Tizen IVI OS의 빠른 부팅을 위한 systemd와 wayland사용

양태희, 조금산, 추현승
성균관대학교 정보통신대학
e-mail:{gumy88, josan123}@skku.edu, choo@ece.skku.edu

Tizen IVI OS's fast booting by using systemd and wayland

Taehea Yang, Geumsan Jo, Hyunseung Choo
College of information and Communication Engineering
Sungkyunkwan University

요 약

인텔과 삼성이 손을 잡고 개발한 Tizen IVI(In-vehicle Infotainment) OS는 임베디드 기기의 환경에 맞추어 사용자에게 최대한 빠른 시간 내에 서비스를 제공하기 위하여 여러 가지 방법을 사용한다. 특히 Automobile 산업을 겨냥하여 개발한 Tizen IVI OS는 운전자에게 빠른 서비스를 제공하기 위하여 systemd와 wayland를 활용하여 빠른 운영체제 부팅을 제공하고 있다. 최대 7초 이내의 부팅 속도를 제공하기 위하여 기존의 init process를 대체하는 systemd를 사용한다. 또한 기존의 x-window를 대체하는 wayland를 사용하여 부팅과정의 오버헤드를 줄이려 노력하고 있다. 본 논문에서는 최근 스마트폰의 보편화와 함께 임베디드 기기 상에서 더욱 필요성이 증대되고 있는 운영체제의 빠른 부팅에 대한연구를 소개한다. 특히 Tizen IVI OS에서 빠른 부팅을 위해 사용하는 두 가지 방식에 대해 연구하고, 기존 방식들과의 차이점을 분석한다.

1. 서론

최근 스마트폰, 자동차산업에서 안드로이드, iOS, Tizen IVI등 여러 가지 모바일 플랫폼들이 경쟁을 벌이면서 양질의 서비스와 더불어, 사용자에게 임베디드 기기 시동과 동시에 서비스를 제공하기 위한 여러 가지 방법들이 논의되고 있다. 특히 자동차산업에서는 차체 내 통합 스마트 환경을 위한 IVI 제품들이 지속적으로 개발되고 있다. 이 IVI 산업에 적용되는 플랫폼은 운전자가 탑승 후 시동을 거는 짧은 시간에 부팅을 마치고 서비스를 제공해줘야 한다는 요구사항이 필수이다. 대표적인 IVI 플랫폼은 본 논문에서 논의할 Tizen IVI OS가 있으며 이 운영체제 역시 IVI 제품의 이슈인 부팅속도에 대하여 여러 가지 방안들이 논의된다[1].

그중 한 가지는 대부분의 모바일 플랫폼이 기반으로 삼고 있는 리눅스의 초기 프로세스 로드의 오버헤드를 줄이기 위한 방안이다. 종료 시점의 메모리 및 시스템 리소스를 비휘발성 메모리에 저장해두고, 부팅 시 이를 참조하는 snapshot boot 방식이다[2]. 이 방식은 Tizen IVI OS 에서는 사용되지 않지만 최근 FA-LINUX사의 제로부트 등 여러 가지 부트 로더에서 제안하고 있는 방식이다. 또한 snapshot boot를 사용하지 않고 기존의 리눅스 부팅방식처럼 /etc/inittab 과 /etc/rc.d/* 을 참조하여 초기 프로세스들을 로드하되 기존에 순차적으로 처리하던 이 로드 과정을 socket and D-Bus activation을 사용하여 병렬적으로 바꾸어 부팅하고, cgroup(control group)을 내부적으로 사용하여 로드되는 프로세스들을 tracing 하여 결과적으로 1번 프로세스인 init-process를 대체하는 systemd 사용 방식이 있다.

이 외에도 부팅 후 최대한 빠른 시간에 사용자에게 서비스를 제공할 수 있는 환경구축과 관련된 연구도 활발하다. 그 중 현재 배포되고 있는 모든 리눅스에서 사용하던 x-window[3] 의 문제점을 해결하는 것도 한 가지 해결책으로 대두되고 있다. x-window는 누구나 수정 가능한 리눅스의 특징을 살려 x-server 와 x-client를 두어 서버와 클라이언트 간 약속한 프로토콜만 만족시키면 어떤 어플리케이션이든지 리눅스에서 제공하는 GUI 환경을 사용하게 해주는 네트워크 지향적인 방법이다. 하지만 x-window는 성능상의 문제점이 있다. 가장 핵심적인 문제는 호환성을 위해 두었던 x-server가 렌더링 결과를 커널로 보내어 실제로 그림을 그리는 과정을 막고 있어 오버헤드가 생기게 된다. 이 같은 문제점 때문에 wayland 에서는 렌더링을 담당하는 compositor가 직접 커널과 통신하게 하는 방식으로 GUI의 오버헤드를 줄이려 노력하고 있다.

본 논문 2장에서는 Tizen IVI OS의 성능향상에 사용하는 systemd와 이것을 구성하는 cgroup에 대하여 알아본다. 또한 이를 통해 systemd가 대체하는 init-process에 대하여 알아보고, 기존의 방식과 systemd방식의 차이점을 알아본다. 또한 기존 x-window의 내용과 문제점을 파악한

후 wayland와 비교하여 필요성을 도출한다. 3장에서는 Tizen IVI OS가 아직 소스코드를 공개하지 않은 이유로 systemd를 사용하는 페도라16과 리눅스 최초로 wayland를 사용하겠다고 발표한 우분투12.10을 분석하여 어떻게 사용하고 있는지 자세히 살펴보고 최종적으로 Tizen IVI OS로의 적용에 대해서 연구한다. 4장에서는 결론적으로 리눅스에서 systemd와 wayland의 필요성을 설명한다.

2. 관련 연구

Tizen IVI OS 에서는 부팅 속도 향상을 위하여 크게 세 가지 방법을 제시한다. 첫째는 init-porcess의 순차적인 초기화를 병렬적으로 바꾸는 systemd 사용방법이다. 두 번째는 x-window의 태생적인 유저-커널영역간의 오버헤드를 줄이기 위한 wayland 사용이다. 세 번째는 root file system 을 최적화하여 초기화간 사용되는 모듈 로딩들을 최소화하는 방식이다. 하지만 본 논문에서는 다른 대부분의 리눅스에서 사용되는 root file system 최적화는 다루지 않고, systemd와 wayland만 연구한다.

2.1. Linux booting과 init process

리눅스 기반 운영체제에서는 하드웨어, 커널, 램 디스크 초기화 이후 0번 프로세스인 초기화 프로세스가 idle 상태가 된다. rest_init를 통해 생성된 1번 프로세스, 즉 init-process를 생성한다. 이 프로세스는 유저의 컴퓨팅 자원의 원활한 접근을 위한 초기화를 주목적으로 하여 쉘, 쉘변수, 환경변수, fsck, 네트워크, 루트 파일시스템, 모듈 로딩 등을 수행하고 pid 가 1보다 큰 모든 프로세스들의 부모 프로세스가 된다.

<표 1> 일반적인 리눅스 부팅 과정

일반적인 리눅스 부팅 순서 수행 과정	
1	시스템 결함 점검
2	커널 이미지, 램디스크 램으로 복사
3	커널 이미지 압축 해제 후 재배치
4	하드웨어 스캔
5	파일시스템 확인 후 마운트
6	/etc/inittab을 참조하여 init-process 설정
7	init-process(/sbin/init) 실행
8	/sbin/rc.d* 의 프로세스들을 통해 초기화 진행
9	쉘 로그인

또한 실행 시 /etc/inittab 스크립트를 참조하여 init process 실행을 위한 설정을 로드하고 이후 /etc/rc.d의 스크립트들에 작성된 작업을 수행하게 된다[4]. 이때 초기화 작업을 수행하는 과정에서 기존의 리눅스는 스크립트들에 적힌 작업들을 순차적으로 처리하게 되어 부팅 속도 저하에 영향을 미치게 된다.

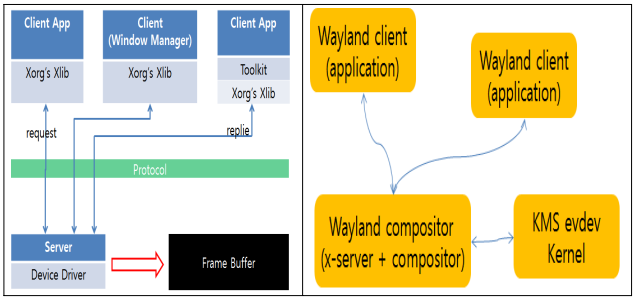
2.2. cgroup and systemd

cgroup 은 사용자가 CPU 시간, 시스템 메모리, 네트워크 대역폭과 같은 자원이나 조합을 시스템에서 실행 중인 사용자 정의 프로세스 간에 할당할 수 있게 해주는 기능이다. 또한 설정한 그룹을 모니터링하거나 특정 자원으로의 cgroup의 액세스를 거부하는 것 이외에 실행 중인 시스템에서 cgroup을 동적으로 다시 구성할 수 있게 해준다[5].

systemd 에서는 socket and D-Bus activation을 사용해 의존성 있는 start-up 서비스들을 병렬적으로 수행하며 better-framework를 제공한다. 또한 위와 같은 cgroup을 사용하여 프로세스들을 트래킹 할 수 있다. systemd는 위 두 가지 기능을 통하여 기존의 linux-booting에서 사용되는 init-process-daemon을 대체한다.

2.3. x-window and wayland

x-window는 MIT 대학을 중심으로 1984년에 개발된 이후로 오늘날까지 대부분의 리눅스에서 사용하고 있는 윈도우 시스템이다. 이 시스템의 가장 큰 동기는 어떤 컴퓨터에서 운용되는 어플리케이션이 다른 컴퓨터의 디스플레이 장치를 이용할 수 있으며, 이때 이 두 개의 하드웨어는 동일한 구조, 운영체제를 가질 필요가 없다는 것이다. 즉 어플리케이션 프로그램은 장치로 부터 독립적이다.



(그림 1) x-window 와 wayland 비교

하지만 과거에 설계된 x-window는 오늘날에 들어 성능상의 이슈에 봉착했다. 가장 큰 이유는 그래픽 관련 작업을 수행하는 과정에서 어플리케이션과 디바이스 간에 x-server라는 추가된 레이어가 있기 때문이다. 클라이언트에서 요청된 GUI 처리 루틴이 서버를 통해 compositor에서 렌더링 되고, 이 결과가 디바이스에 써지는 과정에서 상당한 오버헤드가 존재한다. 이러한 문제점에서 착안하여 Wayland의 새로운 GUI 메커니즘은 x-server와 렌더링을 담당하는 compositor를 하나로 통합하여 GUI 처리루틴의 한 과정을 생략하고, 조금 더 커널에 밀접한 구조를 통하여 시스템 콜로 인한 컨텍스트 스위칭의 오버헤드를 최대한 줄여 GUI 성능상의 이점이 있다[6].

3. 사용 분석

Tizen IVI OS는 현재 커널 소스가 제공되지 않는다. 때문에 3.1절에서는 최초로 systemd를 부팅과정에 사용한 페도라16로 분석한다. 또한 wayland도 현재 사용하는 배포판이 없고 출시될 우분투12.10에서 사용될 예정이기 때문에 3.2절에서 우분투12.10로 분석한다.

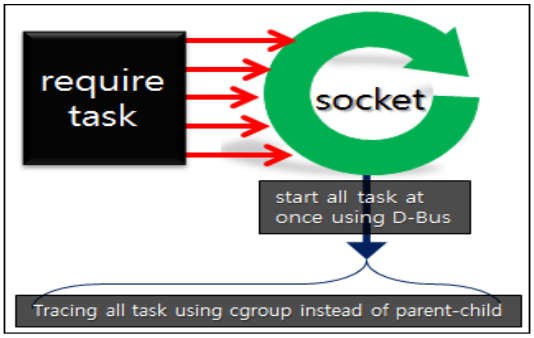
3.1 fedora16 에서의 systemd 분석

systemd는 maturity 가 2년여 밖에 안 되는 신생 프로젝트이다. 하지만 기존에 부팅과정에서 사용하던 sysvinit 이나 upstart에 비하여 지원하는 인터페이스와 기능들이 현격히 뛰어나다. 그중 가장 핵심적인 차이점은 socket and D-Bus를 사용하고 내부적으로 cgroup를 사용한다는 것이다. 이 외에도 마운트와 스왑, 암호화된 하드디스크를 핸들링 하는 등 부팅 속도를 향상시킬 수 있는 다양한 장점들을 갖추고 있다.

<표 2> sysvinit, upstart, systemd 비교

핵심특징 비교(1.sysvinit 2.upstart 3.systemd)			
	1	2	3
Interfacing via D-Bus	no	yes	yes
Socket-based Activation	no	no	yes
Interactive boot-up	no	no	yes
Control Group Control(cgroup)	no	no	yes
Mount handling	no	no	yes
fsck handling	no	no	yes
Early boot /dev/log logging	no	no	yes
Swap handling	no	no	yes

systemd가 부팅 과정을 살펴보면, 첫째로 systemd는 초기화 데몬이 실행되기 전 미리 listening socket을 생성하여 부팅 과정 간 로드될 프로세스들을 accept 시켜둔다. 모든 프로세스 가 accept되면 소켓에 쌓인 task들을 D-Bus를 사용하여 동시에 실행시킨다. 이 과정 중 task간 의존성이 생기면 해당 task를 블록하고 다른 task의 응답을 기다리게 된다. 하지만 이 과정에서도 이 두 task 이외 나머지 프로세스들은 병렬적으로 진행되고 있기 때문에 결과적으로 순차적인 기존의 초기화 과정보다 빠른 booting 속도를 확보할 수 있다. 또한 기존 순차적 초기화 과정에서는 부모 자식 관계로 task들을 관리하던 것을 각각의 task에 libcgroups 라이브러리를 이용하여 cgroup을 붙여서 초기화중 task들을 tracing 할 수 있게 한다.



(그림 2) systemd flow

3.2 우분투12.10 에서의 wayland 분석

관련 연구에서 알아봤듯이 오늘날의 x-window는 client, helper libraries, 또는 host operating system kernel 등을 모두 통합하기 위하여 x-stack 이 과도하게 복잡해졌다. 또한 x-server와 compositor의 분리로 인한 오버헤드 때문에 성능 저하가 발생한다. 이러한 문제점 때문에 cairo와 같은 2D 라이브러리들은 x-window와 독립적으로 렌더링 하기 위하여 Qt와 GTK+같은 독립적인 compositor를 사용하게 됐다. 또한 같은 맥락으로 x-window로 부터 겪는 성능저하를 막기 위해서 최근 Gui library 들은 메모리 관리나 display 관리를 커널에 의존하고 있다.

wayland의 설계는 위 문제점에서 착안됐다. 과도하게 많은 기능들을 포괄하고 있어 성능상의 저하를 유발하는 x-server를 삭제하는 대신 x-client 즉 각각의 어플리케이션들에게 기존에는 x-server를 통해 수행하였던 윈도우 관리, 화면 렌더링을 담당하게 변경한다. 그리고 wayland compositor 에서는 위 과정에서 사용될 버퍼 입/출력을 시스템 전반적인 메모리 관리 인터페이스를 통해 커널과 통신할 수 있게 한다.

4. 결론

fedora16과 우분투12.10에서의 systemd, wayland 사용을 살펴본 결과 기존 시스템의 문제점에서 도출된 문제들을 해결하기 위한 방안으로 나온 기술들이기 때문에 과도기가 지나고 안정성이 확보되면 다른 대부분의 리눅스들에서도 차용할 기술들이다. 특히 운전자에게 빠른 부팅을 필요로 하는 IVI 플랫폼에서의 systemd와 wayland는 필수적인 기술이다.

Acknowledgement

본 연구는 지식경제부(정보통신산업진흥원) 대학ITRC, 교육과학기술부(한국연구재단)의 차세대정보컴퓨팅기술개발사업 및 중점연구소지원사업의 일부지원으로 수행되었음(NIPA-2012-(H0301-12-3001), 2012-0006420, 2012-0005861).

참고문헌

[1] 한국자동차공학회 편집부 (편집자) “IVI 단말기의 SW 기술과 표준화 동향” 한국자동차공학회 Workshop 2011.

[2] 박세진, 송재환, 박찬익 “개선된 스냅샷 부트를 이용한 임베디드 리눅스의 빠른 부팅 기법” 정보과학회논문지:컴퓨팅의 실제 및 레터 제14권 제6호, 2008.8.

[3] 이경호, 한순흥, 이동곤, 이규열 “객체지향 개념과 가시화 기법에 의한 선박 개념설계용 그래픽 사용자 인터페이스 모델” 대한조선학회논문집 제29권 제4호, 1992.11.

[4] 신광무, 박성호, 정기동 “임베디드 시스템에서 리눅스의 빠른 부팅” 한국정보과학회 2005 가을 학술발표 문집(Ⅰ)제32 제2호, 2005.11.

[5] P. Menage, linux kernel document [Internet] www.kernel.org/doc/Documentation/cgroups/

[6] Wayland team, Wayland Architecture,[Internet] waw.wayland.freedesktop.org/architecture/