

# 스마트 디바이스를 위한 Wayland window system 과 X window system에 관한 비교 연구

김민정\*, 이광림\*, 최진희\*  
 \*삼성전자 소프트웨어 센터  
 e-mail : minjjj.kim@samsung.com

## A comparative Study of Wayland window system and X window system for Smart Device

Min-Jeong Kim\*, Gwang-Lim Lee\*, Jinhee Choi\*  
 \*Software R&D Center, Samsung Electronics

### 요약

Wayland은 open source community에서 30년 가까이 사용되어온 X window system을 대체하기 위해 개발된 새로운 window system이다. X window system은 network transparent한 특성을 기반으로 여러 영역에서 사용되어 왔지만 단일 기기에서의 UX에 필수적인 rendering, event processing, 그리고 compositing 등의 특성에 구조적으로 최적화되어 있지 않다는 문제가 있다. 이 논문에서는 Tizen에 적용된 case를 통해 X window system과 Wayland의 구조적인 장단점을 비교하고 실측 데이터를 통해 구조적 차이로 인한 성능 차이를 설명한다.

### 1. 서론

Window system은 윈도우, 아이콘, 메뉴, 포인터와 같은 시각적 구성요소를 구현해 사용자에게 GUI(Graphical User Interface)를 제공하는 시스템으로 display server, GUI client 그리고 display server protocol로 구성된다. 이중 display server는 운영체제와 GUI client 간의 사용자 입력 이벤트와 화면 출력 요청을 처리하는 window system의 핵심 요소다.

1984년부터 개발되어온 X window system은 여러 운영체제에서 광범위하게 사용되고 있는 대표적인 open source window system이다. X window system의 display server인 X server는 network-transparent protocol을 통해 X client와 통신하는 client-server 모델로 동작한다. X server는 X client의 요청들에 대한 수행 연산과 window system 자원 관리 기능을 제공하고, X client에서 생성한 window들의 위치, 크기 및 stack 순서 결정에 대한 window 관리 정책은 별도의 X window manager에서 수행하는 모델을 갖고 있다. Window 관리 정책과 메커니즘을 분리시킨 디자인은 각 단말에 적합한 window manager를 X server 변경 없이 적용 가능하게 해준다. 2004년 추가된 composite 확장 기능은 window manager가 application window를 대상으로 다양한 화면 효과 및 유려한 UX를 제공해 줄 수 있게 해주었다. Figure 1은 X server, X window manager, X client 간의 S/W 계층 구조를 보여준다.

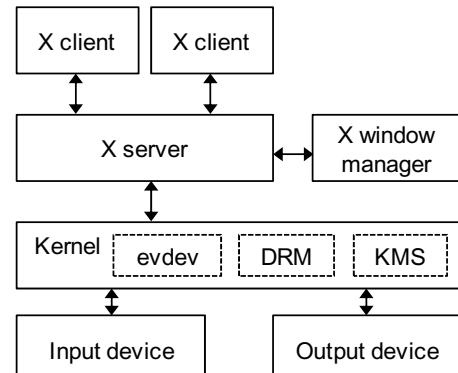


Figure 1. X window system 구조

### 2. Wayland window system

Wayland window system은 현대적인 그래픽스 하드웨어에 적합하지 않은 X window system의 단점을 보완하기 위해 2008년부터 개발되어온 open source project다.<sup>1</sup> Wayland display server는 객체지향 비동기 protocol을 통해 client와 통신하는 client-server 모델로 동작한다.<sup>2</sup> Server는 client로부터 전달된 이벤트를 처리하며 동시에 window system 자원 관리, window compositing, window 관리 정책을 수행한다. Figure 2는 Wayland display server와 Wayland client 간의 S/W 계층 구조를 보여준다.

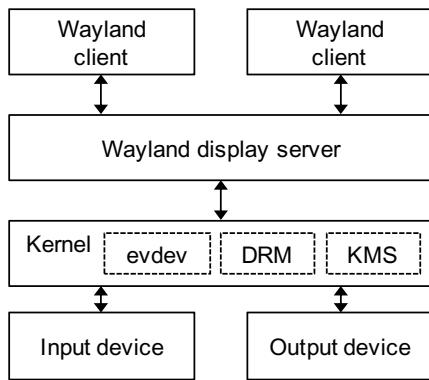


Figure 2. Wayland window system 구조

Wayland display server 는 window 관리 정책이 통합되어 있어 window system 전체 IPC 비용이 X window system 에 비해 낮으며 server 와 window manager 사이의 이벤트 race condition 문제가 발생 되지 않는 장점을 갖고 있다. Wayland client 는 그래픽스 하드웨어를 이용해 직접 rendering 을 수행하고 rendering 완료된 window 버퍼를 server 로 전달해 compositing 시키는 모델을 갖고 있다. 이러한 모델은 display server 의 rendering 기능을 server side compositing 에 집중시킬 수 있도록 해준다. 그 밖의 X window system 과 Wayland window system 의 기능별 주요 차이점은 Table 1과 같다.

주요 기능	X	Wayland
Window 관리	Server 로 부터 분리	Server 통합
Composition	선택적 (확장 지원)	필수
Window 버퍼 관리	Server	Client
Protocol 정의	C 코드 구현	XML parsing
Rendering protocol 지원	Legacy API 지원	미 지원
원		
Client 간 IPC 지원	지원 (ICCCM)	미 지원
Network-transparent protocol 지원	지원	미 지원

Table 1. X vs. Wayland 기능 비교

### 3. Tizen window system

Tizen 은 다양한 스마트 디바이스 개발을 목표로 하는 open source operating system 으로 Mobile, TV, Wearable, IVI 프로파일을 지원한다.<sup>3</sup> Tizen application 은 rendering 기능과 window system 추상 layer 를 제공하는 open source framework 인 EFL (Enlightenment Foundation Libraries)을 이용한다. EFL 내부 모듈인 Ecore 는 상위 layer 에 event handling 기능과 window system 추상화를 제공 한다.<sup>4</sup>

Tizen 3.0 은 Wayland window system 으로 동작한다. Tizen 2.x 까지 사용 되었던 X server 와 X protocol 이

각각 Wayland display server 와 Wayland protocol 로 교체 되었다. X window manager 인 Enlightenment 가 Wayland display server 로 동작 할 수 있도록 기능이 확장 되었으며, Wayland window system 추상화를 수행하는 Ecore\_Wayland 가 EFL 내부 모듈로 추가 되었다.

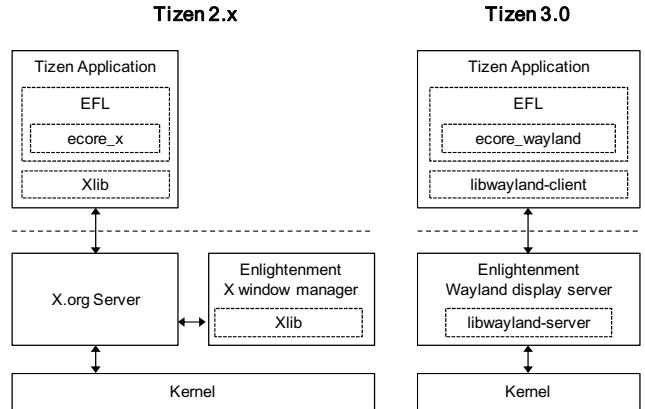


Figure 3. Version 별 Tizen window system 구조

Ecore 가 제공하는 window system 추상화로 Tizen 2.x 기반 application 의 Wayland window system migration 작업은 application 의 큰 수정 없이 완료 되었다. Figure 3 의 좌측이 Tizen 2.x 에서 동작하는 X window system 이며, 우측이 Wayland 로 변경된 3.0 시스템 이다.

### 4. X window vs Wayland on Tizen

Tizen 에 적용된 X window system 과 Wayland 간의 성능 차이를 비교하기 위해 application launching time, rendering 성능, CPU 사용률, window resize 성능에 대한 항목을 설정하였다. Tizen TV profile 의 X11 과 wayland 플랫폼 바이너리를 GPU driver 와 함께 ARM Cortex-A9 기반의 Odroid U3 디바이스에 설치하여 두 window system 의 성능을 비교 측정하였다.<sup>5,6,7,8</sup>

#### 4.1 Application Launching Time 비교

Application launching time 은 사용자의 입력 시점부터 화면에 해당 application 의 첫 화면이 출력될 때까지의 경과 시간을 의미한다. Launching time 측정에 사용된 application 은 EFL 로 작성되었으며, 측정 구간은 실험의 오차를 줄이기 위해 process fork 시점부터 첫 화면의 rendering 완료 시점으로 단순화하였다.

5 번 반복 측정한 launching time 결과는 Figure 4에 도식화 하였다. 각 측정 데이터의 유의미한 편차는 측정되지 않았으며, X window system 하 app 의 평균 launching time 은 411.8ms 그리고 Wayland 하 app 의 평균 launching time 은 302ms 으로 Wayland 기반 시스템에서 약 36%의 성능 향상을 보였다. 이 성능 차이는 두 window system 의 구조와 단순화된 protocol 로 인한 IPC 의 횟수 차이로 판단되며, 이 가설을 입증하기 위한 IPC 횟수 측정 실험을 수행하여 그 결과를 Figure 5에 도식화 하였다.

IPC 의 측정 횟수는 application launching 과정에서 발생한 IPC 의 총 수를 count 한 것이며, IPC 발생의

판단 기준은 server 측 socket에서 발생한 sendmsg (Wayland의 경우), writev (X server의 경우), recvmsg 호출 횟수이다.

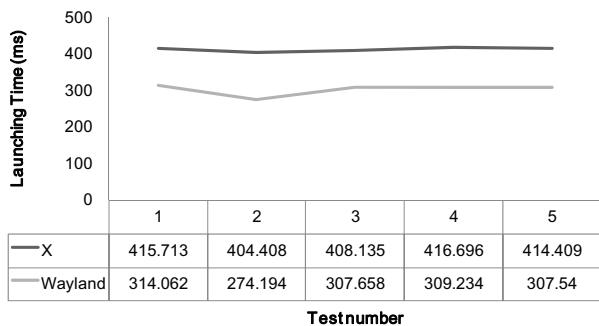


Figure 4. Application launching time 비교

X server는 application launching 시점에 window manager 및 application과 평균 945회의 이벤트를 주고 받았으나, Wayland display server는 application과 평균 55회의 이벤트를 주고 받았다. Window manager 기능이 Wayland display server로 통합된 구조적 이점과 단순하게 설계된 Wayland protocol이 IPC overhead를 감소 시켜 X server 대비 높은 application launching time 성능을 보여준다.

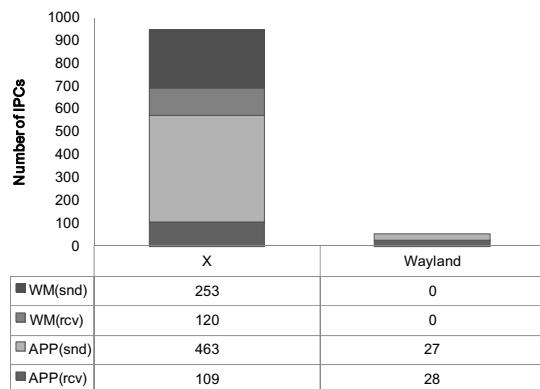


Figure 5. Application launching 시 IPC 수 비교

#### 4.2 Rendering 성능 비교

Rendering 성능 측정은 EFL의 benchmark suite인 expedite를 사용하였으며, 총 91개의 test case 중 rendering 측정에 유의미한 65개 case를 대상으로 측정 하였다. 또한, rendering 성능의 최대 값을 정확히 측정하기 위해 LCD VSYNC 값인 60HZ 제한을 제거하고 X-GL과 Wayland-GL 엔진에 대하여 5회씩 실험을 수행한 후 그 평균값을 기준으로 두 window system의 성능을 비교하였다.

expedite의 최종 score는 X server의 경우 49.6 fps (frame per second), 그리고 Wayland의 경우 65.9 fps가 측정되었다.

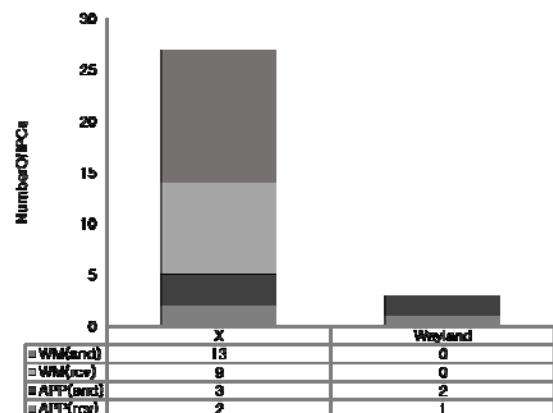


Figure 6. Rendering 시 IPC 수 비교

또한, IPC 수가 rendering 성능에 얼마나 영향을 주는지 상세 분석을 위해 평균 score와 가장 가까운 값을 기록한 test case인 Image Blended Smooth Scaled test의 drawing 명령을 flush하는 시간을 측정/비교하였다. X-GL 엔진의 경우 1.388ms, Wayland-GL 엔진은 0.243ms가 측정되었는데 실제 GPU가 drawing 자체를 수행하는 시간은 window system과 상관없이 동일하기 때문에 이 결과의 차이는 drawing 시간과 실제 drawing 된 결과가 window system에 반영되는 과정에서 발생하는 window system과 application 간의 동기화 시간이라고 해석할 수 있다.

Figure 6에서 확인할 수 있듯이 동일한 test case에서 application이 한 번 rendering을 하는 동안 발생한 IPC의 횟수는 window system에 따라 큰 차이를 보인다. X server는 window manager 및 application과 평균 23회의 이벤트를 주고 받았고, Wayland display server는 application과 평균 3회의 이벤트를 주고 받았다. 결과적으로 Wayland display server는 rendering 수행 시 X server에 비하여 1/7의 IPC밖에 발생하지 않았다. 독립된 compositor 대신 직접 composition을 수행하는 Wayland display server의 구조적 특징이 IPC의 수를 눈에 띄게 감소시켰고 그 결과 application의 rendering 성능이 20% 이상 향상 되었다고 분석된다.

#### 4.3 CPU 사용률 비교

CPU 사용률 비교는 window system과 관련된 프로세스들의 CPU 사용률을 EFL 기반 application의 실행부터 종료까지 측정한 테스트다.

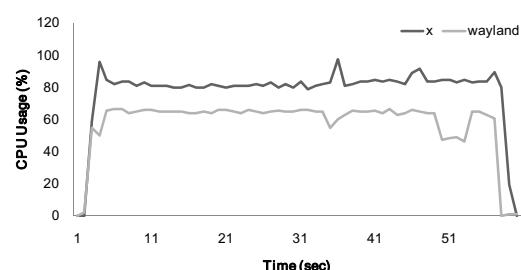


Figure 7. CPU 점유율 비교

X server, X window manager, X client 가 실행 되는 X window system 의 최대 CPU 점유율은 97.6% 였고, Wayland display server, Wayland client 가 실행 되는 Wayland window system 의 경우는 최대 점유율이 66.7%로 기록되었다. 두 window system 을 비교 측정 해본 결과 독립된 compositor 가 필요하지 않고 단순화된 프로토콜로 적은 양의 IPC 횟수를 요구하는 Wayland 는 X 대비 적은 수의 display event 가 발생하며 그 결과 computation 양이 줄어드는 장점이 있다는 것을 확인할 수 있다. 동일 operation 에 대해 적은 computation power 를 요구한다는 것은 저전력 특성이 중요한 배터리 기반 디바이스에는 무시할 수 없는 이점이 될 수 있다.

#### 4.4 Window Resize 성능 비교

Window resize 성능 비교는 application 이 display server 에 resize 를 요청한 시점부터 window resize 가 완료된 시점까지 소요된 시간을 비교 측정하였다. 각 테스트에서 수행한 실험을 위한 window resize 크기 구성은 Table 2에 명시되어 있다.

실험 번호	변경하는 크기
실험 1	가로 100px 세로 100px
실험 2	가로 1000px 세로 240px
실험 3	가로 850px 세로 900px
실험 4	가로 400px 세로 1000px
실험 5	가로 1800px 세로 500px
실험 6	가로 999px 세로 999px
실험 7	가로 333px 세로 999px
실험 8	가로 1920px 세로 1080px

Table 2. Window resize configuration

Table 2에 기술되어 있는 각 항목은 실험 1부터 실험 8 까지 순서대로 총 10 회 수행되었고, 이에 대한 산술 평균값을 통해 X server 와 Wayland 에서 한 window 를 정해진 크기로 resize 하는데 걸리는 시간을 측정 하였다. 그 결과는 Figure 8에서 확인할 수 있듯이 모든 실험에서 Wayland 의 Window Resize 소요시간이 X server 에서의 소요시간 보다 적게 측정되었다.

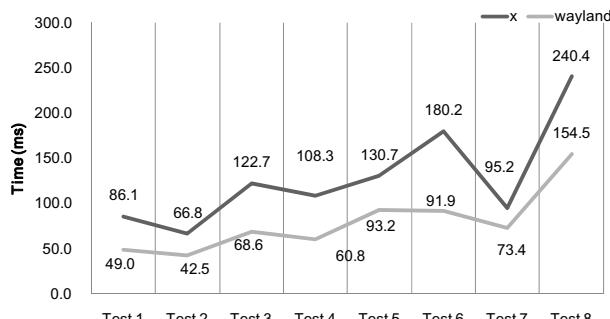


Figure 8. Window resize 시간 비교

Figure 9은 application 이 window resize 되는 동안 주고받는 IPC 횟수를 기록한 그래프이다. X serve 의 경우 server 는 application 과 window manager 와 평균 약 87 회의 IPC 가 발생한 반면 Wayland 의 경우 server 와 application 사이에 약 8 회의 IPC 가 발생하였다. Wayland 는 resize 시 X server 에 비해 1/10 에 해당하는 양의 IPC 만으로 resize 가 가능하기 때문에 전체 resize 에 소요되는 시간이 모든 실험의 경우에서 20% 이상 적게 측정된 것으로 판단된다.

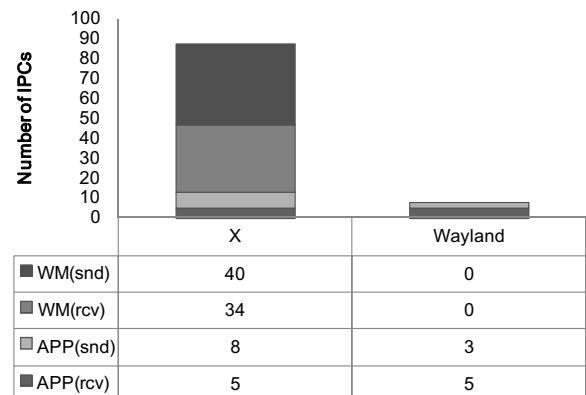


Figure 9. Window resize 과정에서 발생한 IPC 수 비교

## 5. 결론

본 논문에서는 Wayland window system 이 갖는 구조적, 성능적 장점을 X window system 과 비교하여 설명하였다. Wayland 는 단순한 protocol 구조를 갖고 있으며, Wayland display server 에 window 관리 정책이 통합되어 X server 대비 동일 기능 제공에 필요한 IPC 오버헤드가 적다. Tizen 3.0에 적용된 Wayland 와 X server 의 launching time, rendering 성능, CPU 사용량, window resize 성능 등의 성능 비교 실험을 통해 IPC 최적화의 장점을 확인할 수 있었으며, 공통적인 성능 향상에는 IPC 오버헤드 감소가 그 중심에 있음도 더불어 확인할 수 있었다.

## 참고문헌

- [1] Wikipedia, Wayland (display server protocol) [Internet], [https://en.wikipedia.org/wiki/Wayland\\_\(display\\_server\\_protocol\)](https://en.wikipedia.org/wiki/Wayland_(display_server_protocol)).
- [2] K. Högsberg, Wayland Protocol and Model of Operation [Internet], <http://wayland.freedesktop.org/docs/html/ch04.html>.
- [3] Tizen Project, About [Internet], <https://www.tizen.org/about>.
- [4] raster, About EFL [Internet], <https://www.enlightenment.org/about-efl>.
- [5] Hardkernel, Odroid U3 Features [Internet], [http://www.hardkernel.com/main/products/prdt\\_info.php?g\\_code=g138745696275](http://www.hardkernel.com/main/products/prdt_info.php?g_code=g138745696275).
- [6] Tizen Project, Tizen 3.0 TV profile binary [Internet], <https://download.tizen.org/snapshots/tizen/tv/>.
- [7] M. Lee, Mali DDK for Tizen 3.0 (X11 enabled) [Internet], <https://source.tizen.org/mali-ddk-tizen-3.0-x11-enabled>.
- [8] Sanjeev BA, Mali DDK for Tizen 3.0 (Wayland enabled) [Internet], <https://source.tizen.org/mali-ddk-tizen-3.0-wayland-enabled>.