

From Separated ECUs to a Display Cluster

Oct 11, 2018

Violin Yanev, Vaclav Kyba
BMW Group

Challenge

- Multiple displays
 - Different hardware, different companies
 - Seamless integration of content
- Content not fixed to one display



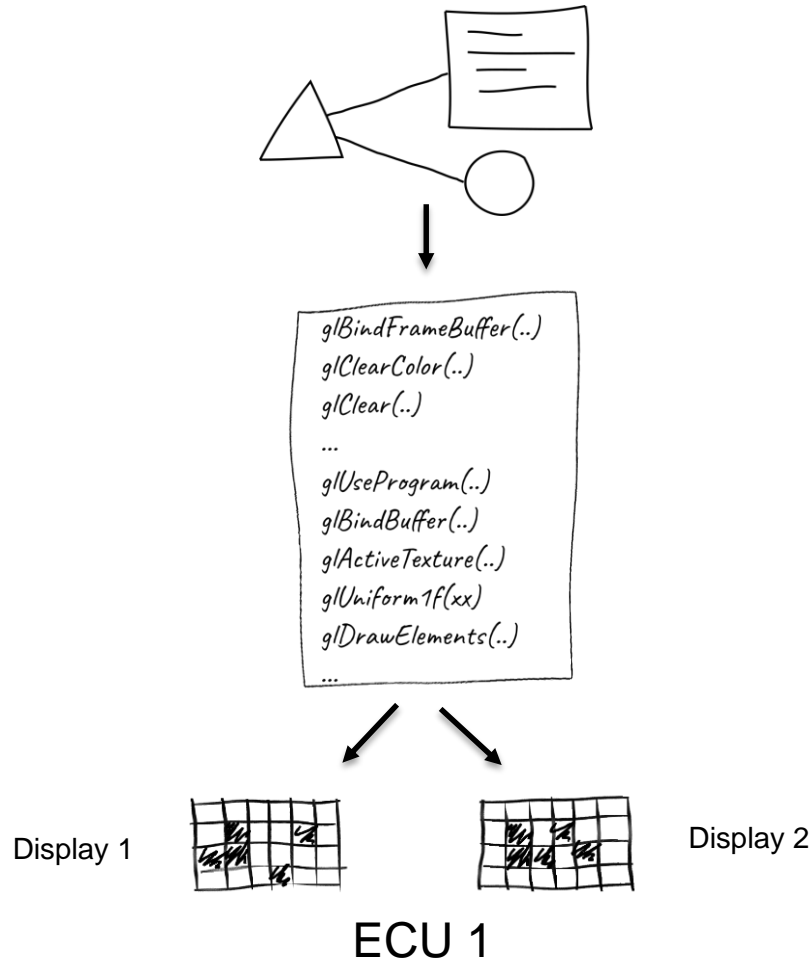
Agenda

1. Challenge
2. Solution Ideas
3. RAMSES Concepts & Features
4. Live Demonstration
5. Hands-on workshop

Solution ideas

Solution ideas:

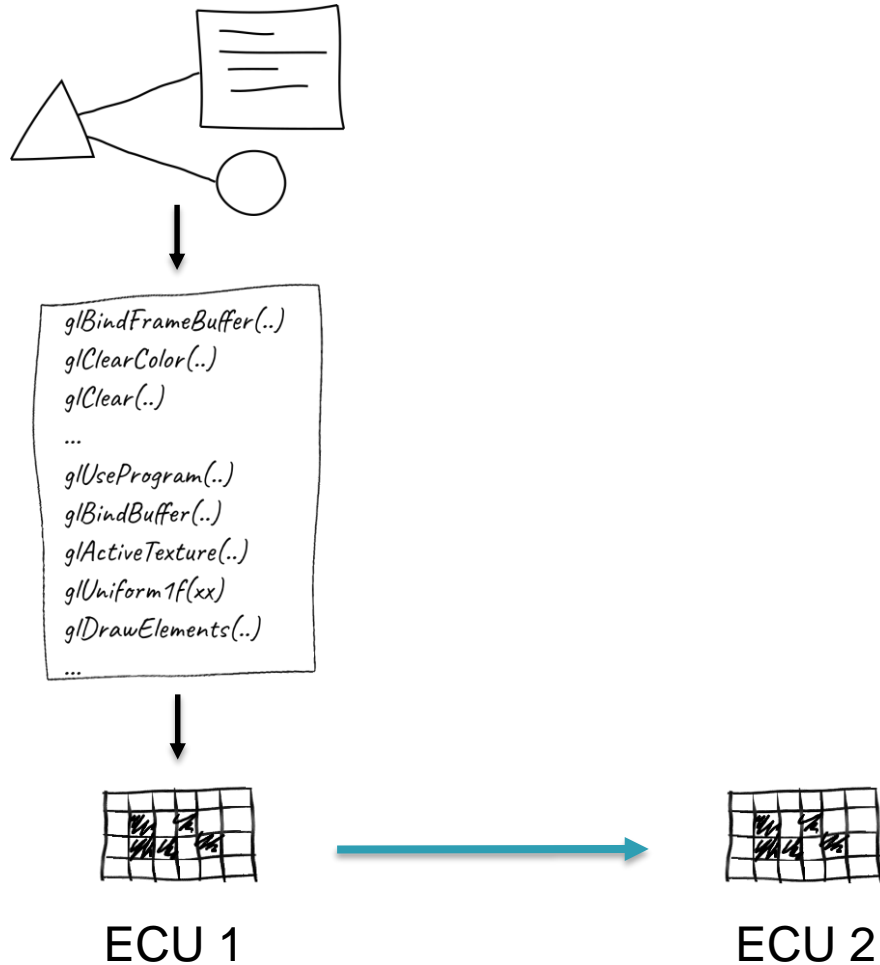
1. One ECU with multiple displays



- + No solution for distribution necessary
- + No network issues
- High computation power needed, scaling to more displays problematic
- Interaction between content from different processes limited

Solution ideas:

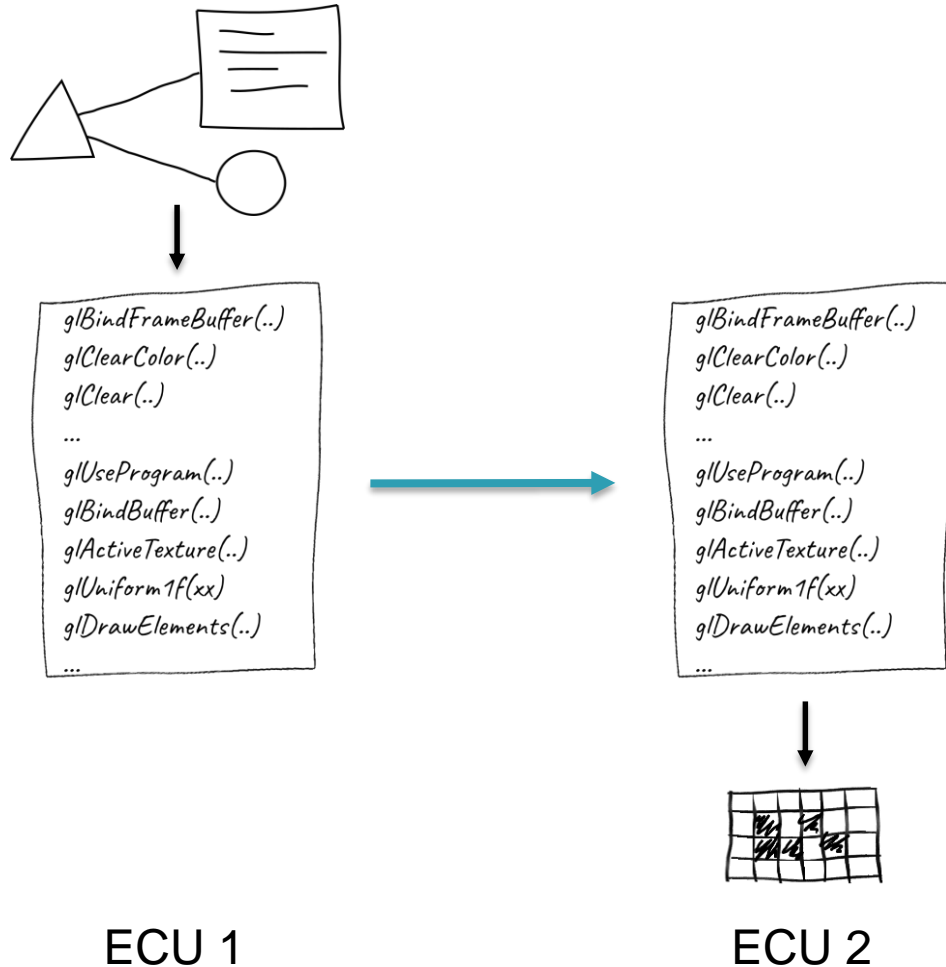
2. Video distribution



- + Easy integration of existing applications
- High computation power needed
- High bandwidth requirements
- Compression artifacts possible
- Availability of hardware encoders and decoders can limit deployment
- Interaction between content from different sources limited

Solution ideas:

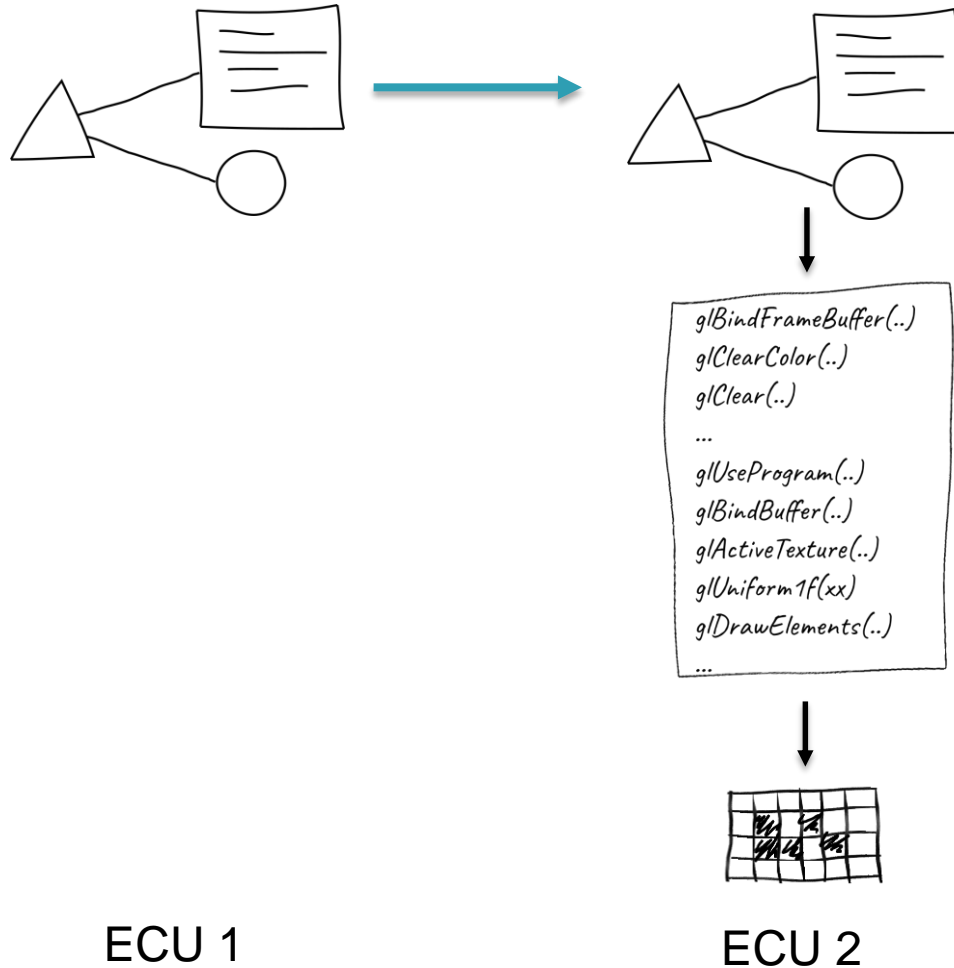
3. OpenGL commands streaming



- + Easy integration of OpenGL-based applications
- + No compression artifacts
- + Easier scaling to higher resolutions
- + No GPU needed on sending side
- Limited to OpenGL-based applications
- Medium bandwidth requirements (full description for each single frame has to be transferred)
- Platform-dependencies with receiving side
- Interaction between content from different sources complex

Solution ideas:

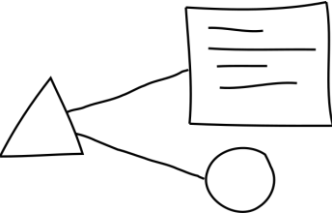
4. Scene-based distribution



- + Low network bandwidth needed especially after initial transfer
 - + No compression artifacts
 - + Easier scaling to higher resolutions
 - + No GPU needed on sending side
 - + Graphical interaction possible between scenes from different ECUs
- Application has to provide content with special API

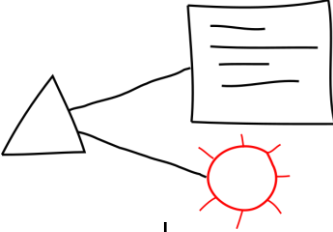
Update of frames: Video distribution

Frame A

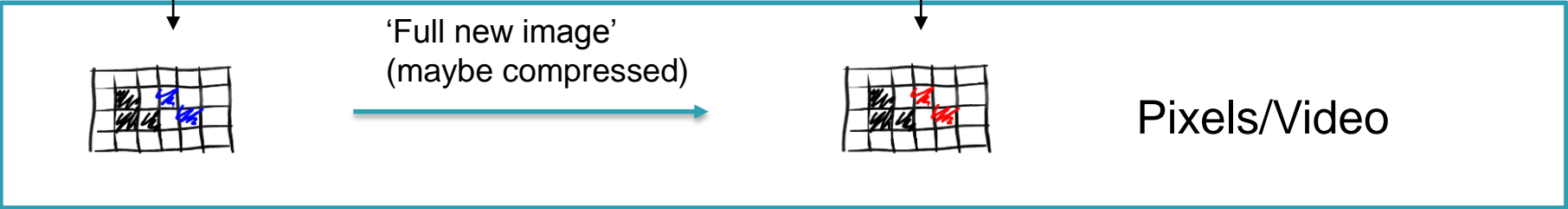


```
glBindFramebuffer(..)  
glClearColor(..)  
glClear(..)  
...  
glUseProgram(..)  
glBindBuffer(..)  
glActiveTexture(..)  
glUniform1f(xx)  
glDrawElements(..)  
...
```

Frame B

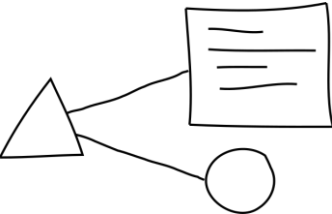


```
glBindFramebuffer(..)  
glClearColor(..)  
glClear(..)  
...  
glUseProgram(..)  
glBindBuffer(..)  
glActiveTexture(..)  
glUniform1f(yy)  
glDrawElements(..)  
...
```

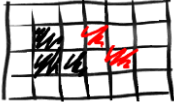
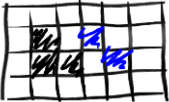
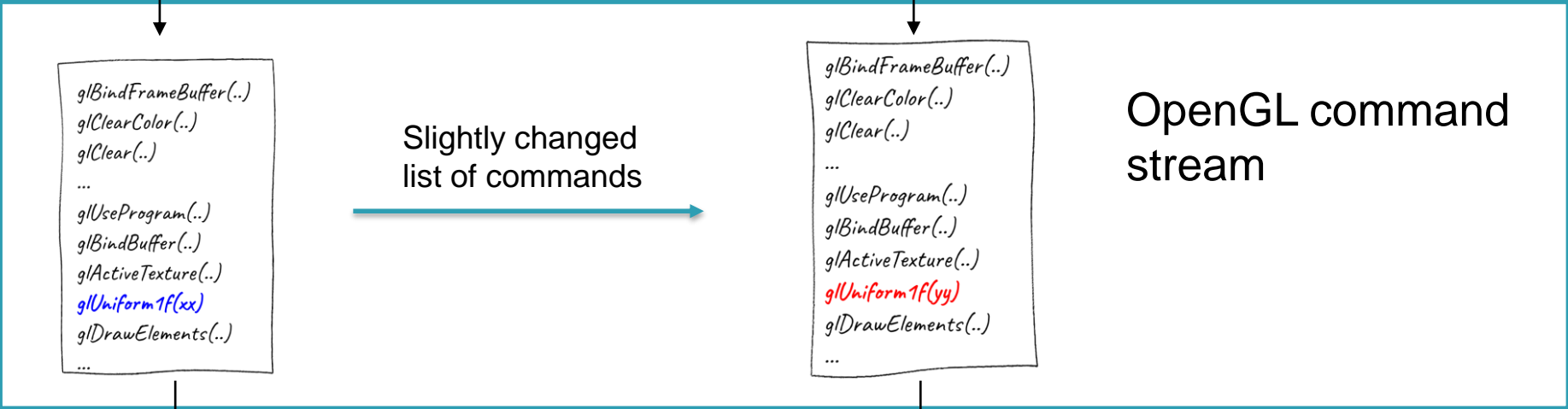
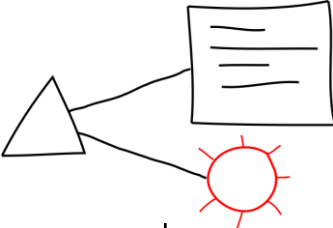


Update of frames: OpenGL commands streaming

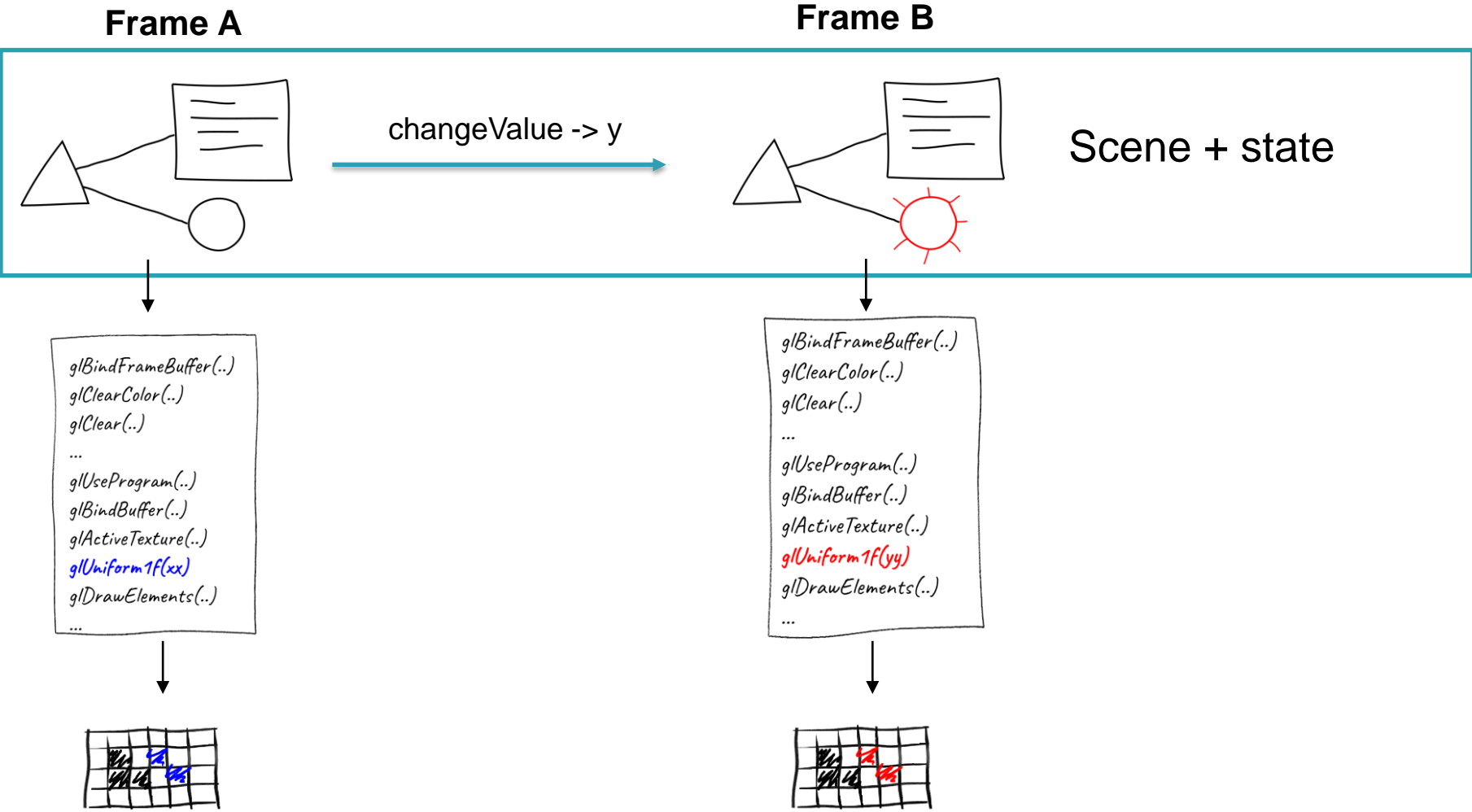
Frame A



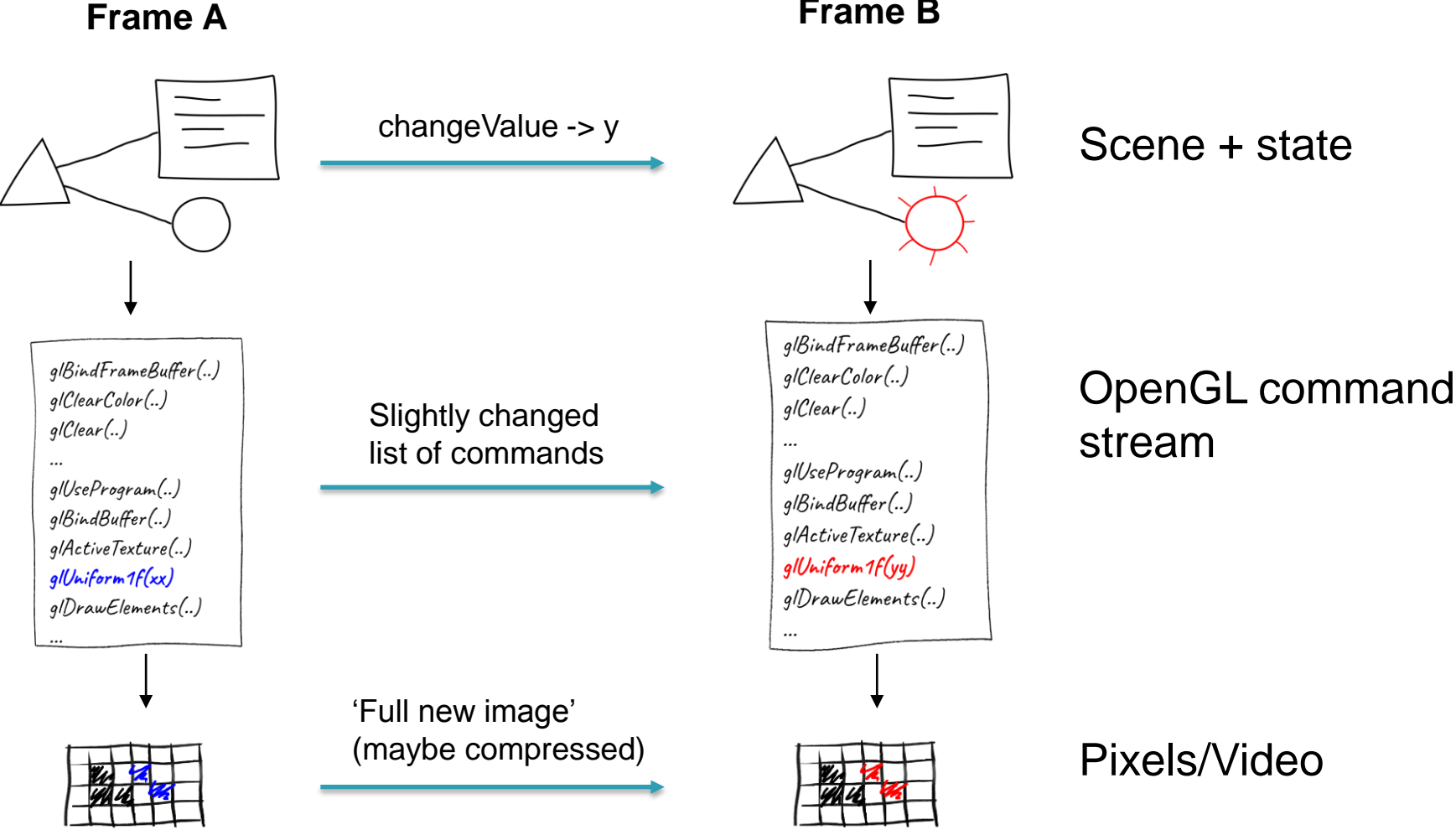
Frame B



Update of frames: Scene-based distribution

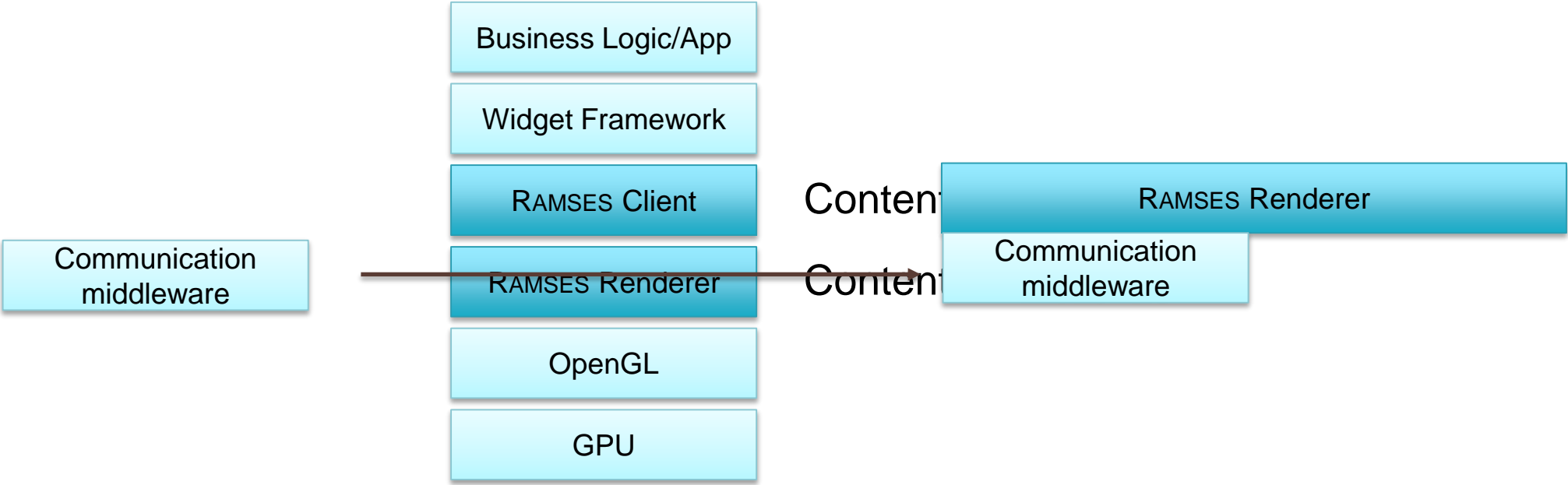


Update of frames



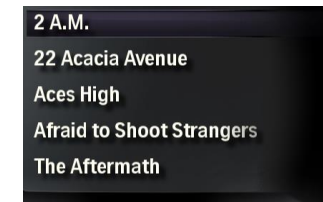
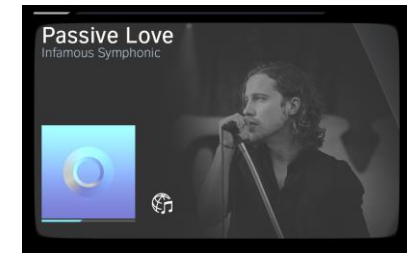
RAMSES Concepts & Features

RAMSES Software Stack



RAMSES scenes

- RAMSES works with scenes
- A scene == content which belongs together
- For example, a radio application could have two scenes:
 - Scene which has the radio's own UI
 - Scene which shows the list of all songs (targeted for display on different ECU)

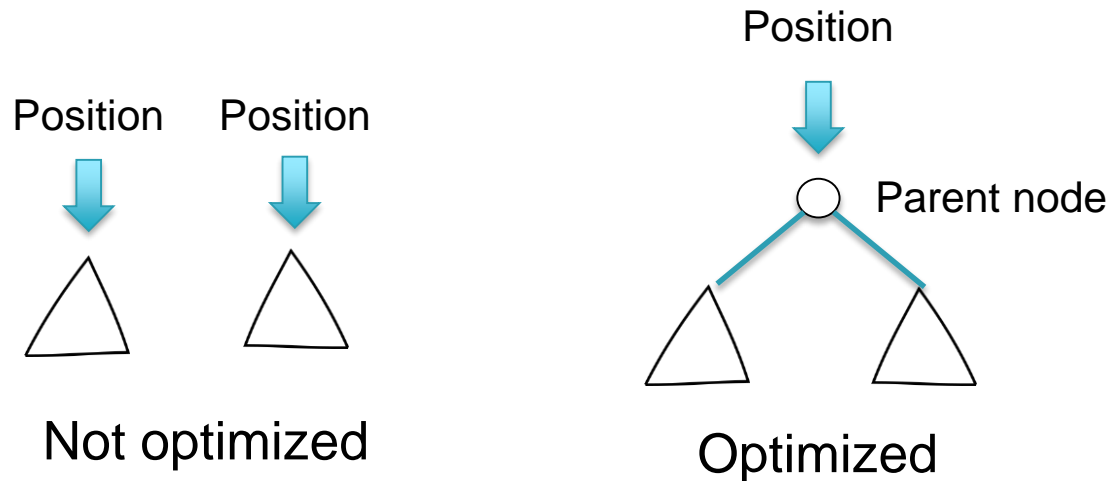


RAMSES scenes compared to OpenGL

- Converting OpenGL to RAMSES content is mostly easy
- Most OpenGL constructs have a RAMSES counterpart, e.g.:
 - `glDrawElements()` ~ `ramses::MeshNode`
 - `glCreateProgram() + glCompileShader()` ~ `ramses::Effect`
 - `glBindFramebuffer()` ~ `ramses::RenderTarget`
- Difference:
 - OpenGL's frame is continuously “recreated” – even with small changes
 - RAMSES objects lifecycle is not per-frame
 - Selective changes possible, can change individual objects or groups

RAMSES scenes compared to OpenGL

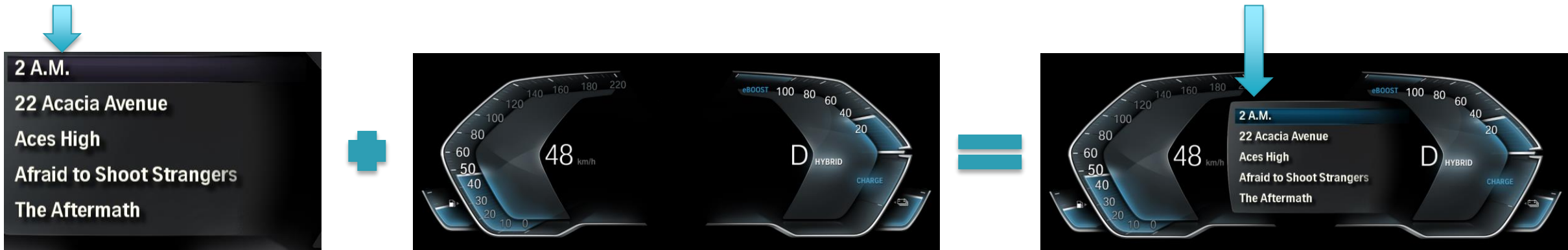
- RAMSES offers additional features on top of OpenGL that help to reduce data bandwidth
- For example, have a scene graph instead of list of draw commands:



- Such optimizations benefit remote **and** local scenes

Interaction between scenes

- Independent scenes can exchange data via RAMSES
 - Any “uniform” or “constant” data – colors, animated values, etc.
 - Textures
 - Positions
- Example with color:



Interaction between scenes

- Independent scenes can exchange data via RAMSES
 - Any “uniform” or “constant” data – colors, animated values, etc.
 - Textures
 - Positions
- Example with color:



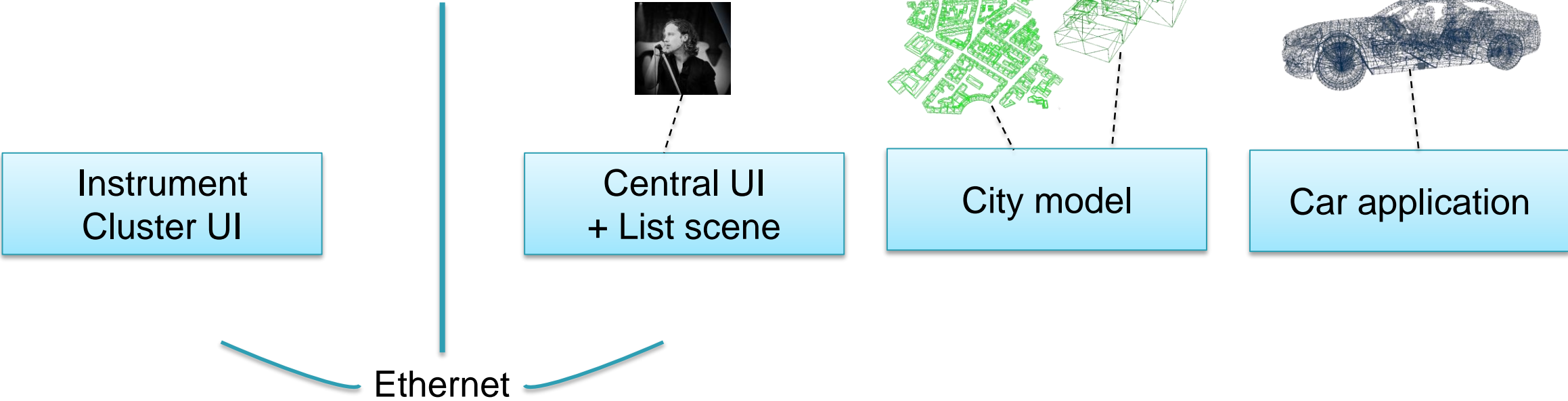
Further features

- Cross platform:
 - Windows, Linux, Integrity OS
 - Wayland, X11, WGL, Integrity OS window system
 - Desktop OpenGL (4.2, 4.5)
 - Embedded OpenGL (ES 3.0+)
 - Clang, GCC, MSVC, Integrity OS compiler
- Wayland support with nested compositing
- Text rendering
- Animations
- Content authoring tool: RAMSES Studio

Live Demonstration



Demonstrator setup



PC 1 (Linux)



PC 2 (Windows)

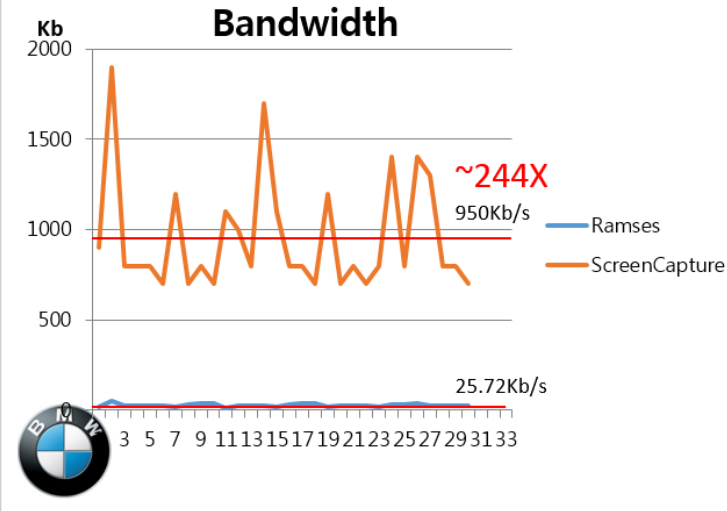
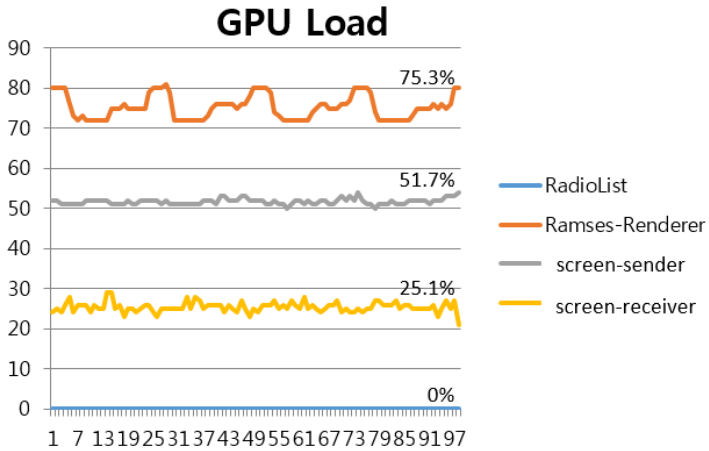
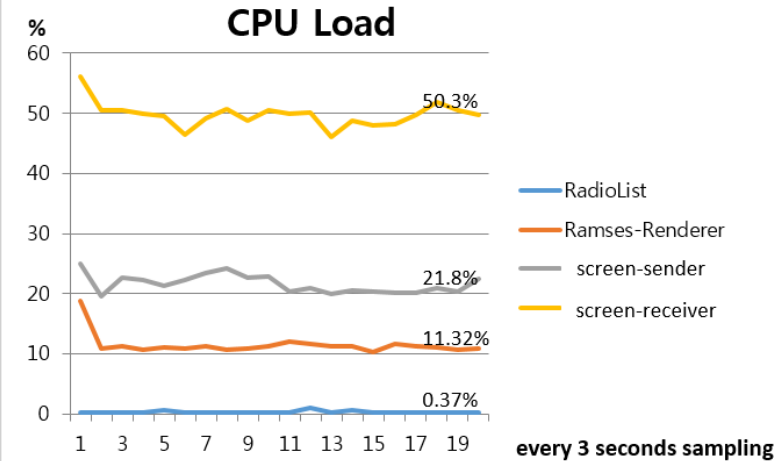


- All code/rendering is live with RAMSES
- Each application is own process

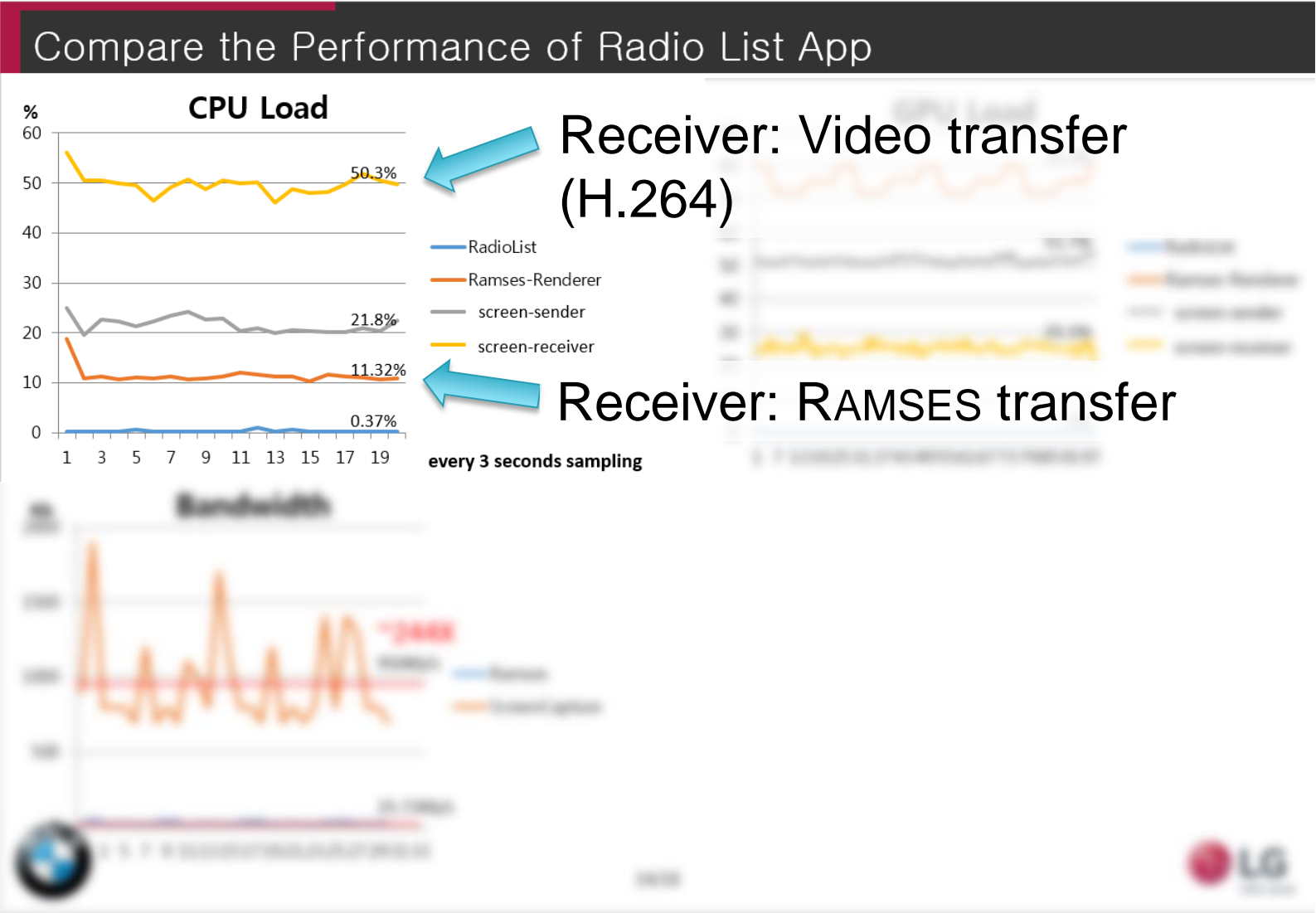


Benchmarks by LG Electronics

Compare the Performance of Radio List App

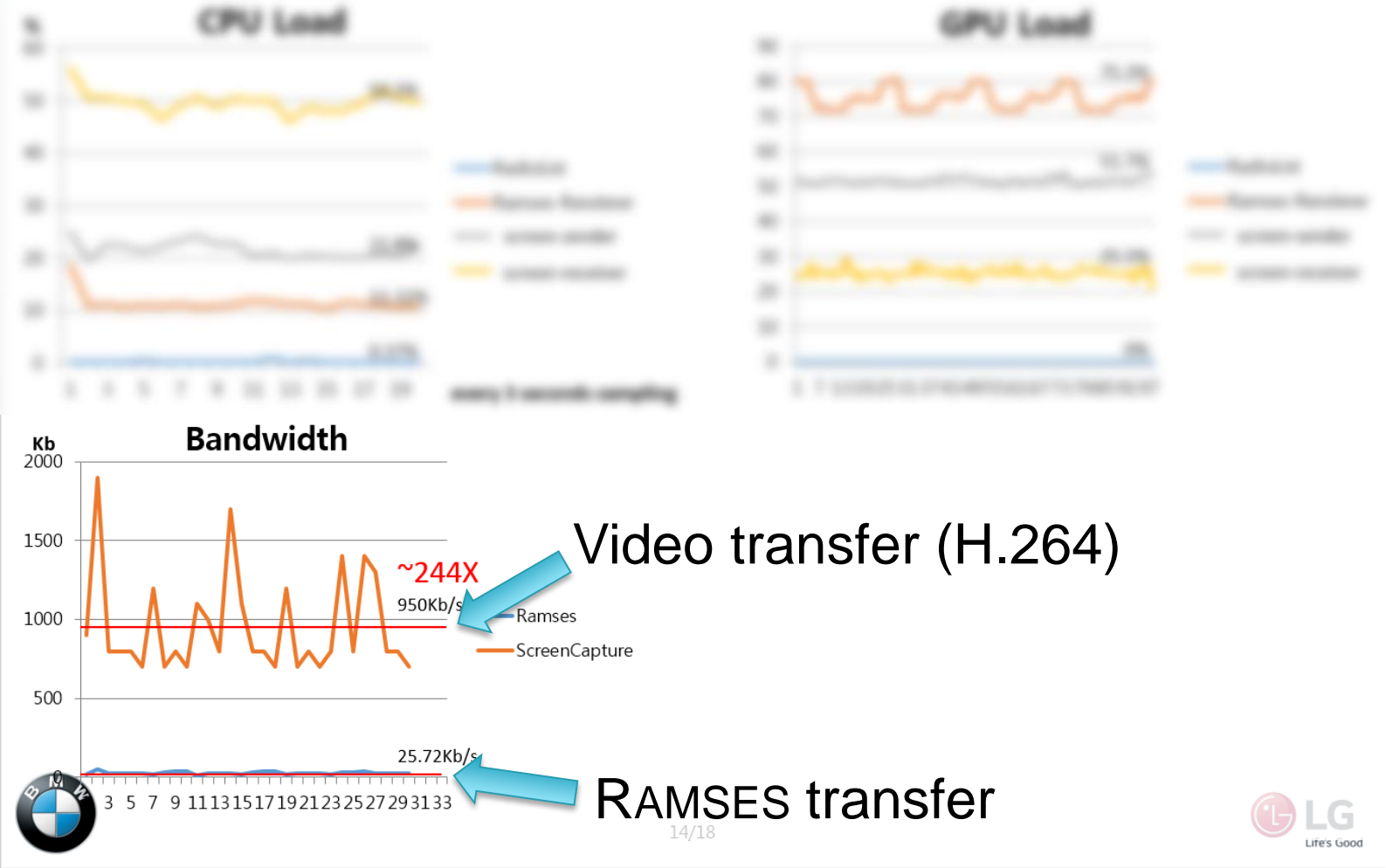


Benchmarks by LG Electronics



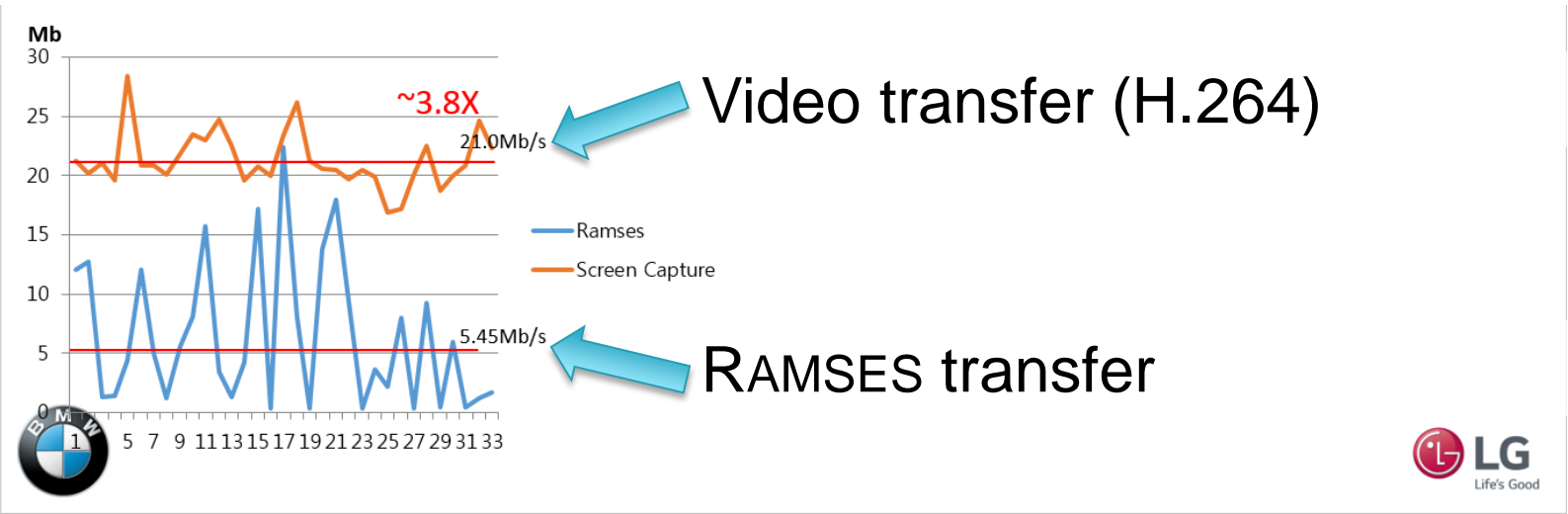
Benchmarks by LG Electronics

Compare the Performance of Radio List App



Benchmarks by LG Electronics (2)

Compare the Performance of Navigation App



Wrap-up

- Applications (or underlying Widget framework) must be adapted to use RAMSES API
- + More interaction of content than video allows seamless UI
- + Graphical flexibility
- + Low bandwidth

Questions?

Thank you!

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

Contact us: help@genivi.org

Contact the RAMSES team: ramses@genivi.org

Picture sources:

<https://www.press.bmwgroup.com/deutschland/photo/compilation/T0247965DE/bmw-auf-der-consumer-electronics-show-ces-2016-in-las-vegas>

<https://www.press.bmwgroup.com/deutschland/photo/compilation/T0274140DE/bmw-concept-x7-iperformance-eine-neue-grosszuegigkeit>

GENIVI is a registered trademark of the GENIVI Alliance in the USA and other countries.

Copyright © GENIVI Alliance 2018.

