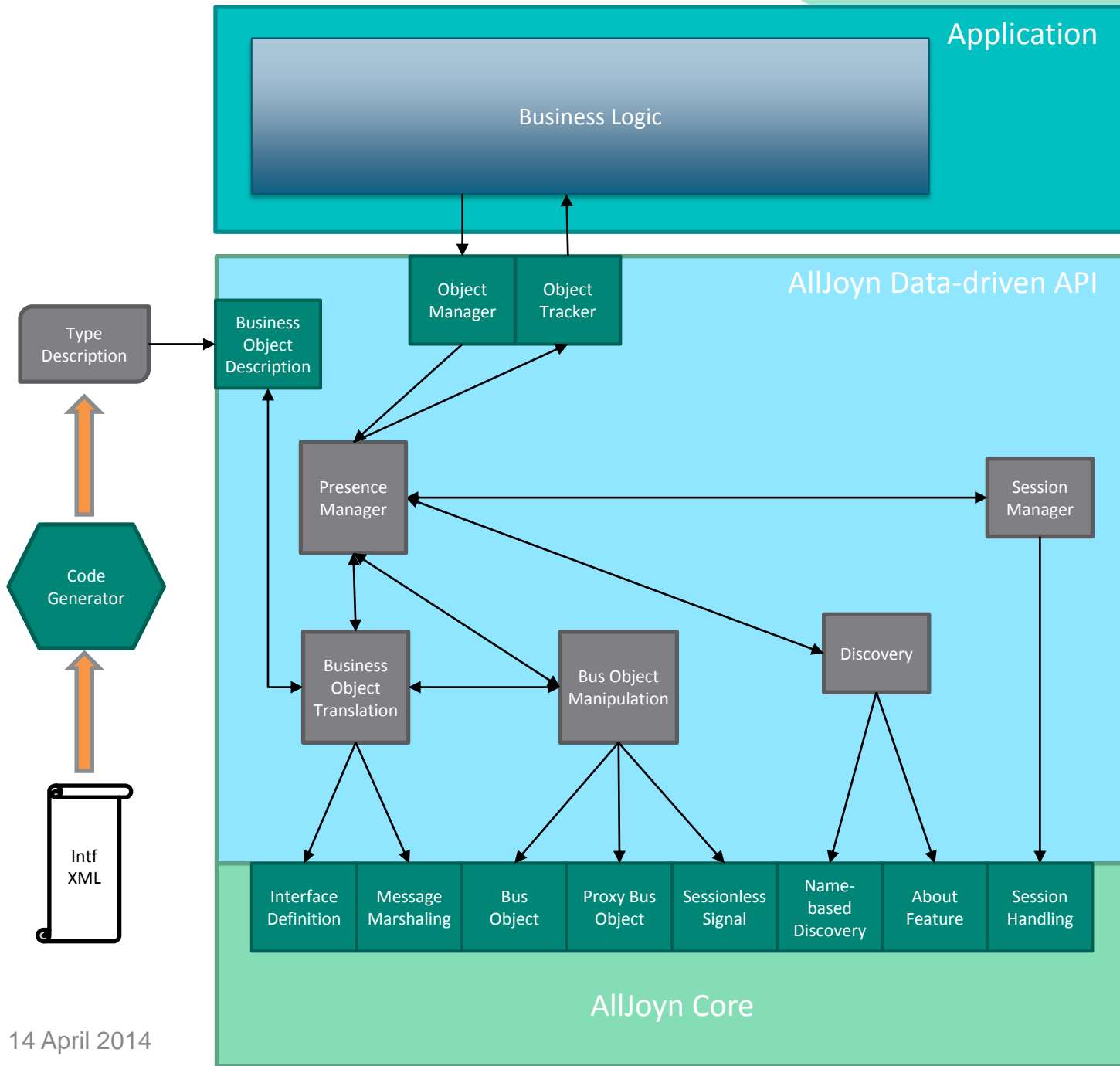
Smaller number of distinct API concepts

reduces learning curve for developers

- One approach for discovery and session setup
better interoperability, easier for developers
- Introduce data-oriented, publish subscribe paradigm
good fit for IoT use cases
- Extend expressiveness of type system
make interfaces more self-describing
- Leverage code generator tooling
less boilerplate code, friendlier C/C++ API

Classic AllJoyn Application





Main Driver

A lot of IoT's potential value lies in devices that respond intelligently to their surroundings

- **data-driven approach**
communicate with others based on the data they offer/need
- **understandable data**
make sense of the data you gather
- **accessible data**
know how to get the data
- **ubiquitous data**
the more inputs, the better

Data-driven approach

Discover peers based on the data they offer

- current discovery is mainly based on who you are (name-based discovery)
- change this to what you offer (interface-based discovery)

Introduce the ability to define interfaces as data-oriented instead of service-oriented

- current AllJoyn interfaces focus on interactions (methods), observable properties are an afterthought
- make observable properties first-class citizens
build publish-subscribe system to observe them

Understandable data

Data is only useful if I can make sense of it

- formalized data model description language
based on existing Introspection XML language
- extensions to type system to make data models more self-describing
named types, enumerations, optional fields
- best practices for data model design
discourage use of variant types, expose state as observable properties, ...
- common repository for standardized data models (out of scope for this proposal)
 - one way to describe temperature, light bulbs, GPS position, ...
 - data-centric take on the AllSeen standardized services

Accessible Data

Data is only useful to me if I can access it

- AllJoyn has a toolkit-based approach to discovery and session setup
 - lots of building blocks, build the system that suits you best out of this
 - great for ad hoc distributed applications
 - reusable distributed components need a standardized, well-understood way of doing discovery and session setup
- The same concept applies to security
 - security is mandatory for IoT
 - need a standardized mechanism for authentication & access control
 - not covered in this proposal, separate efforts underway

Ubiquitous Data

Devices become more useful if they have more data available as input

- AllSeen ecosystem gets more valuable as it expands
- Need to make it easy for developers and device manufacturers to step in
- Simplified API with reduced learning curve can lower barrier to entry

Practical Approach

R14.06

- extensions to type system
 - proposal on Alliance Wiki, discussions on mailing list
 - implementation of named types and enumerations in the code generator
- marshaling improvements for C/C++
 - generic marshaling/unmarshaling code as library
 - code generator to generate type descriptions out of interface XML
 - not full application skeleton as for Thin Client
- prototype of data-driven API
 - Initially just for C++
 - as a separate library on top of AllJoyn Core
 - primary goal: gather feedback from the community

Practical Approach

R14.10

- data-driven API
 - Integrate data-driven C++ API in core taking advantage of e.g. Next-Gen Name Service
 - Add language bindings (Android/C/Thin Client/...)
- extensions to the type system
 - optional fields
 - extend to all language bindings
- extensive documentation
 - best practices for data model definition
 - data-driven API reference guide
 - data-driven API user guide

Wireshark Plugin for the AllJoyn Protocol

Joseph Huffman

What is Wireshark?

It's hard to overstate how useful this tool is.

Craig Dowell, AllJoyn Senior Staff Engineer
April 14, 2014

Wireshark is described as "The World's Most Popular Network Protocol Analyzer".

Wireshark captures and displays network traffic in human readable format with powerful protocol specific filters to make debugging network communication dramatically easier. See <http://www.wireshark.org/> for more information.

An AllJoyn written protocol dissector has just been accepted for inclusion in the next Wireshark stable release, version 1.12. This is scheduled for some time in June.

A developmental release is scheduled for April 15th.

Sample output

The image shows a Wireshark packet capture of a D-Bus message. The filter is set to 'ajns or aj'. The packet list shows a message from 192.168.1.114 to 192.168.1.119. The packet details pane shows the message structure, including a signature and parameters. The packet bytes pane shows the raw data.

Filter: `ajns or aj`

No.	Time	Source	Destination	Protocol	Length	Info
333	7.953029000	fe80::a021:da60:e018:e9f7	224.0.0.113	ALLJOYN-NS	126	VERSION 1 ISAT
339	8.016663000	192.168.1.114	224.0.0.113	ALLJOYN-NS	108	VERSION 1 ISAT
340	8.016730000	fe80::a021:da60:e018:e9f7	ff02::13a	ALLJOYN-NS	128	VERSION 1 ISAT
342	8.104681000	192.168.1.119	192.168.1.114	ALLJOYN	250	Message 0000000021: 'Method call' BusHello (su)
343	8.105076000	192.168.1.114	192.168.1.119	ALLJOYN	202	Message 0000000053: 'Method reply with returned data' Replies to: 0000000021 (ssi
348	8.106609000	192.168.1.119	192.168.1.114	ALLJOYN	370	Message 0000000022: 'signal emission' ExchangeNames (a(sas))
349	8.106843000	192.168.1.114	192.168.1.119	ALLJOYN	393	Message 0000000054: 'signal emission' ExchangeNames (a(sas))
350	8.115821000	192.168.1.119	192.168.1.114	ALLJOYN	438	Message 0000000026: 'Method call' AttachSession (qsssssa{sv})

String Data: :rcBRSb1i.1

- Header field: Signature (0x08)
 - Field code: 0x08
 - 0x01 byte
 - Type id: 'g' => signature
 - 0x00 byte
 - signature length: 6
 - signature: a(sas)
- Parameters
 - Array of 2 '(' elements
 - struct (sas)
 - String Size 32-bit: 11
 - String Data: :rcBRSb1i.1
 - Array of 4 's' elements
 - String Size 32-bit: 20
 - String Data: org.freedesktop.DBus
 - String Size 32-bit: 15
 - String Data: org.alljoyn.Bus
 - String Size 32-bit: 18
 - String Data: org.alljoyn.Daemon
 - String Size 32-bit: 14
 - String Data: org.alljoyn.s1
 - struct (sas)
 - String Size 32-bit: 11
 - String Data: :rcBRSb1i.2
 - Array of 0 's' elements

0060 00 00 00 00 00 02 01 73 00 12 00 00 00 6f 72 s.....or
0070 67 2e 61 6c 6c 6a 6f 79 6e 2e 44 61 65 6d 6f 6e g.alljoy n.Daemon
0080 00 00 00 00 00 03 01 73 00 0d 00 00 00 45 78 s.....Ex
0090 63 68 61 6e 67 65 4e 61 6d 65 73 00 00 00 06 01 changena mes....
00a0 73 00 12 00 00 00 6f 72 67 2e 61 6c 6c 6a 6f 79 s.....or g.alljoy
00b0 6e 2e 44 61 65 6d 6f 6e 00 00 00 00 00 00 07 01 n.Daemon
00c0 73 00 0b 00 00 00 3a 72 63 42 52 53 62 31 69 2e s.....r cBRSb1i.
00d0 31 00 00 00 00 00 08 01 67 00 06 61 28 73 61 73 1..... g..a(sas
00e0 29 00 00 00 00 00 84 00 00 00 00 00 00 00 0b 00)......
00f0 00 00 3a 72 63 42 52 53 62 31 69 2e 31 00 5b 00 ..:rcBRS bli.1.[.
0100 00 00 14 00 00 00 6f 72 67 2e 66 72 65 65 64 65pr g.freede
0110 73 6b 74 6f 70 2e 44 42 75 73 00 00 00 0f 00 sktop.DB us.....
0120 00 00 6f 72 67 2e 61 6c 6c 6a 6f 79 6e 2e 42 75 ..org.al ljoyn.Bu
0130 73 00 12 00 00 00 6f 72 67 2e 61 6c 6c 6a 6f 79 s.....or g.alljoy
0140 6e 2e 44 61 65 6d 6f 6e 00 00 0e 00 00 00 6f 72 n.Daemonor
0150 67 2e 61 6c 6c 6a 6f 79 6e 2e 73 6c 00 00 0b 00 g.alljoy n.s1....
0160 00 00 3a 72 63 42 52 53 62 31 69 2e 32 00 00 00 ..:rcBRS bli.2...
0170 00 00

String Data (alljoyn.string.data), 21 bytes | Packets: 512 - Displa... | Profile: Default

Filters

- To display only AllJoyn traffic type "aj or ajns" in the "Filter:" toolbar and then click "apply".
- In the filter expression window type "alljoyn." to see AllJoyn data types available to filter upon. E.g. applying "alljoyn.string.data" limits the packets shown to only packets that contain that data type.
- Apply the filter "alljoyn.string.data==org.datatypes.test" to see only packets with AllJoyn string data values of "org.datatypes.test".
- Detailed filters down to the bit level cases with a filter like: "alljoyn.header.flags.allowremotemessages==true".

Sample output

The image shows a Wireshark packet capture window titled "PicardLaforgeDataTypes.pcapng [Wireshark 1.11.3-2394-g45143c6 (v1.11.3-rc1-2394-g45143c6 from master)]". The filter bar shows "alljoyn.header.flags.allowremotemessages==true". The packet list shows a single packet (No. 342) at time 8.104681000, source 192.168.1.119, destination 192.168.1.114, protocol ALLJOYN, length 250, and info "Message 0000000021: 'Method call' BusHello (su)".

The packet details pane shows the following structure:

- Frame 342: 250 bytes on wire (2000 bits), 250 bytes captured (2000 bits) on interface 0
- Ethernet II, Src: Quantaco_f9:5b:30 (00:16:36:f9:5b:30), Dst: Asustekc_70:b5:b2 (50:46:5d:70:b5:b2)
- Internet Protocol Version 4, Src: 192.168.1.119 (192.168.1.119), Dst: 192.168.1.114 (192.168.1.114)
- Transmission Control Protocol, Src Port: 50876 (50876), Dst Port: alljoyn-stm (9955), Seq: 25, Ack: 38, Len: 196
- Alljoyn Message Protocol
 - Message Header: 6c0104012c0000001500000088000000
 - Endianness: Little endian (108)
 - Message type: Method call (1)
 - Flags: 0x04
 - 0... = Encrypted: False
 - .0... = Compressed: False
 - ..0... = Allow global broadcast: False
 - ...0... .. = Sessionless: False
 -1... = Allow remote messages: True
 -0. = No auto start: False
 -0. = No reply expected: False
 - Major version: 1
 - Body length: 44
 - Serial number: 21
 - Header length: 136
 - Header fields: 01016f00100000002f6f72672f616c6c6a6f796e2f427573...
 - Header field: Object Path (0x01)
 - Field code: 0x01
 - 0x01 byte
 - Type id: 'o' => object path
 - 0x00 byte
 - unsigned int32: 16

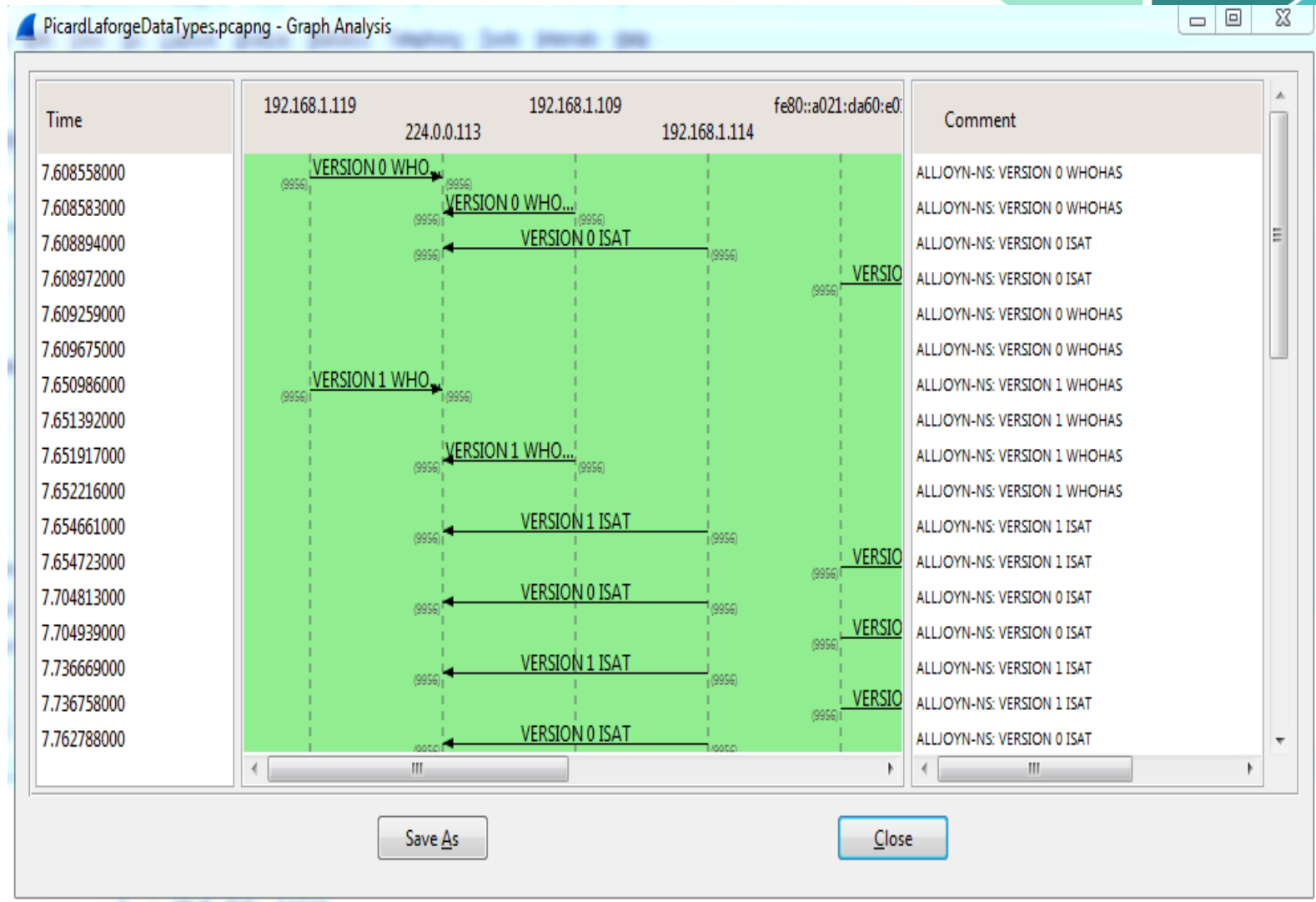
The packet bytes pane shows the raw data in hexadecimal and ASCII. The ASCII column shows the message body: "6.[0..E..7.@...=...W..r.&.Xn..8..P.@.rw..l..@,.....o...../o rg/alljo yn/Bus..S.....or g.alljoy n.Bus..S.....Bu sHello..S.....or g.alljoy n.Bus..S.....:r cBRsbl..1.....g..su..f5664b 5b9c2581 2ca119b3 dca78a56 42.....".

The status bar at the bottom shows "Allow remote messages (alljoyn.header.flags...", "Packets: 512 ...", and "Profile: Default".

Flow Graphs

To see a sequence diagram click on menu item "Statistics", click on "Flow Graph", then click "Okay" on the dialog box that comes up.

Sample output



Report bugs

If you encounter bugs please file them here:

<https://bugs.wireshark.org/bugzilla/>

You can also download the source to fix bugs and/or add features by getting involved in the Wireshark project:

<http://www.wireshark.org/develop.html>

Thank You.