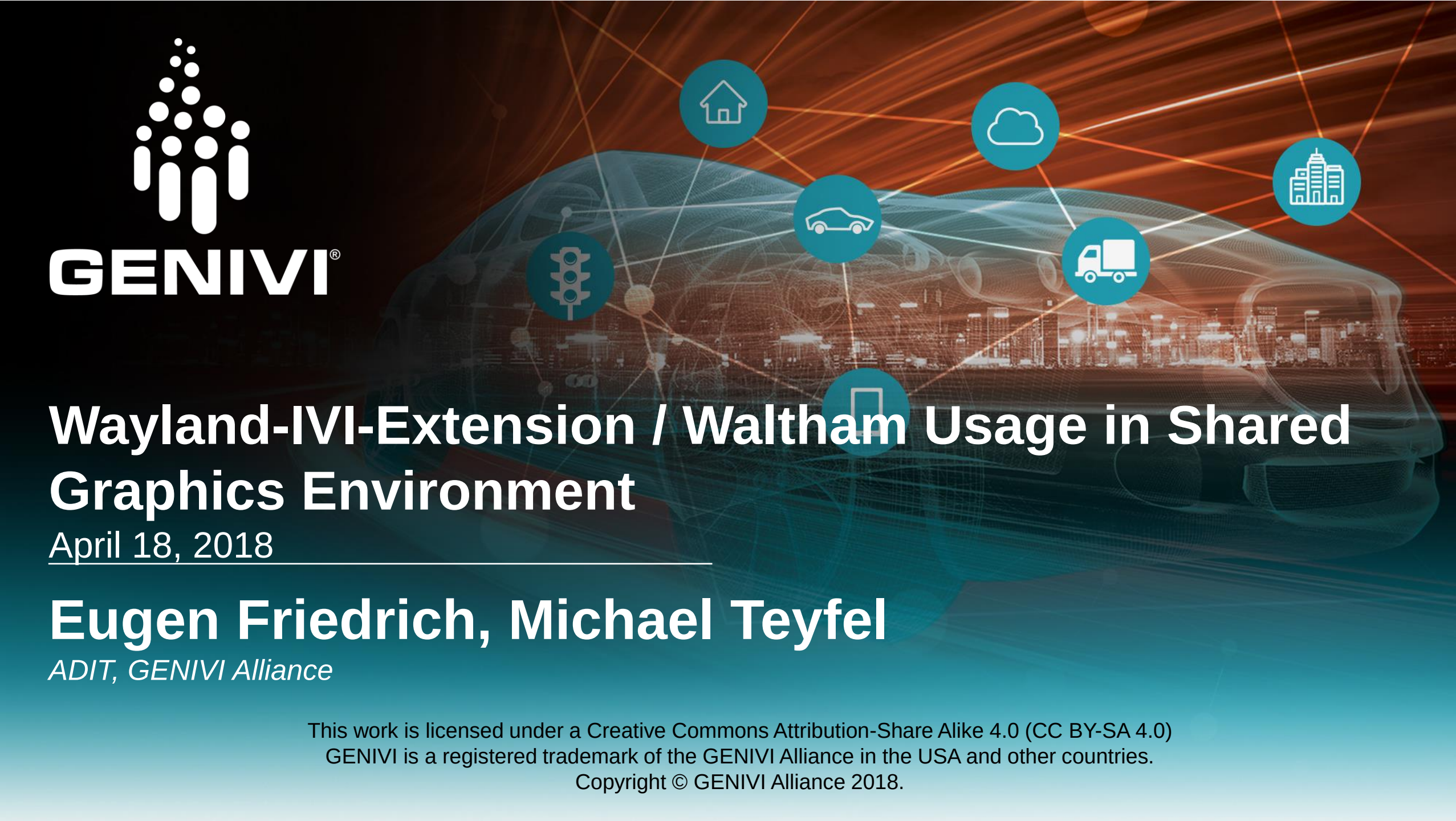


Wayland-IVI-Extension / Waltham Usage in Shared Graphics Environment

April 18, 2018

Eugen Friedrich, Michael Teyfel

ADIT, GENIVI Alliance

Introduction

- Advanced Driver Information Technology GmbH (short: ADIT)
  - Joint Venture between BOSCH and DENSO
  - Platform Development for IVI-Systems
  - OSS Expertise, Genivi, etc.
- Eugen Friedrich
  - Staff Graphics Engineer
  - efriedrich@de.adit-jv.com
- Michael Teyfel
  - Graphics Engineer
  - mteyfel@de.adit-jv.com

GENIVI®

Outline

- Modern HMIs in IVI
- Graphics Sharing within GENIVI
- Generic Use-Cases of Distributed HMI Interaction
  - Display Sharing [Display Sharing]
  - Sharing of Already Rendered Content [Waltham]
  - Sharing Metadata to Be Able to Render Content [Ramses]
- Live Demo and Source Code Walkthrough [Waltham]

GENIVI®

Modern HMIs in IVI

- Multiple displays
- Multiple ECUs
- External content:
  - Smartphone
  - Cloud
- Seamless experience and common user interface
- Several opposing requirements need to be resolved
- New technologies and concepts are required to achieve this goal

GENIVI®

Graphics Sharing within GENIVI

- GPU Sharing
- Display Sharing
- Surface Sharing
- API Remoting
- Shared State, Independent Rendering

GENIVI®

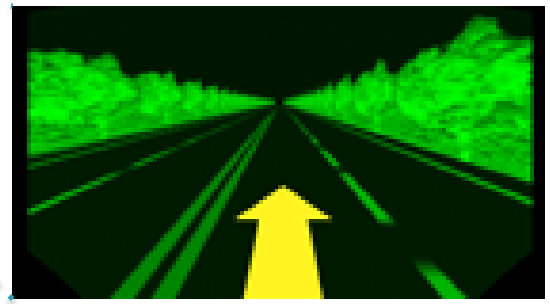
Generic Use-Cases of Distributed HMI Interaction

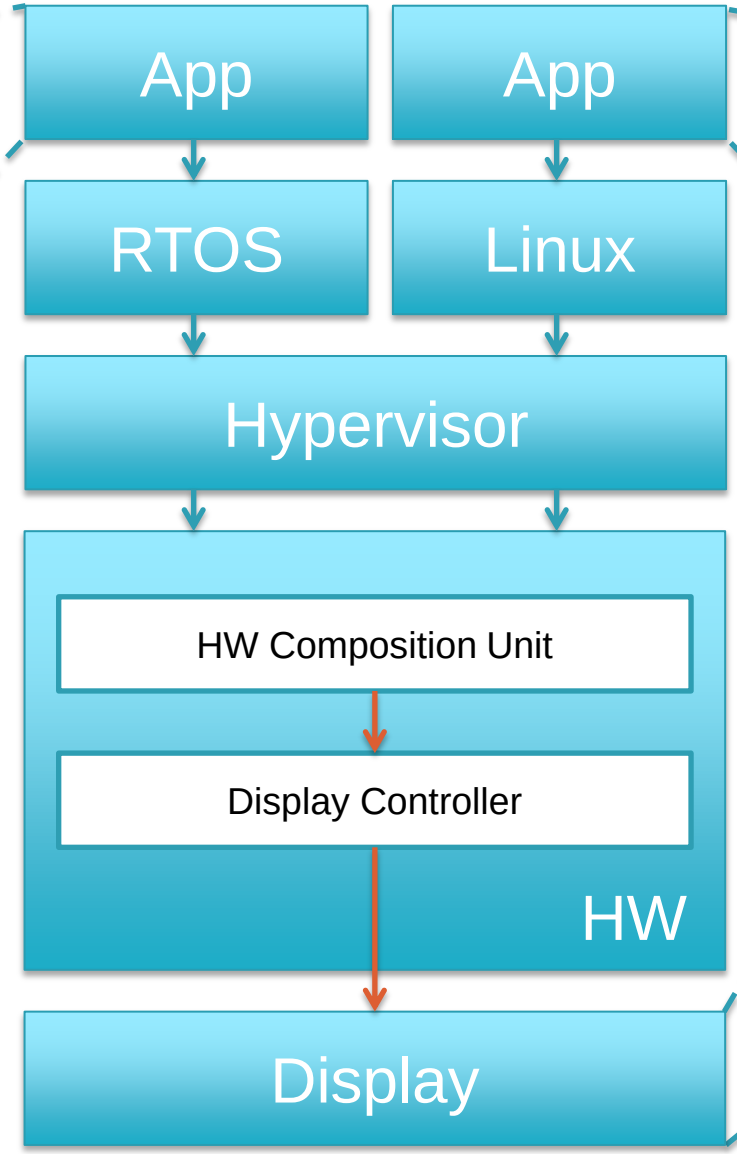
Display Sharing

- A physical display can be shared across multiple operating systems
- HW-compositor-unit composites final display buffer from HW layers of each OS
- Can be realized in virtualized environments
- Implementation can be done in corresponding display drivers
- Support of hardware or hypervisor may be required to share the hw-compositor-unit

GENIVI®

Display Sharing





Each VM has access to limited display resources, i.e displays and layers

Hypervisor has access to all resources

HW compositon unit creates the final framebuffer

Display Sharing

- Pros
  - Sharing is implemented in lower-level software: display driver
  - Upper layer of software are not affected and don’t need any modification
- Cons
  - Requires virtualization environment or specialized hardware
  - Interaction and synchronization between content from different units is difficult to achieve

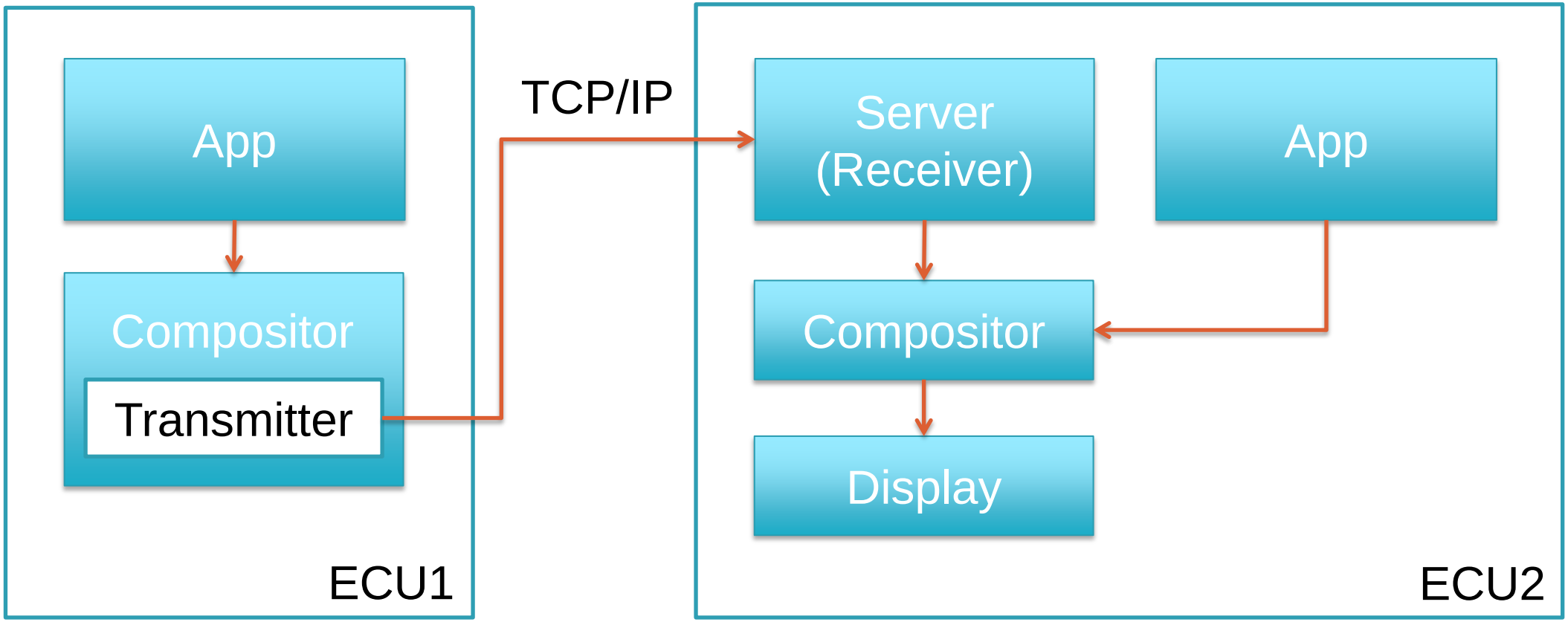
GENIVI®

Sharing of Already Rendered Content

- Operating systems exchange graphical (bitmap) content and each OS has full flexibility to use this content
- Sharing should be implemented on system compositor level
- Exemplary implementation: Wayland / Weston / Waltham Company (Demo)

GENIVI®

Sharing of Already Rendered Content



GENIVI®

Sharing of Already Rendered Content

- Pros
  - Interaction between content from different units is possible to a quite good extend without modification in the applications
- Cons
  - Depending on the system implementations for several system compositors are required
  - Stable network connection between the units is required
  - In case of virtualization shareable graphic memory could be required

GENIVI®

Sharing Metadata to Be Able to Render Content

- Sharing in implemented on the rendering API level, also know as API remoting
  - Remoting the well know OpenGL ES API would keep the application code untouched
    - But has some inherit limitation in term of performance and interactions between different remote streams
  - Introducing new API requires quite big modifications in the application but can solve limitations of the OpenGL ES API remoting and provide new features: RAMSES
- Stable network connection is required
  - With API remoting recovering from the network issues is difficult
    - Frame drop or even restarting of connection could be a consequence

GENIVI®

Sharing Metadata to Be Able to Render Content

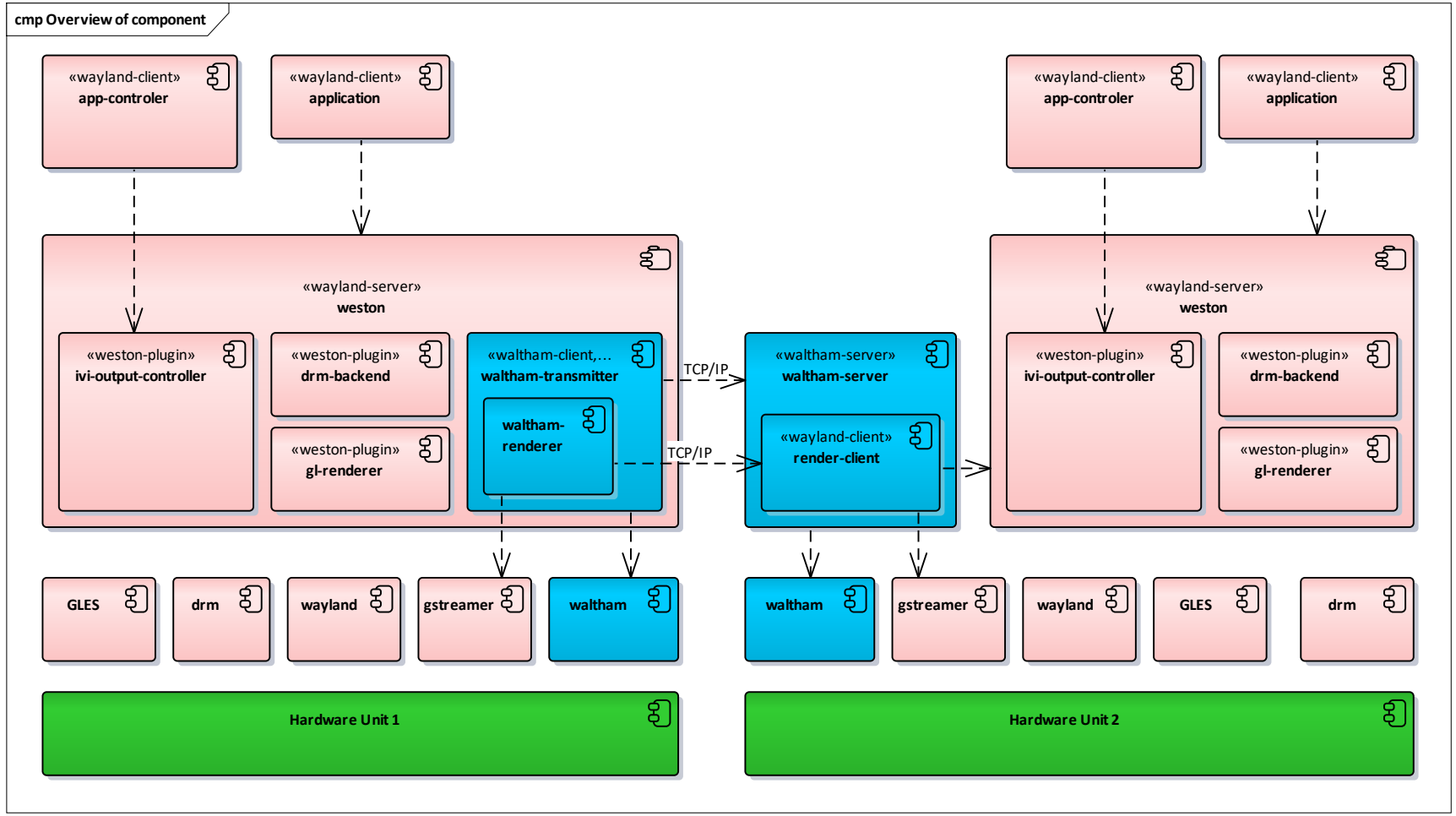
- Pros
  - Implementation of seamless and integrated user experience is possible to a very high degree
- Cons
  - Modifications up to the application level could be required
  - Effort in the design face of the system can be quite high
  - Every receiver of stream requires a rendering hardware

GENIVI®

Live Demo and Source Code Walkthrough (Waltham)

Wayland / Weston / Waltham Company

Hardware unit 1 is sharing content with hardware unit 2



Details of Waltham Related Components

- Waltham
  - A library which implements the communication between Waltham client and Waltham server
- Waltham Transmitter
  - A Waltham client that is implemented as a Weston plugin
  - Has direct access to the application's buffer
  - Uses additional plugin (Waltham renderer) to transmit the buffer to the Waltham server, in the current implementation by using gstreamer
  - Creates an additional Wayland output so app-controller can just add the layer or surface to this output and remoting will be started

GENIVI®

Details of Waltham Related Components

- Waltham Server
  - A component that handles the connections from Waltham clients and receives the buffers
  - Also responsible to provide the buffer to the system compositor

GENIVI®

Live Demo and Source Code Walkthrough

*

TCP/IP

*

Waltham Transmitter
Renesas R-Car Starterkit M3
IP 192.168.2.51
EGLWLMockNavigation is running

Waltham Server (Receiver)
Renesas R-Car Starterkit M3
IP 192.168.2.52

* https://www.digikey.com/catalog/en/partgroup/r-car-starter-kit-pro/70358

GENIVI®

Thank you!

Visit GENIVI at http://www.genivi.org or http://projects.genivi.org

Contact us: genivi-projects@lists.genivi.org

efriedrich@de.adit-jv.com

mteyfel@de.adit-jv.com

GENIVI®