KOCAELİ ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

Linux Ağ Yönetimi

8. Hafta – Dahili Giriş/Çıkış

Yrd. Doç. Dr. A. Burak İNNER

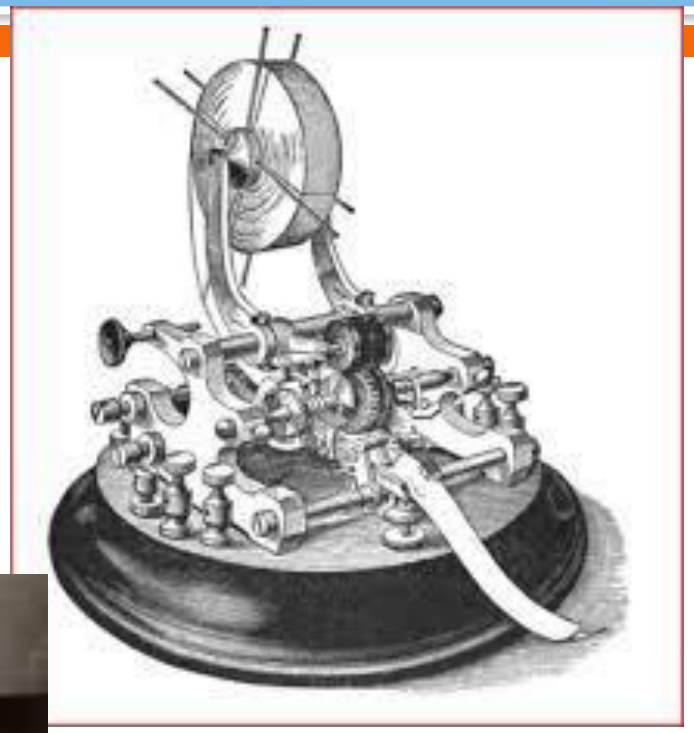
Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr

Dahili Giriş/Çıkış

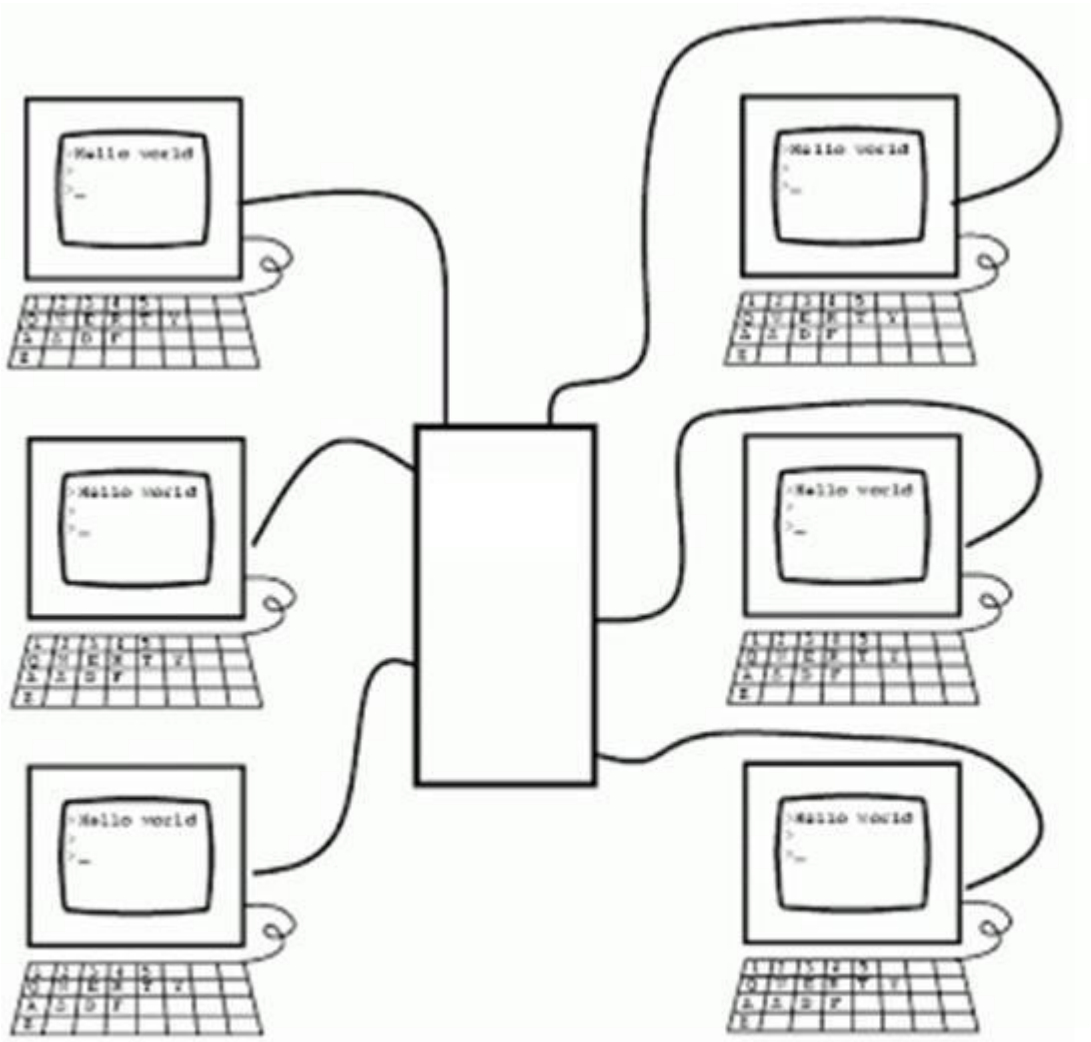
- Sistemdeki her işlem aşağıdakilere sahiptir
  - Çıkış cihazı: stdout tarafından tanımlanıdr
  - Giriş cihazı: stdin tarafından tanımlanır
  - Hata cihazı: stderr tarafından tanımlanır
- Burada mesajların nasıl ekrana yansıtıldığını veya klavyeden nasıl giriş okunduğunu anlamamız gerekiyor.
- Ayrıca I/O nun nasıl yönlendirildiği ve pipe'ların nasıl çalıştığını anlamamız gerekiyor.
- Bu bölüm tüm bu konuları içermektedir.
- Bunların tamamını anlayabilmemiz için öncelikle Linux Kernelinin içinde bulunan TTY altsistem konseptini ve bunun nasıl operasyonları nasıl yönettiğini anlamamız gerekiyor.

TARİHİ ARKAPLAN

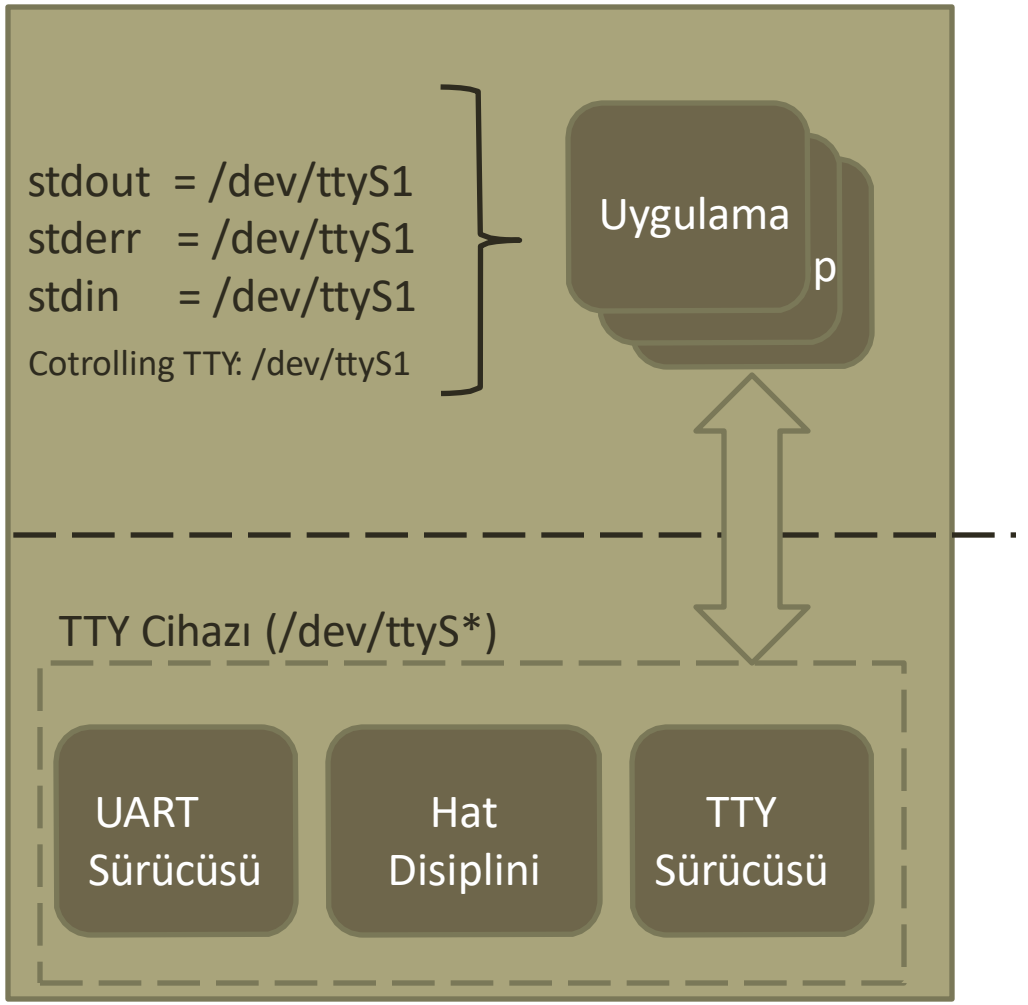
1869
Delikli Kart



BİLGİSAYARIN DOĞUŞU

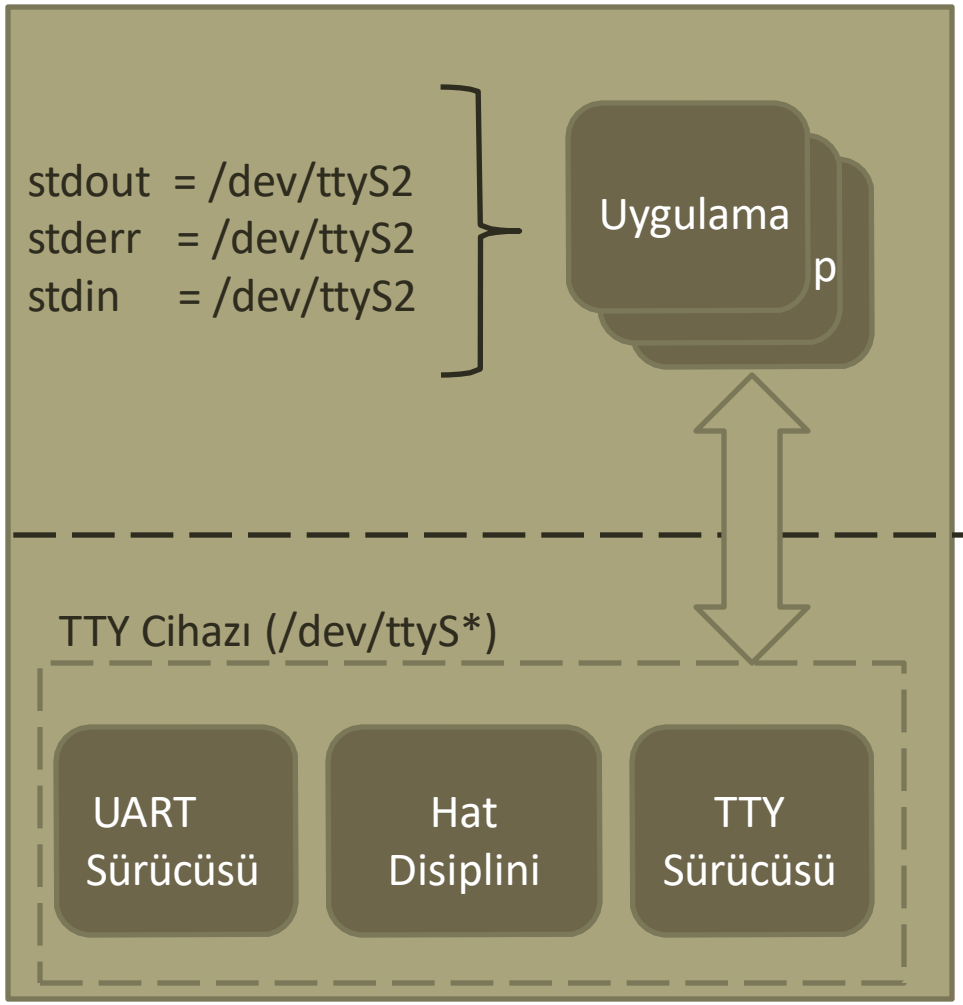

Hello world
Hello world
Hello world
Hello world
Hello world
Hello world

TTY Cihazı


stdout = /dev/ttyS1
stderr = /dev/ttyS1
stdin = /dev/ttyS1
Cotrolling TTY: /dev/ttyS1
Uygulama
p
TTY Cihazı (/dev/ttyS*)
UART Sürücüsü
Hat Disiplini
TTY Sürücüsü

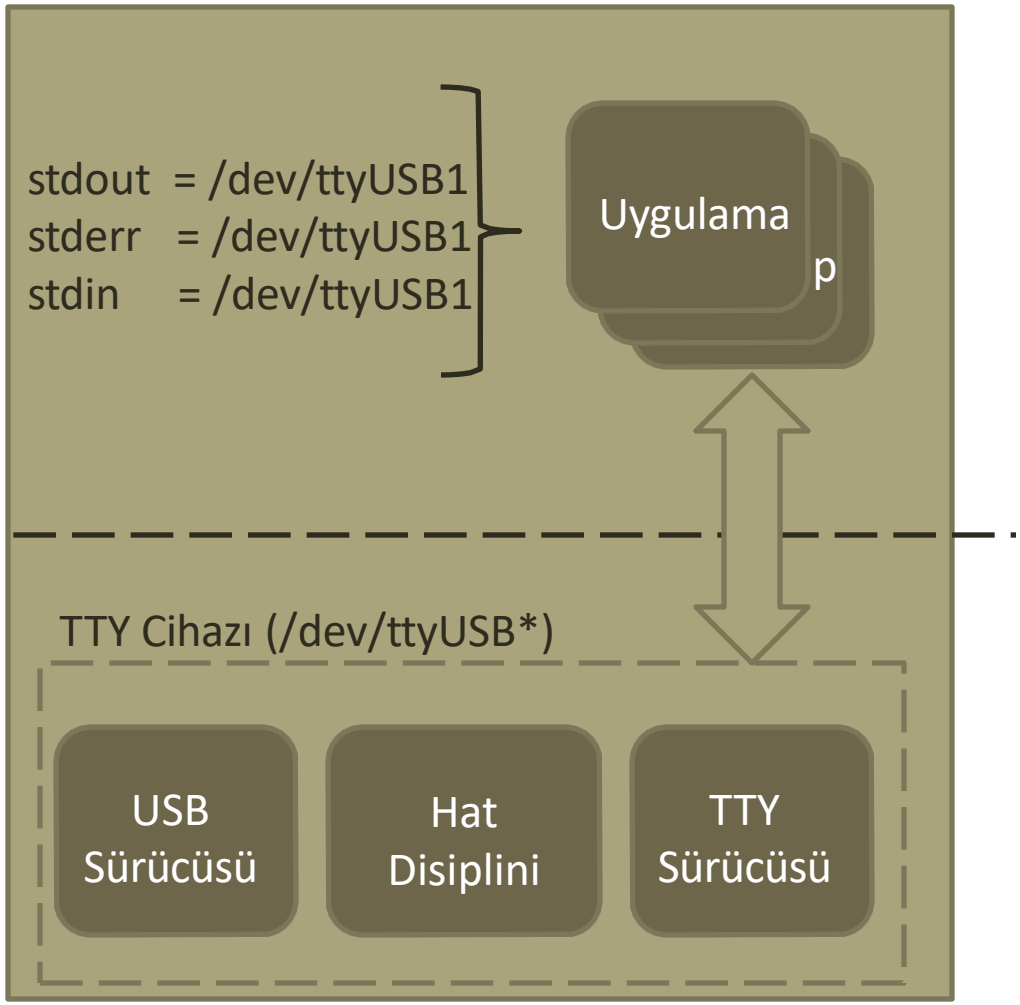
TTY Cihazı


PSTN

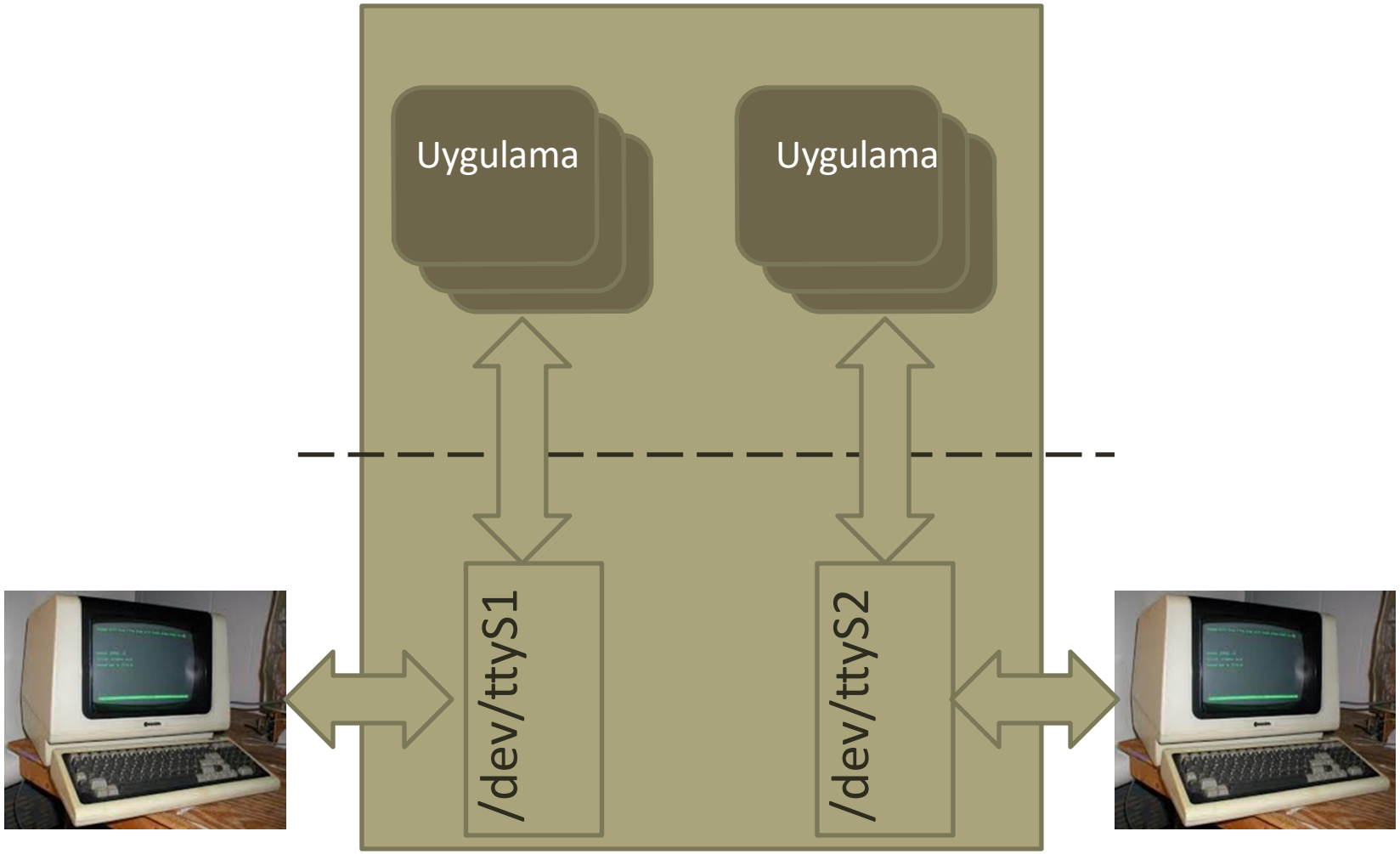

stdout = /dev/ttyS2
stderr = /dev/ttyS2
stdin = /dev/ttyS2
Uygulama
p
TTY Cihazı (/dev/ttyS*)
UART Sürücüsü
Hat Disiplini
TTY Sürücüsü

TTY Cihazı

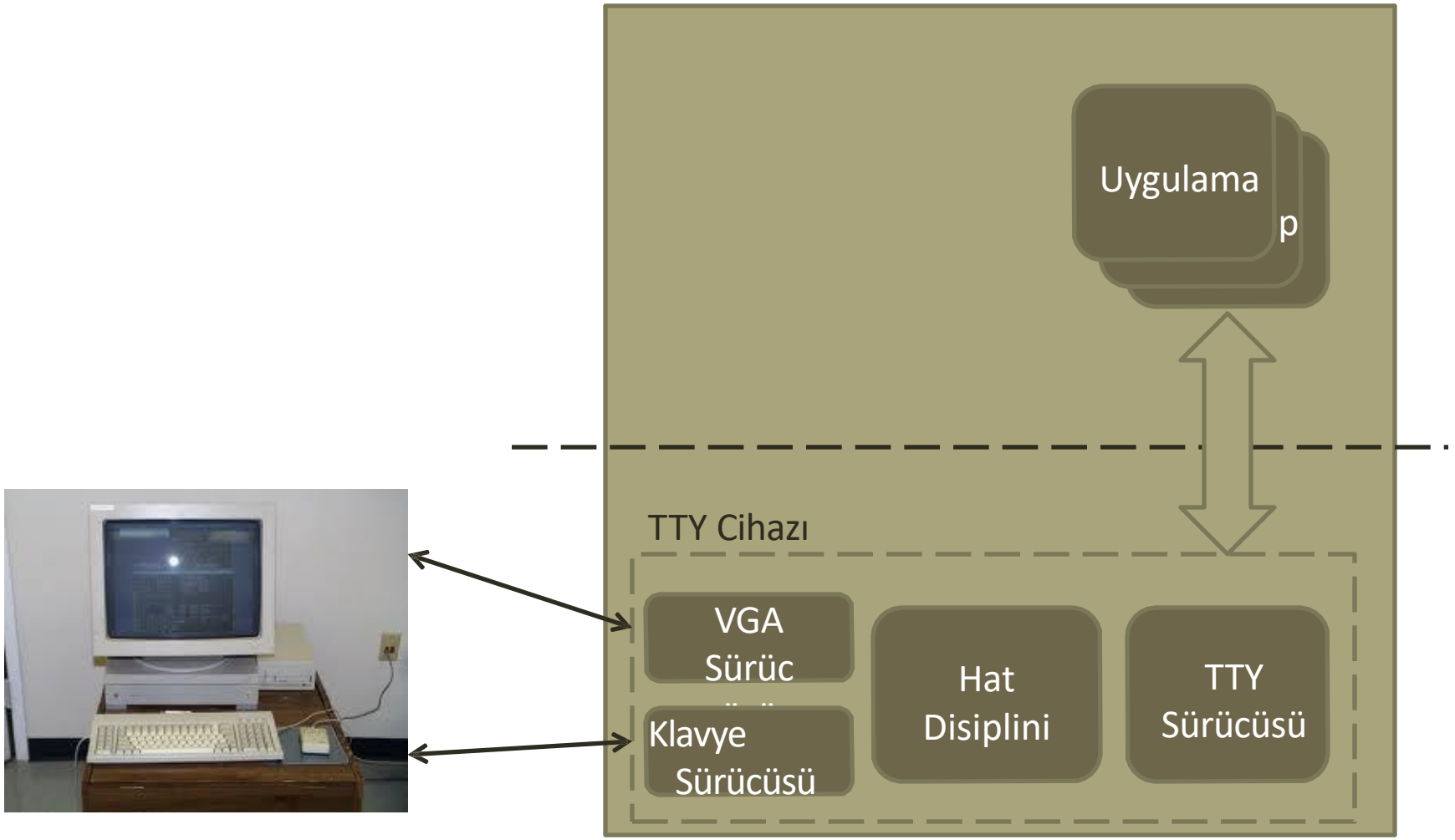

PSTN


stdout = /dev/ttyUSB1
stderr = /dev/ttyUSB1
stdin = /dev/ttyUSB1
Uygulama
p
TTY Cihazı (/dev/ttyUSB*)
USB Sürücüsü
Hat Disiplini
TTY Sürücüsü

Çoklu Terminal



Ve Kişisel Bilgisayarlar Gelir



TTY Devices

- TTY, TeleTYpewriter kısaltmasıdır.
- TTY makineleri ilk olarak delikli kartlar ve Telex makinelerinde kullanıldı.
- Bilgisayarlar tanıtıldığında, Bir bilgisayar büyük bir üniteydi, TTY makinesi I/O terminali olarak kullanılıyordu ve bilgisayara UART aracılığıyla bağlanıyordu.
- Terminal kontrolü bir modem aracılığı ile gerçekleşiyordu.

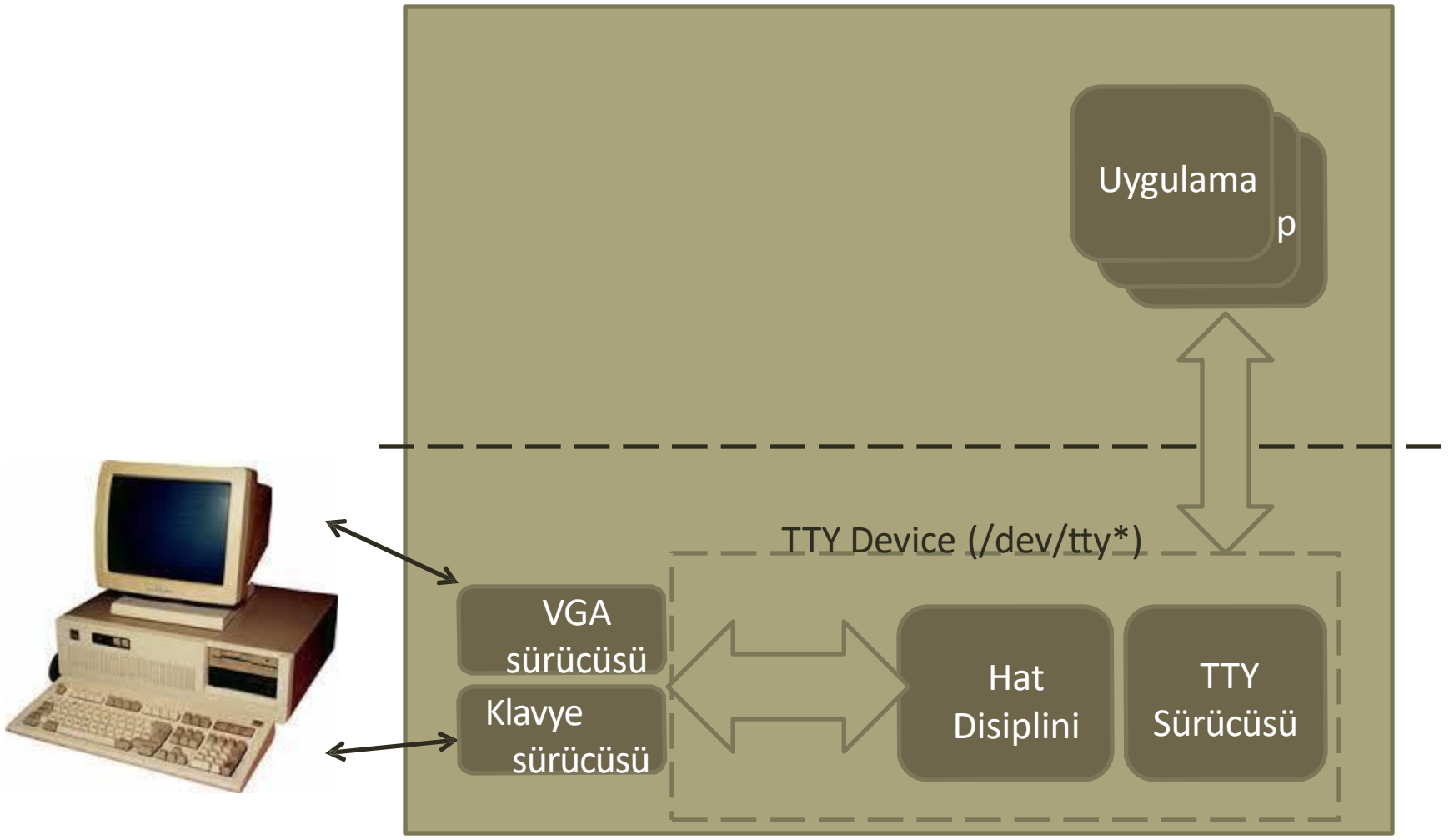
- TTY cihazı 3 katmandan oluşur.
  - Donanım arayüzü için bir sürücü(UART driver, USB driver, VGA/KB drivers, ..)
  - Alt serviye düzenlemeler için bir iletişim hattı (backspace, erase word, clear line, ….)
  - TTY'nin kullanıcı uygulamalarıyla etkileşimi için driver

- Kullanıcı uygulamaları TTY ile sinyal ve çağrılarla ile haberleşir
- Kullanıcı uygulamasının varsayılan I/O cihazı TTY'dir (/dev/tty*)
- Bu TTY terminali ayrıca kullanıcı uygulamalarını da yönetir.

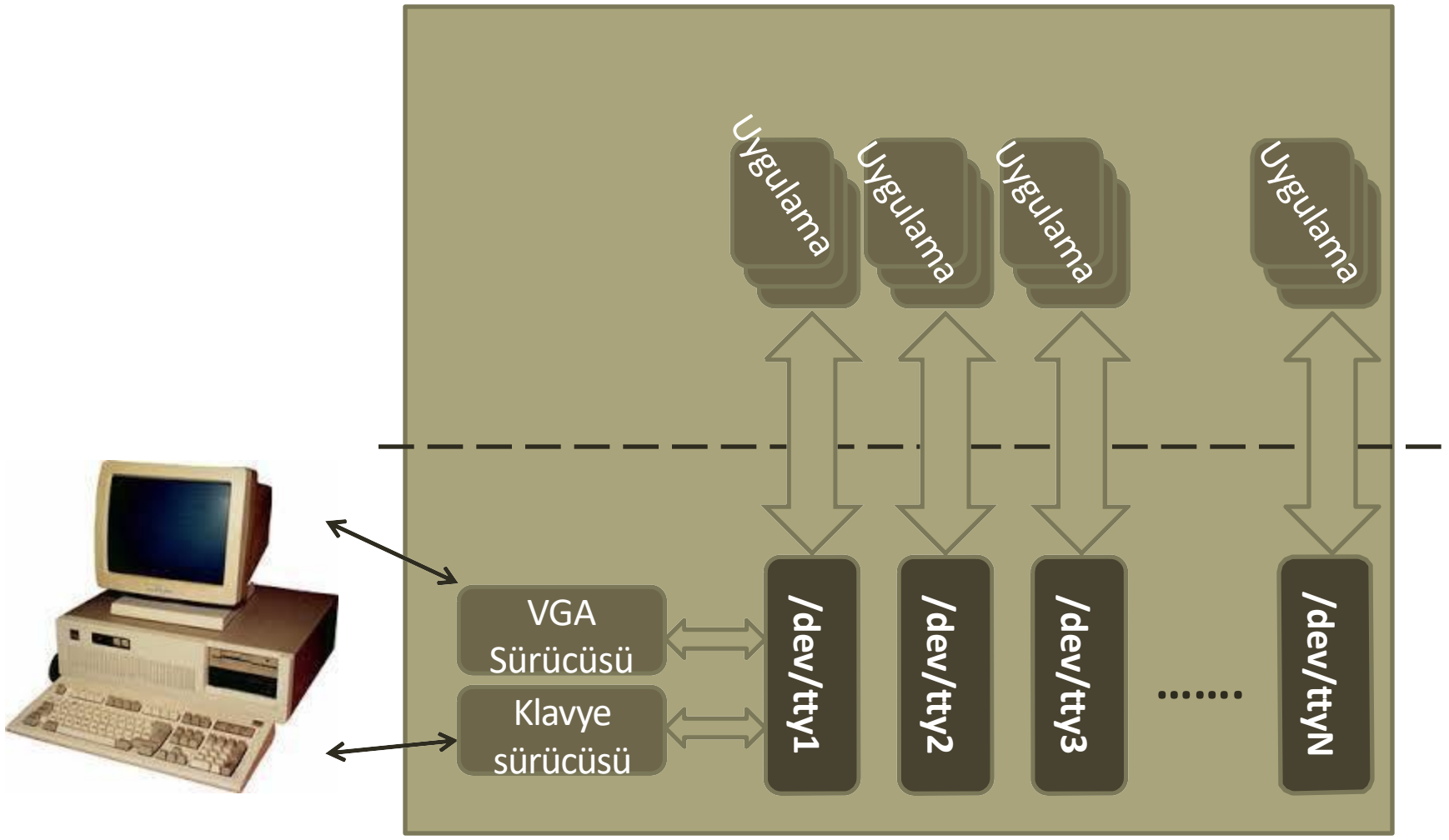
Sanal Terminaller

- Terminale serial interface ile bağlandığımız zaman bilgisayarda birden fazla terminale sahip olabilir.
- Fakat bilgisayara monitör ve klavyeyi doğrudan bağladığımız ve aynı yapıyı kullanıdımız zaman birden fazla terminale sahip olamayız.
- Birden fazla terminali tek bir fiziksel terminal içinde bulundurabilemiz için yeni bir mimariye ihtiyacımız vardır.
- Bu sayede Linux'te sanal terminal konsepti oluştu.

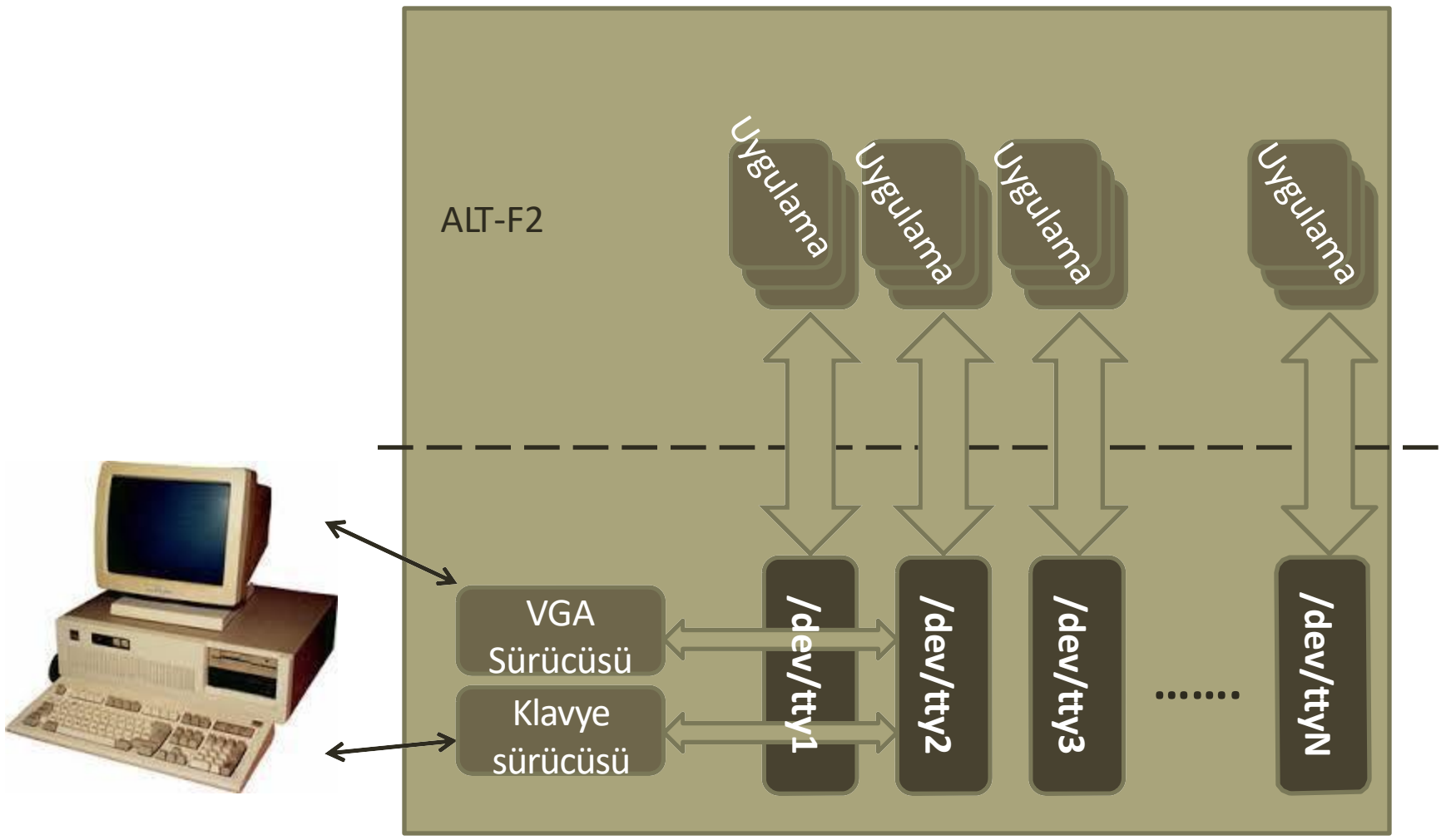
Sanal Terminal



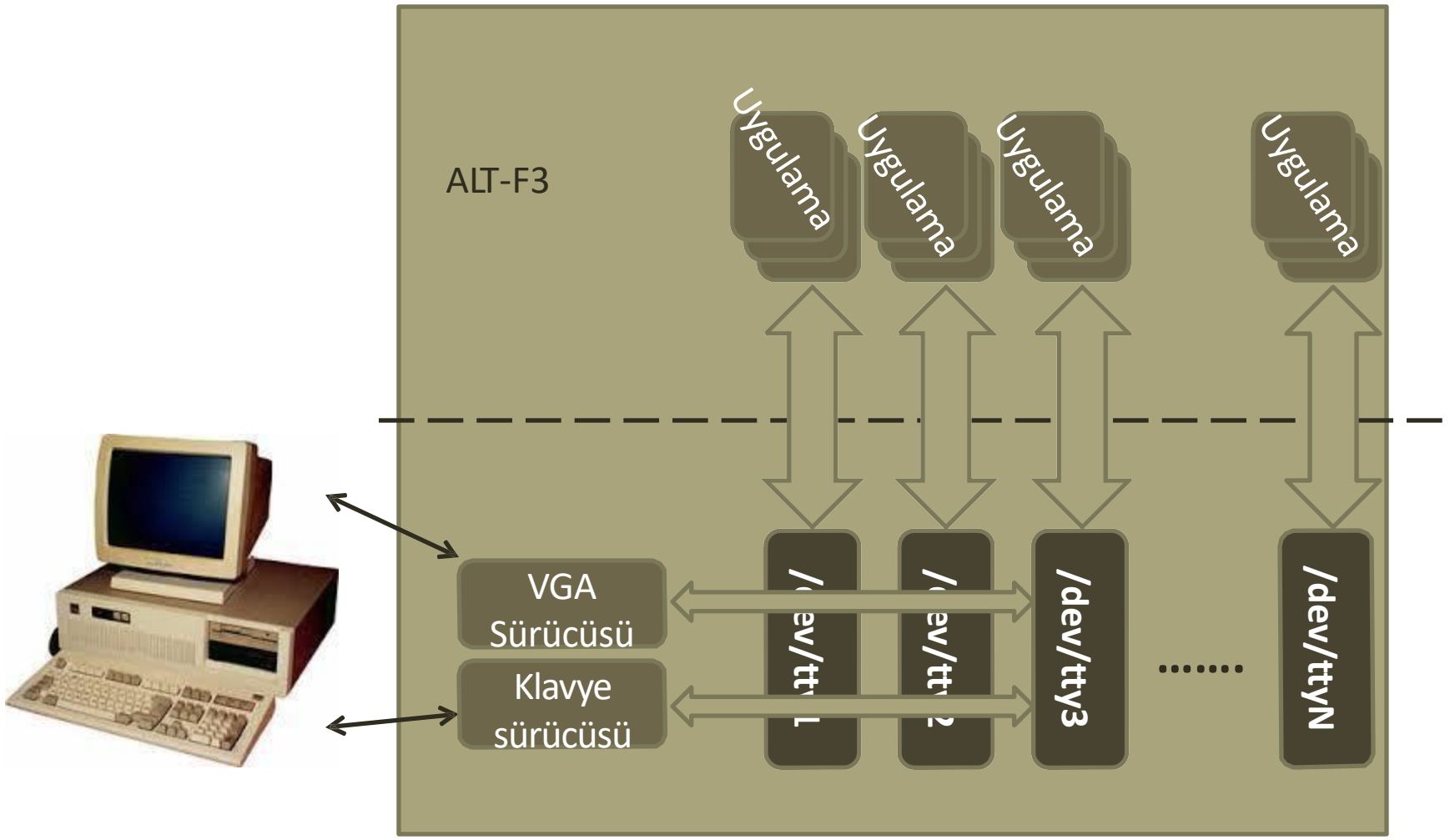
Çoklu Sanal Terminal



Çoklu Sanal Terminal

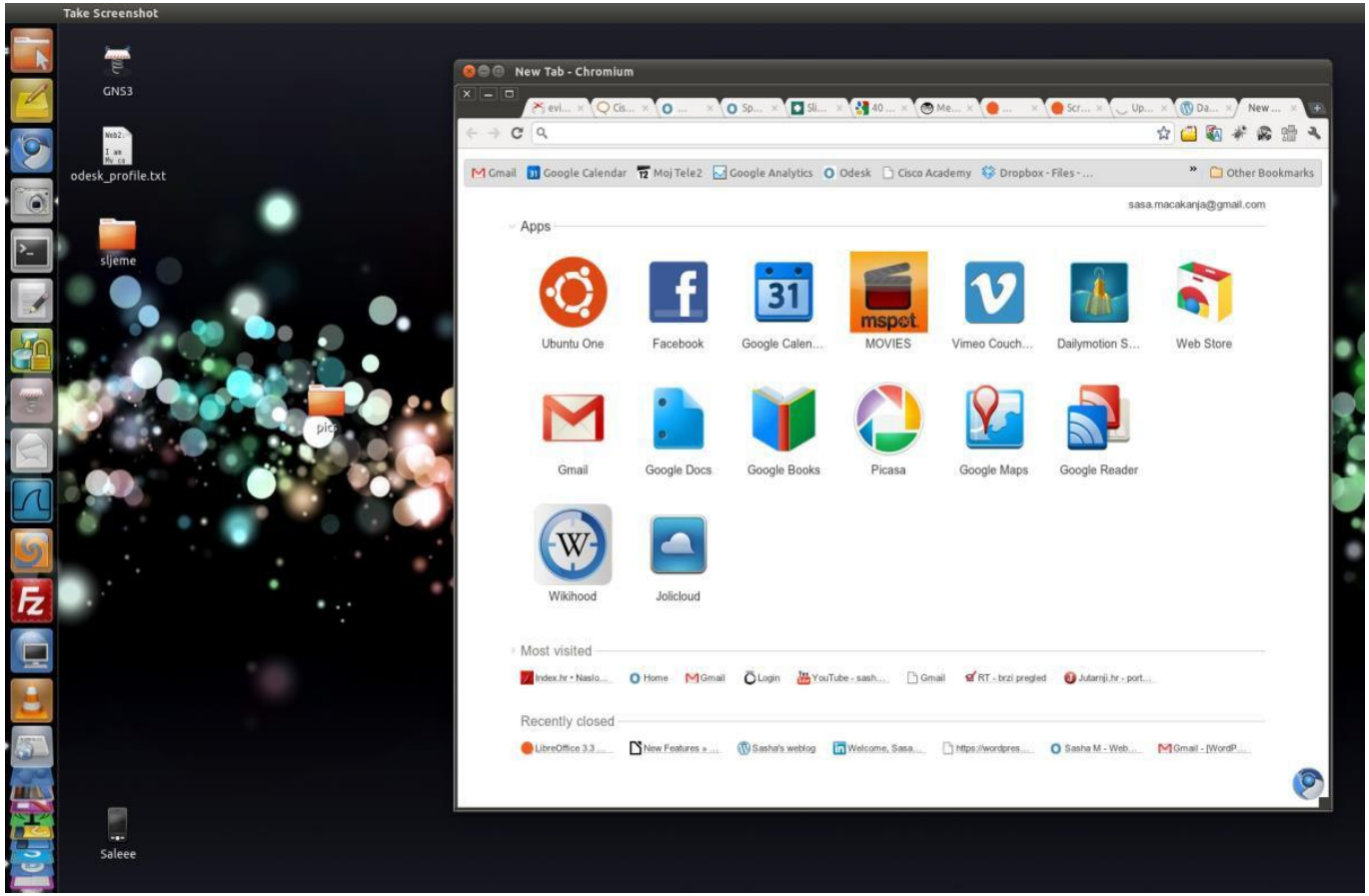


Çoklu Sanal Terminal



Sanal Terminaller

- Yeni mimari birden fazla terminali üzerinde barındırma yeteneğine sahipti.(Aynı fiziksel terminal üzerinde)
- Monitör ve klavye artık TTY cihazının birer parçası değildir. Her biri ayrı birer cihaz artık.
- Linux kerneli 63 sanal terminale kadar destekler. (/dev/tty1 up to /dev/tty63)
- Çoğu dağıtım ise yalnızca 7 terminale kadar destekliyor. (/dev/tty1 - /dev/tty7)
- Her sanal terminalde kullanıcı uygulaması getty başlar.
- (getty=get tty, Terminale yeni bir bağlantı fark ettiğinde kullanıcı adı ve şifre istemek için 'oturum açma' programını çalıştırır.)
- Bütün sanal terminaller aktiftir fakat yalnızca biri fiziksel terminale erişebilir.
- Sanal terminaller arasında geçiş yapmak için ALT-Fn kullanılır.



GRAFİK ARAYÜZÜ

Linux GUI

- Az önce Text I/O hakkında konuşmuştuk.

  İhtiyacımız olanlar
  - Windowing (çoklu pencere, maximize, minimize, ...)
  - Fontlar, renkler, ikonlar
  - Fare hareketlerini algılama
  - Grafik gösterme
- Linux X-Windows sistemi sayesinde tüm bunları gerçekleşetirebilir.
- X-Window Sistemi bitmap ekranlar için bir pencereleme sistemidir.

Aşağıdakilerden oluşur:

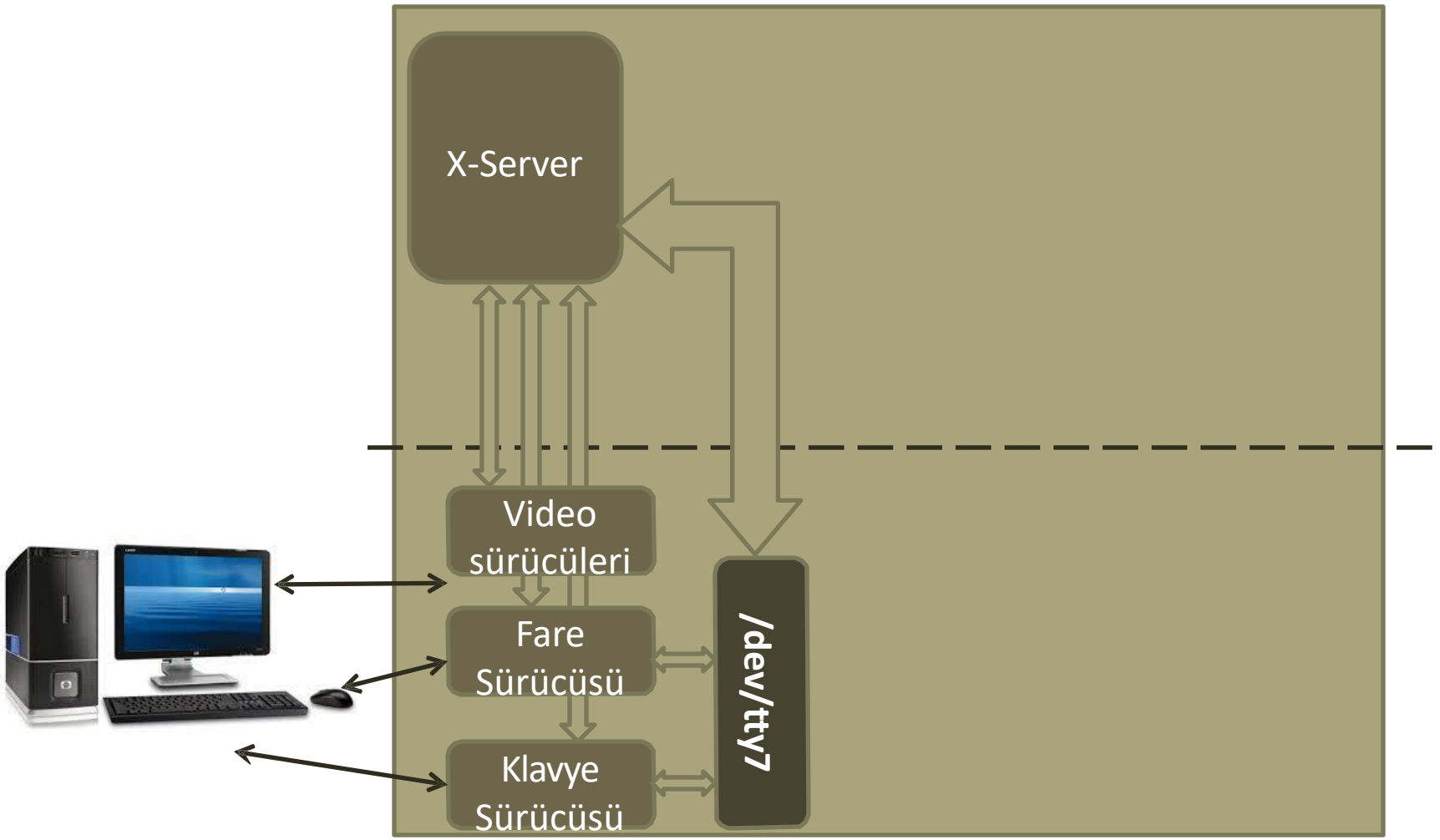
- Sunucu(X-Server):
  - Grafik arayüzü algılayıp sürücü ile bağlayacak bir program.
  - Çoğu Linux dağıtımı Xorg bunun için kullanır.
- Kullanıcı(X-Client)
  - Kullanıcı kısmında tüm uygulumaların GUI'ye ihtiyacı vardır.
  - Uygulamalar X-Server'a ***X Protocol*** ile bağlanır.

X-Server ve Sanal Terminal

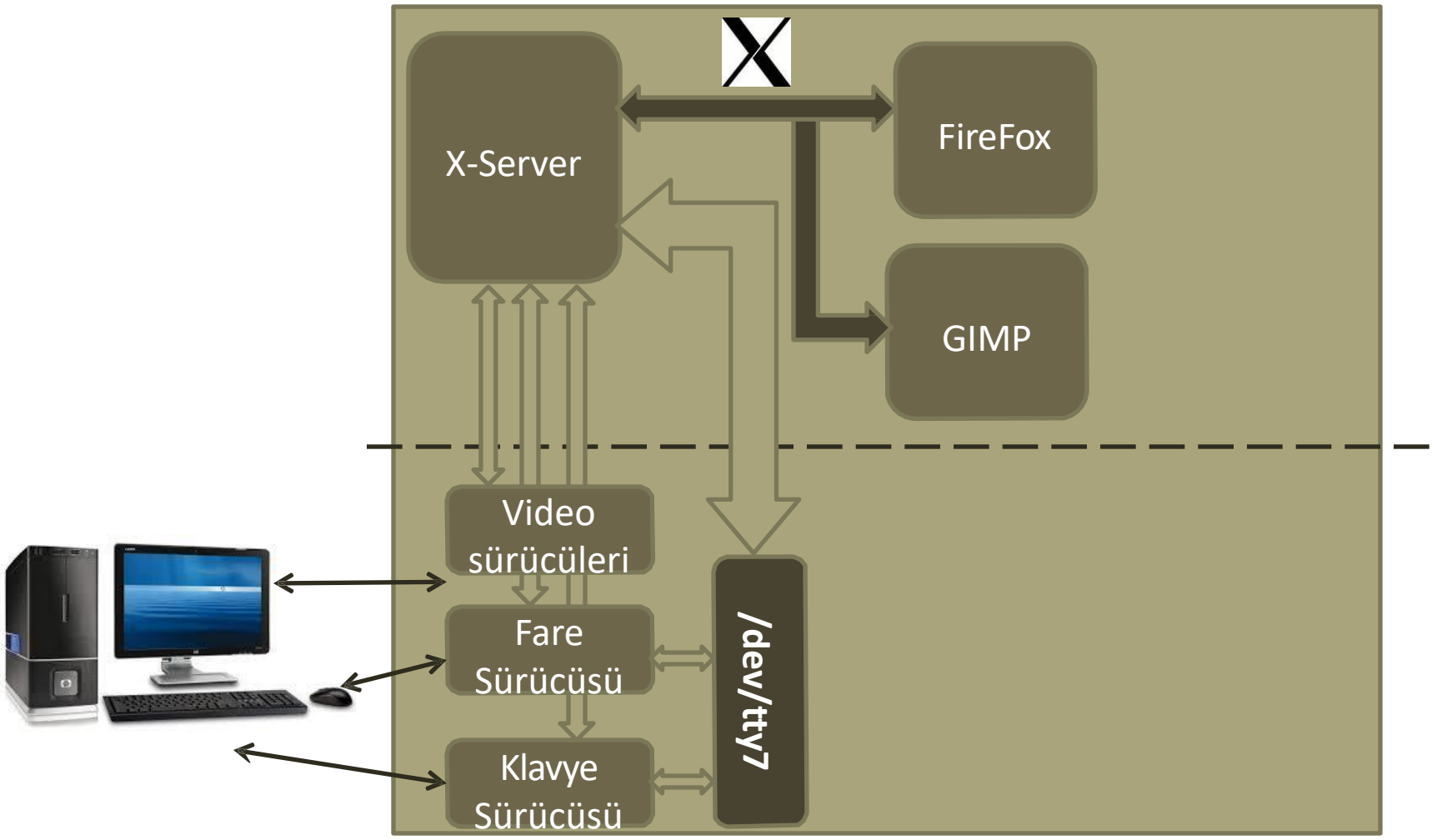
- Daha önce de bahsedildiği gibi, Linux dağıtımları sanal terminal'leri yüklerken getty'yi çalıştırır.
- X-server en az bir sanal terminal üzerinde başlar. Bu X-Session #0 dır.

- Kullanıcı ihtiyacı olurda X-session sayısını arttırıp azaltabilir.
- Alt-Fn'in GUI'de başka bir anlamı olduğu için onun yerine sanal terminal arası geçiş yapmak için CTRL+ALT+Fn kullanılır.
- X-Server'ın çalıştığını varsayarasak (/dev/tty7)
  - SF#1 için ➔ Ctrl-Alt-F1
  - GUI'e dönmek için➔ Alt-F7

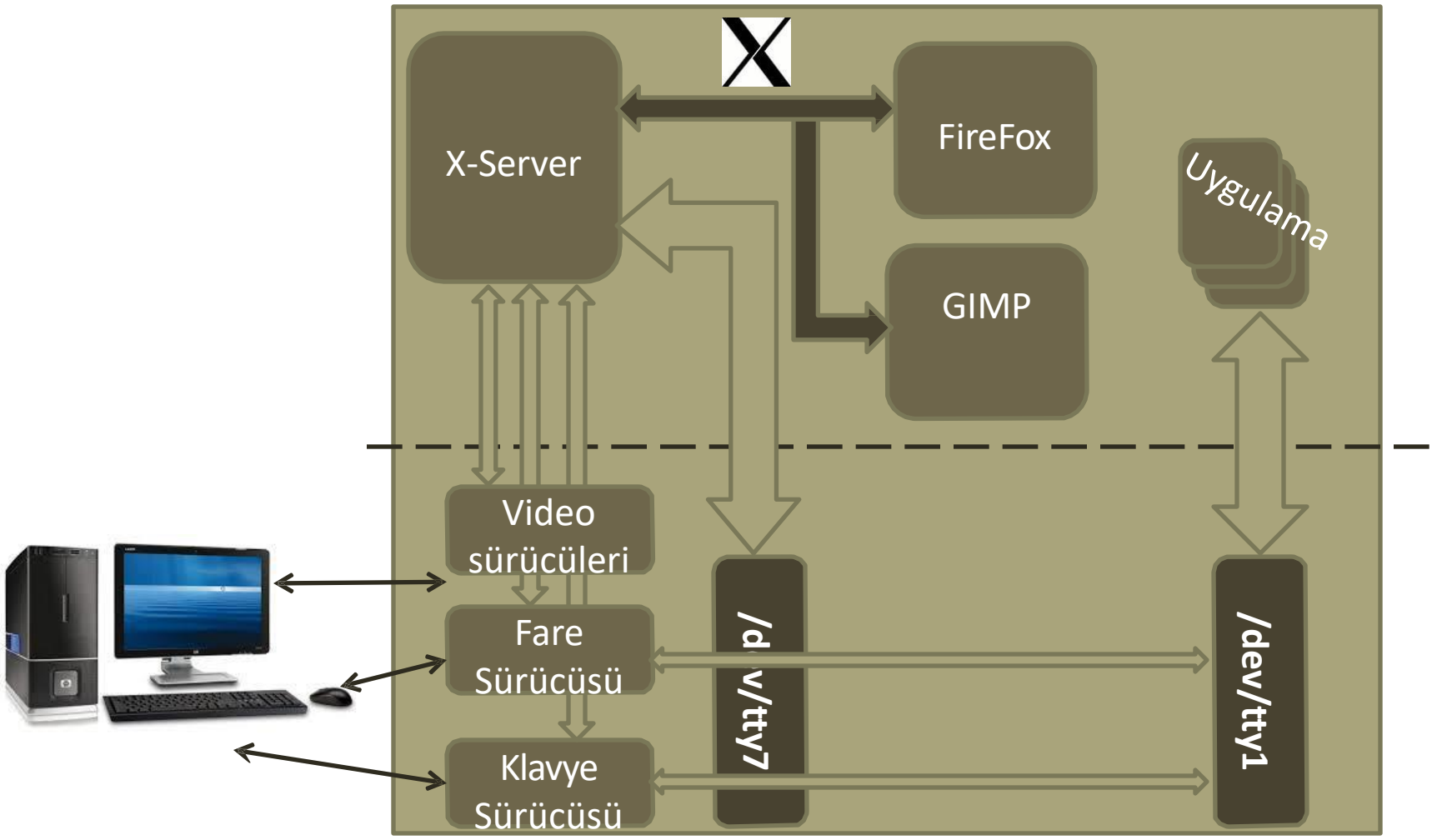
X-Server Kullanımı



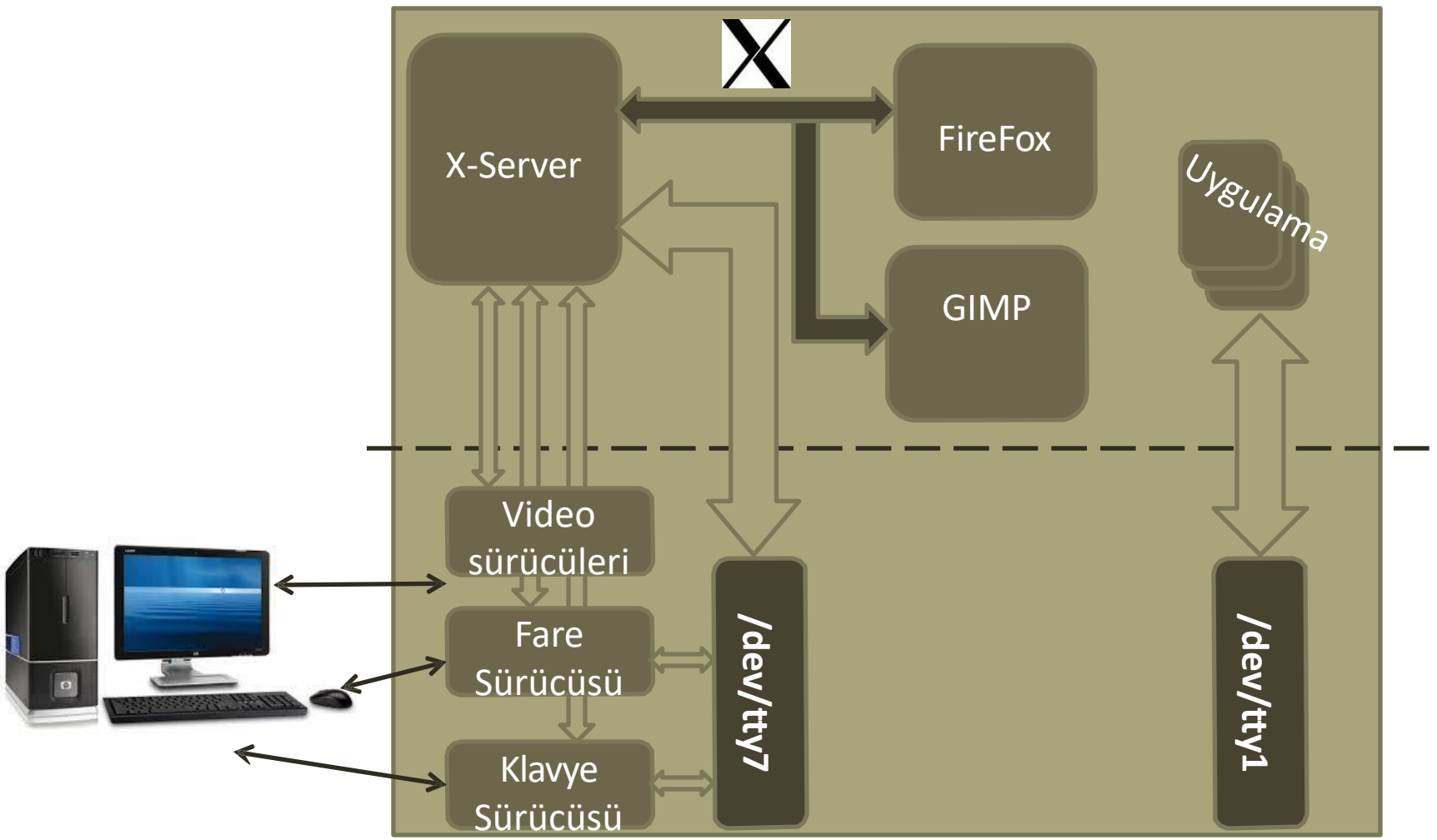
X-Server Kullanımı



X-Server Kullanımı



X-Server Kullanımı



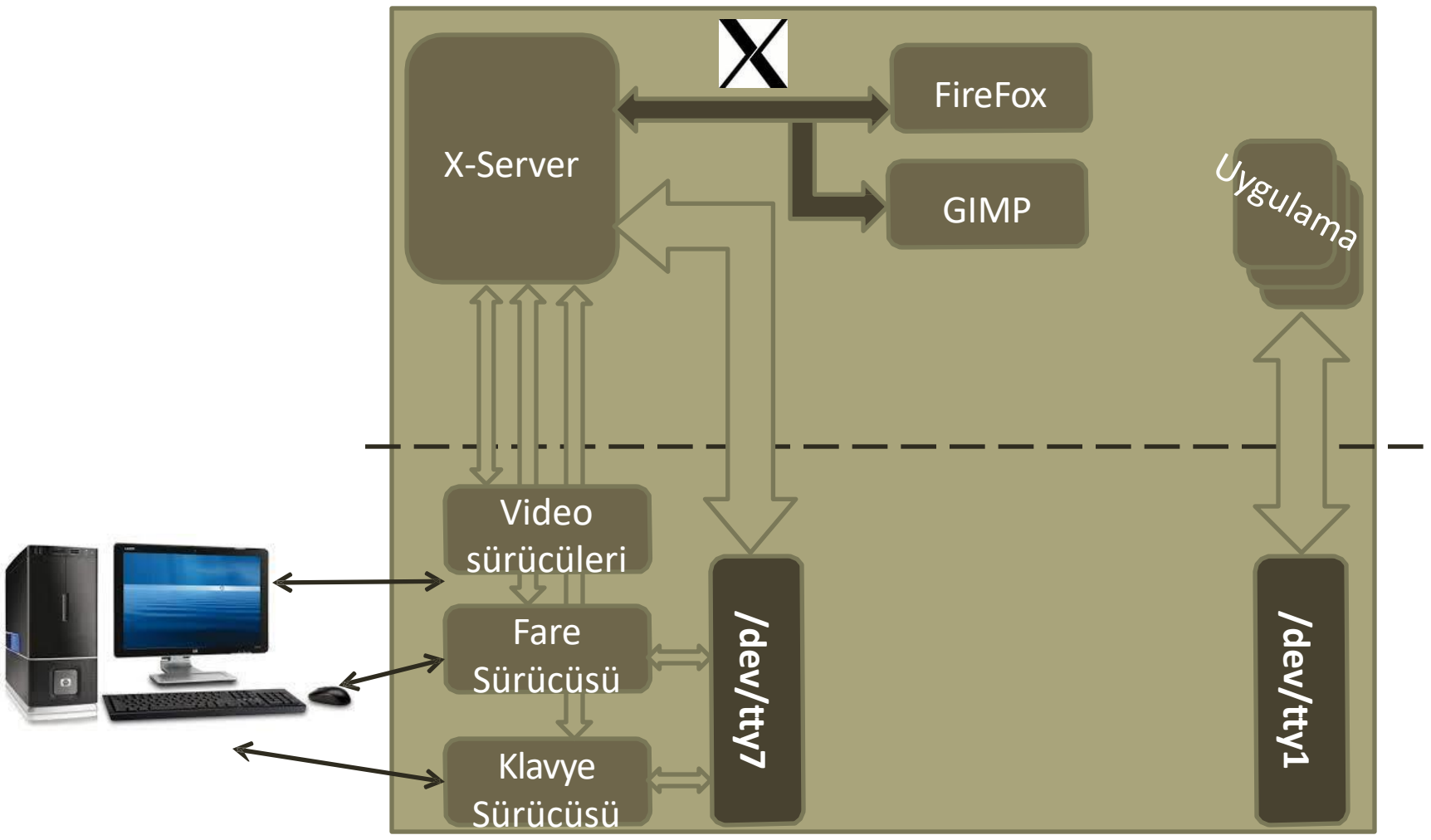
Taklit Terminal Kullanımı

Taklit Terminaller

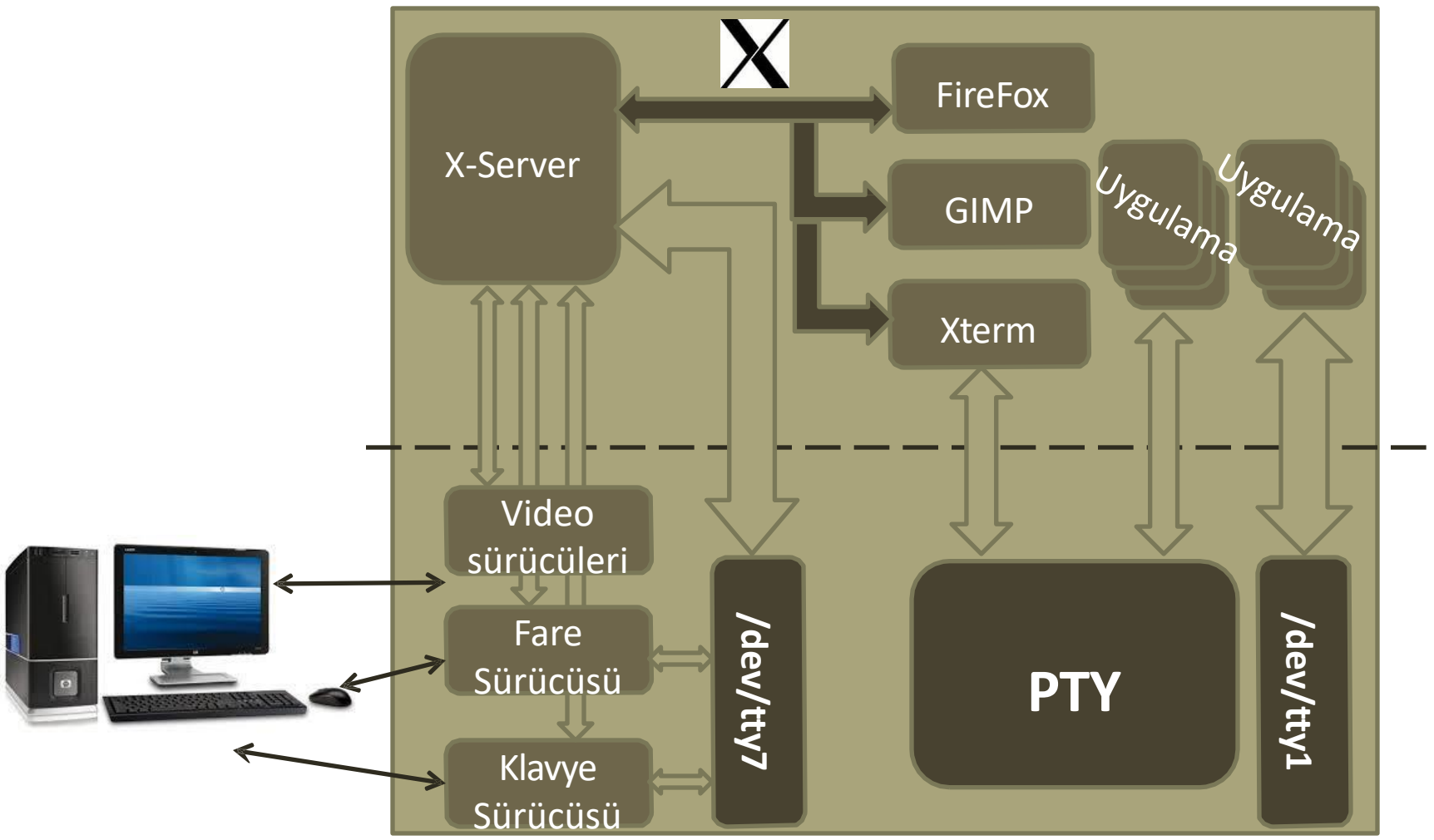
- Bir sonraki adım GUI'de taklit terminal kullanımdır.
- Bir X-app(X-uygulama) ST taklidi yapabilir.
- Bu konsepti desteklemek için, Linux Kernel'i Sözde Terminal konseptini tanıtmıştır.
- X-clients olarak çalışan bazı terminal taklitleri vardır. Örneğin;
  - ***xterm***
  - ***Konsole*** (in KDE)
  - ***gnome-terminal*** (in GNOME)

  Bu kavramı desteklemek için, Linux Çekirdeği, Sözde(pseudo) Terminaller(PTY=pseudo teletype) kavramını tanıtıyor;

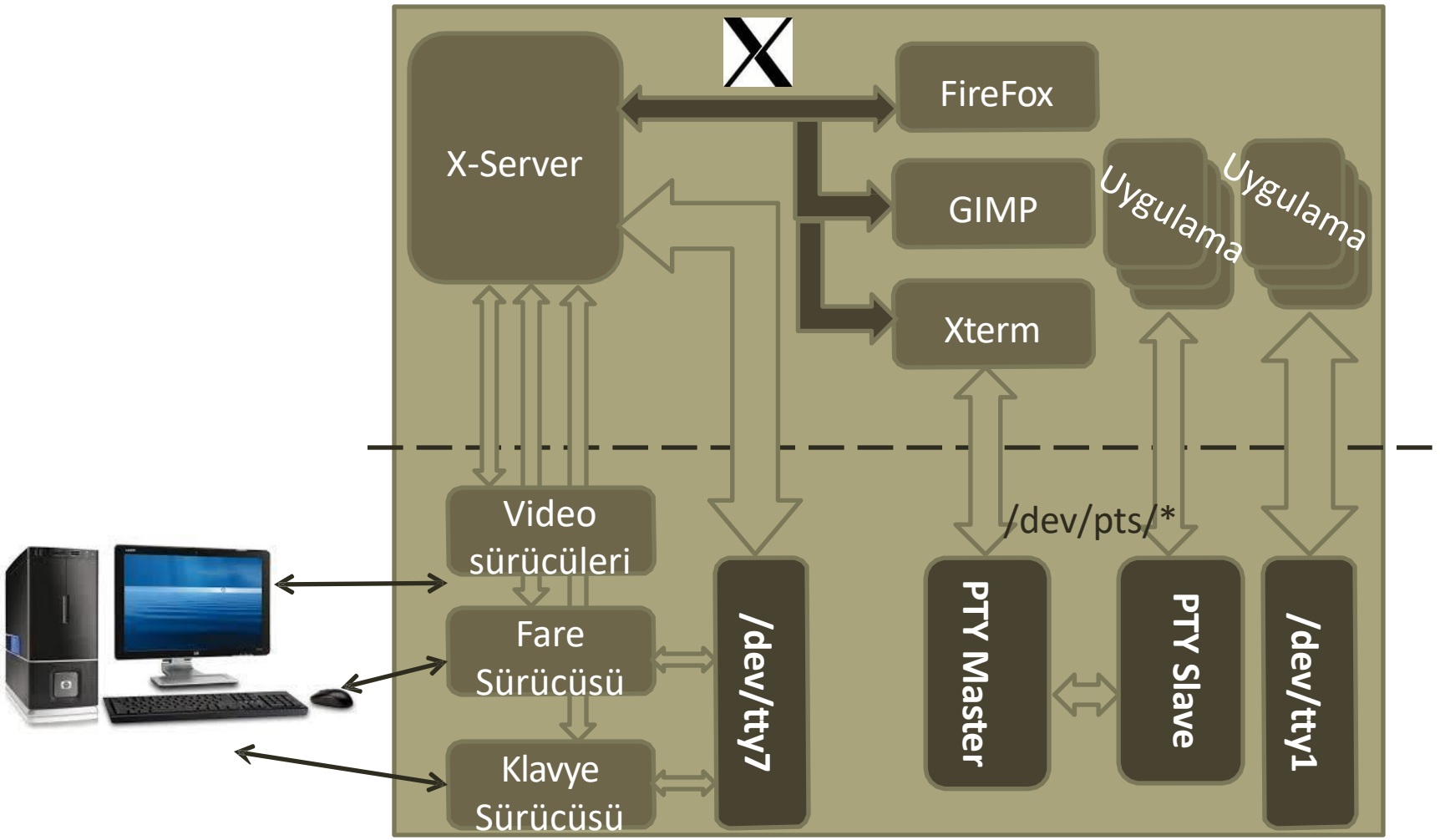
Taklit Terminal Kullanımı



Taklit Terminal Kullanımı

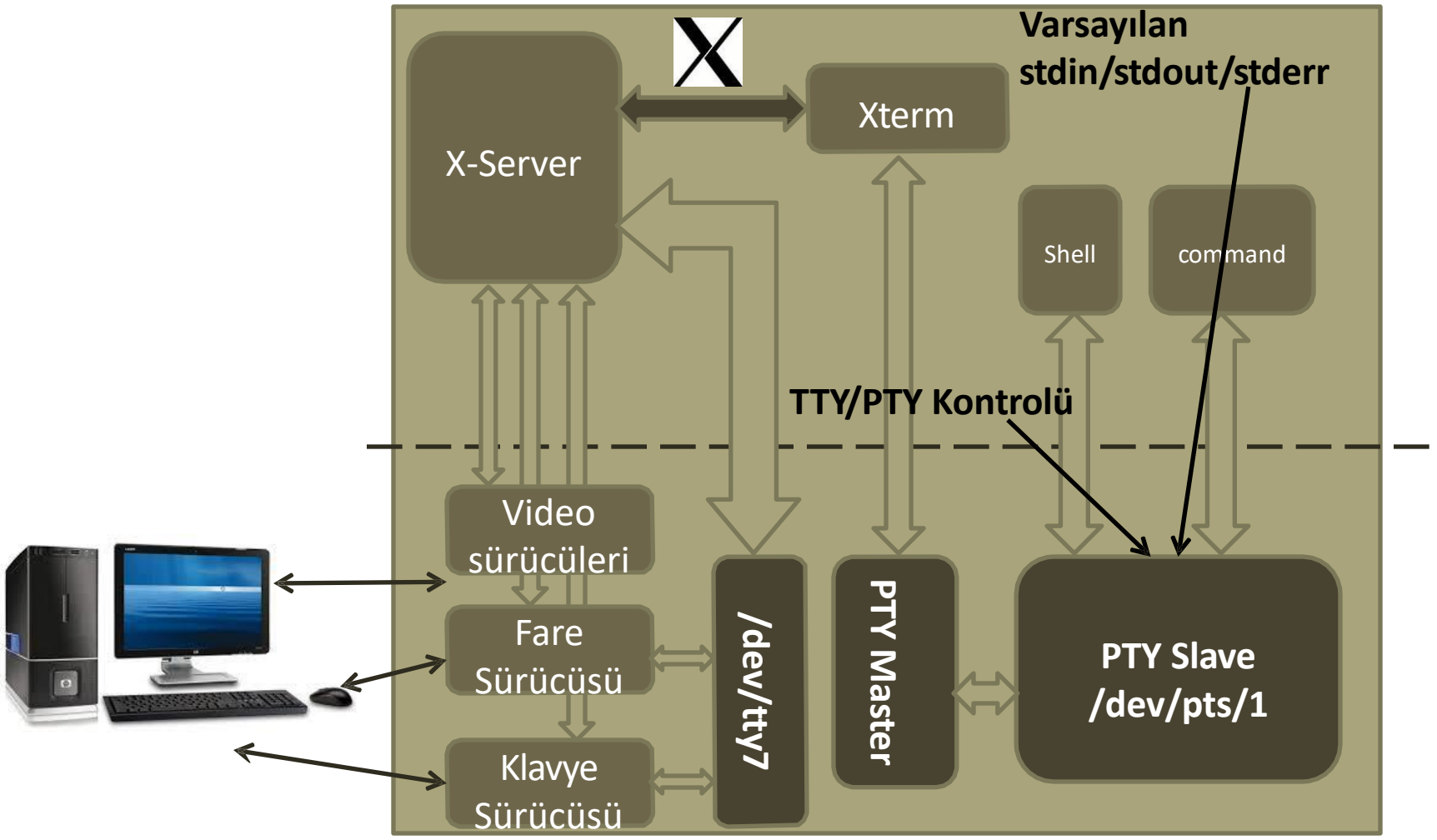


Taklit Terminal Kullanımı

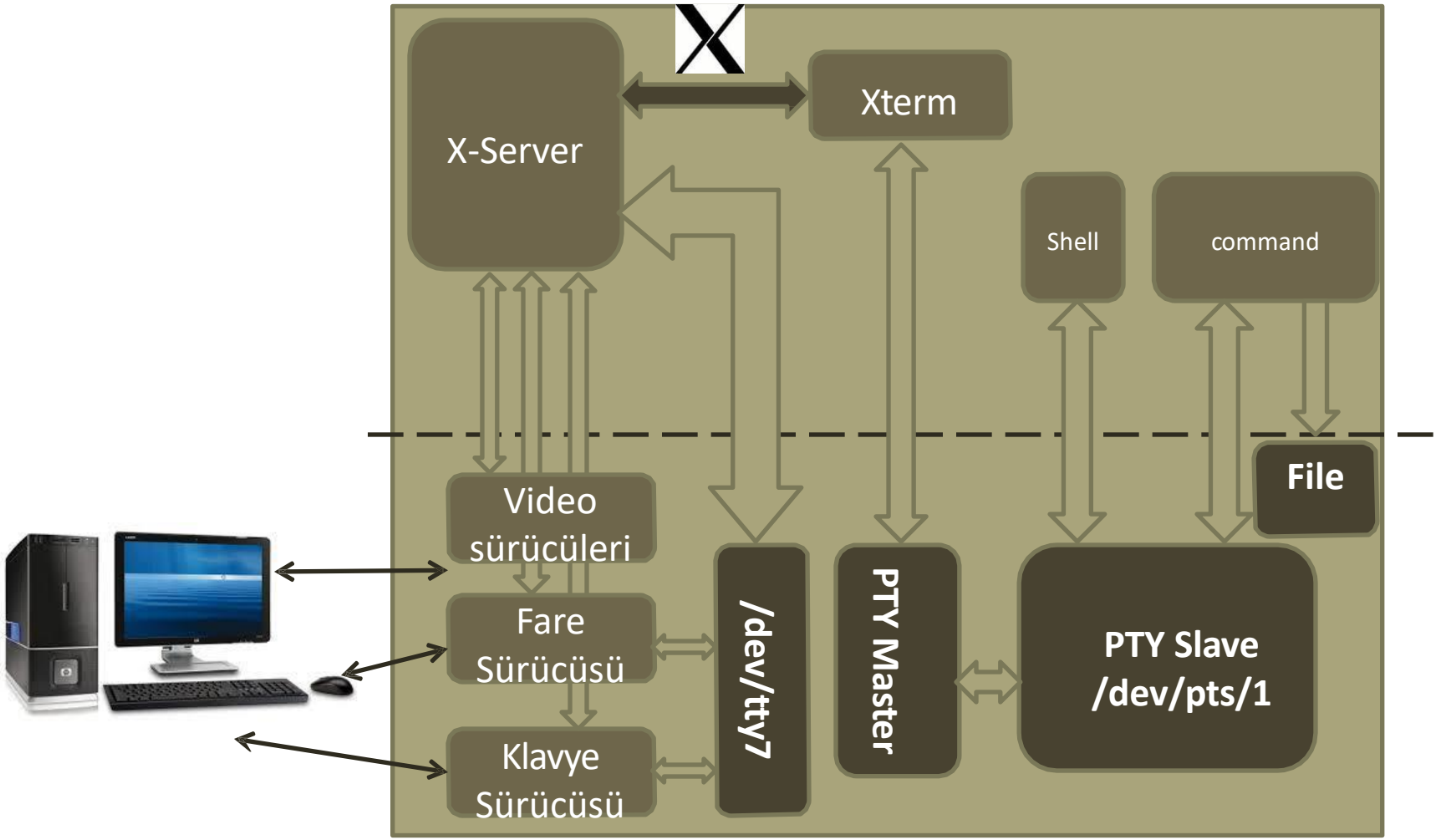


Kullanım Örnekleri

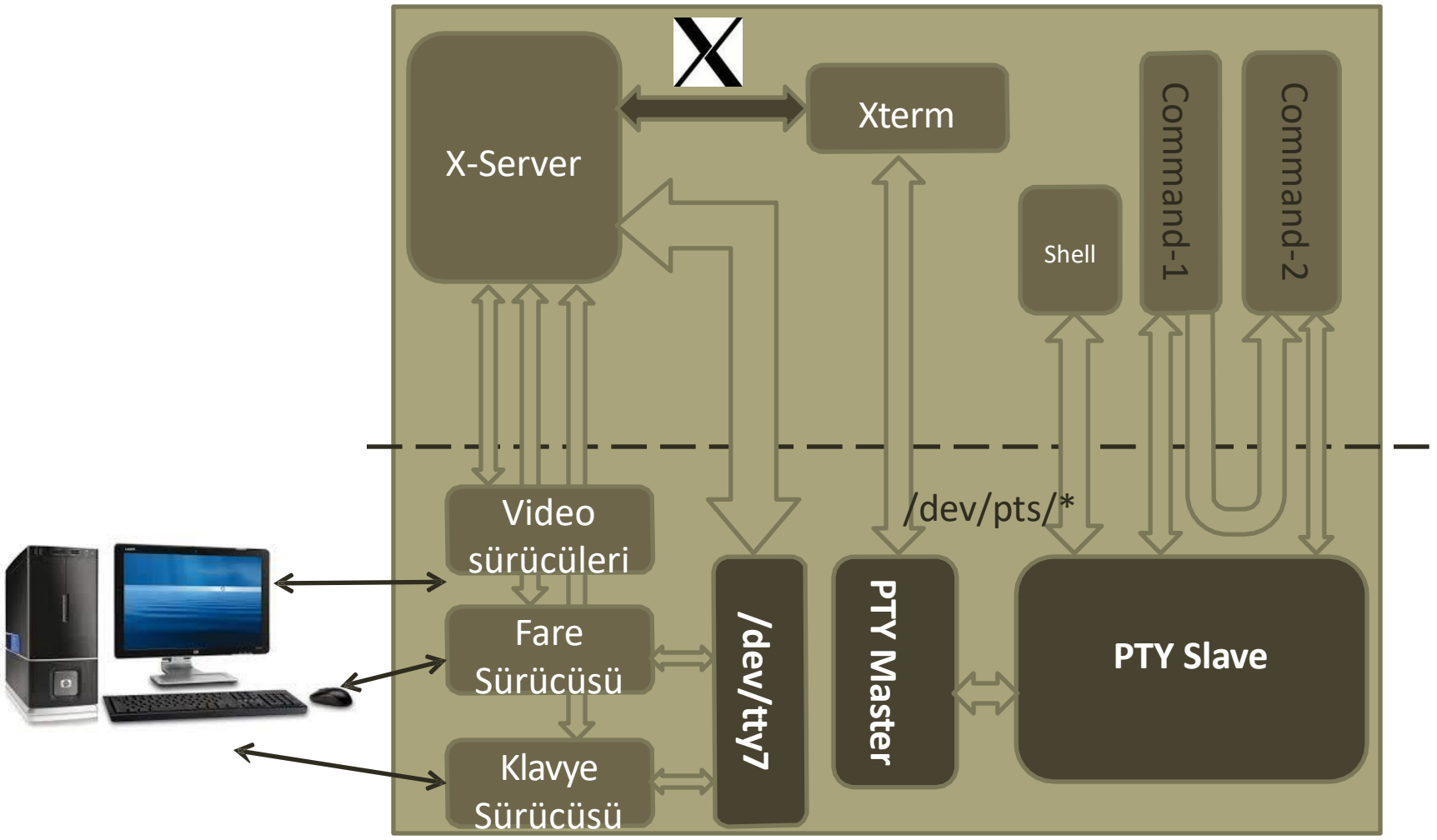
Kabuk Komutu Kullanımı



Stdout'u bir dosyaya yönlendirmek



Piped Kabuk komutu Çalıştırma



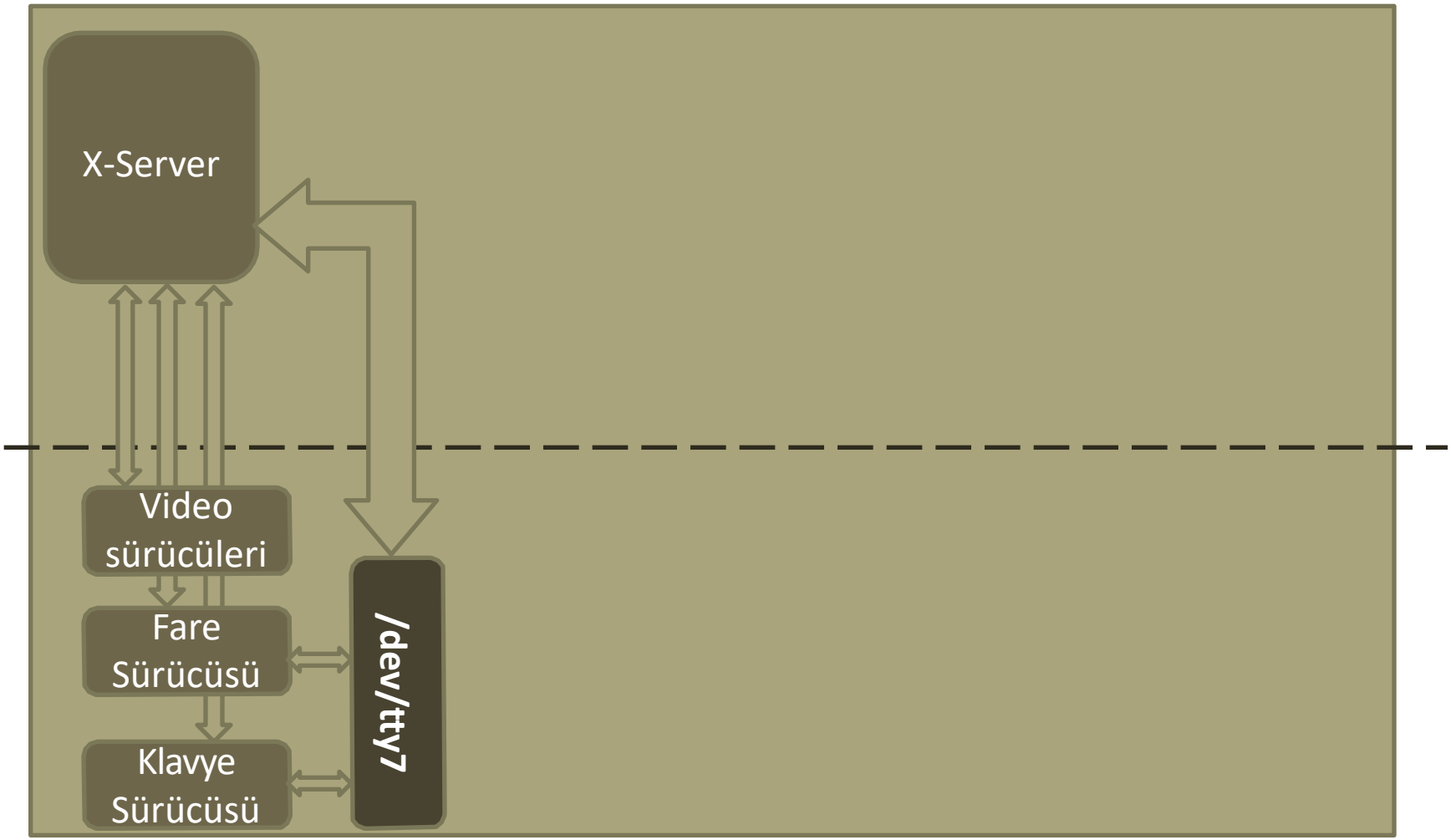
$ tty

- Eğer bu komut çalıştırılırsa (Physical/Virtual/Emulated), Bununla ilişkili TTY / PTY cihazını görüntülenir.

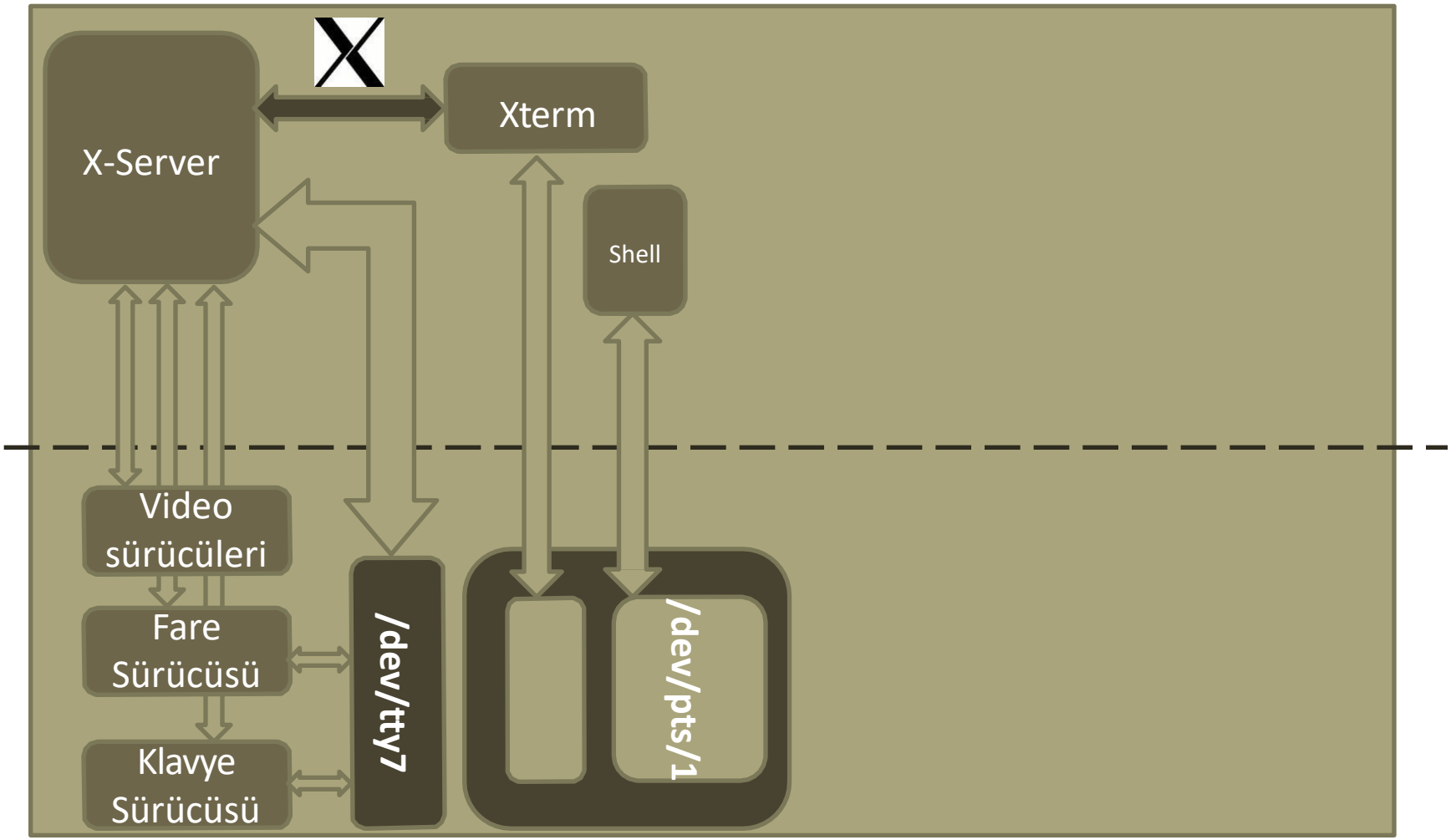
```
linuxagyonetimi@ubuntu:~$ tty
/dev/tty1
linuxagyonetimi@ubuntu:~$
```

Komutlarla Sanal Oturumları Yönetmek

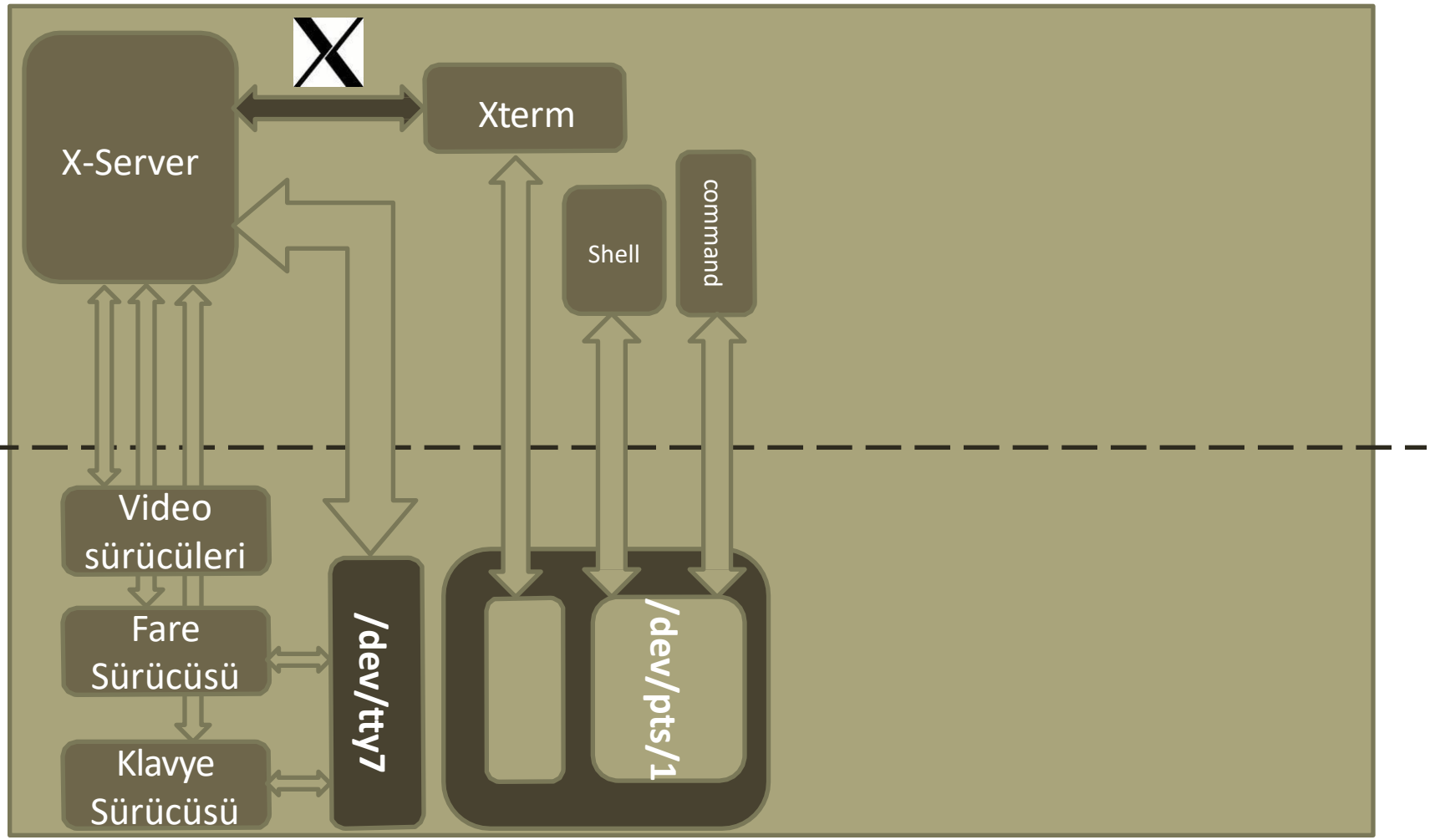
Ekran komutlarıyla çalışmak



Ekran komutlarıyla çalışmak

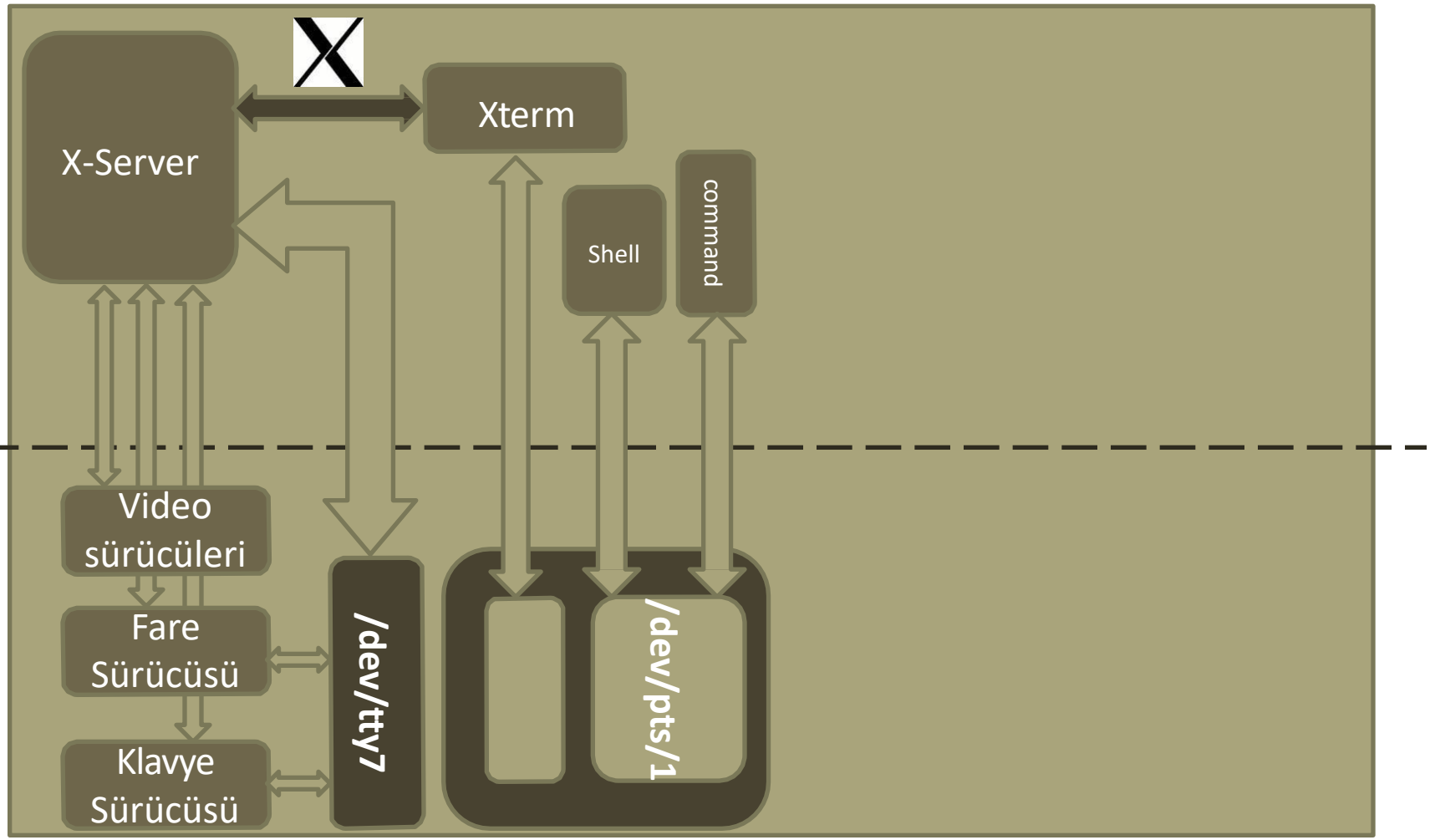


Ekran komutlarıyla çalışmak

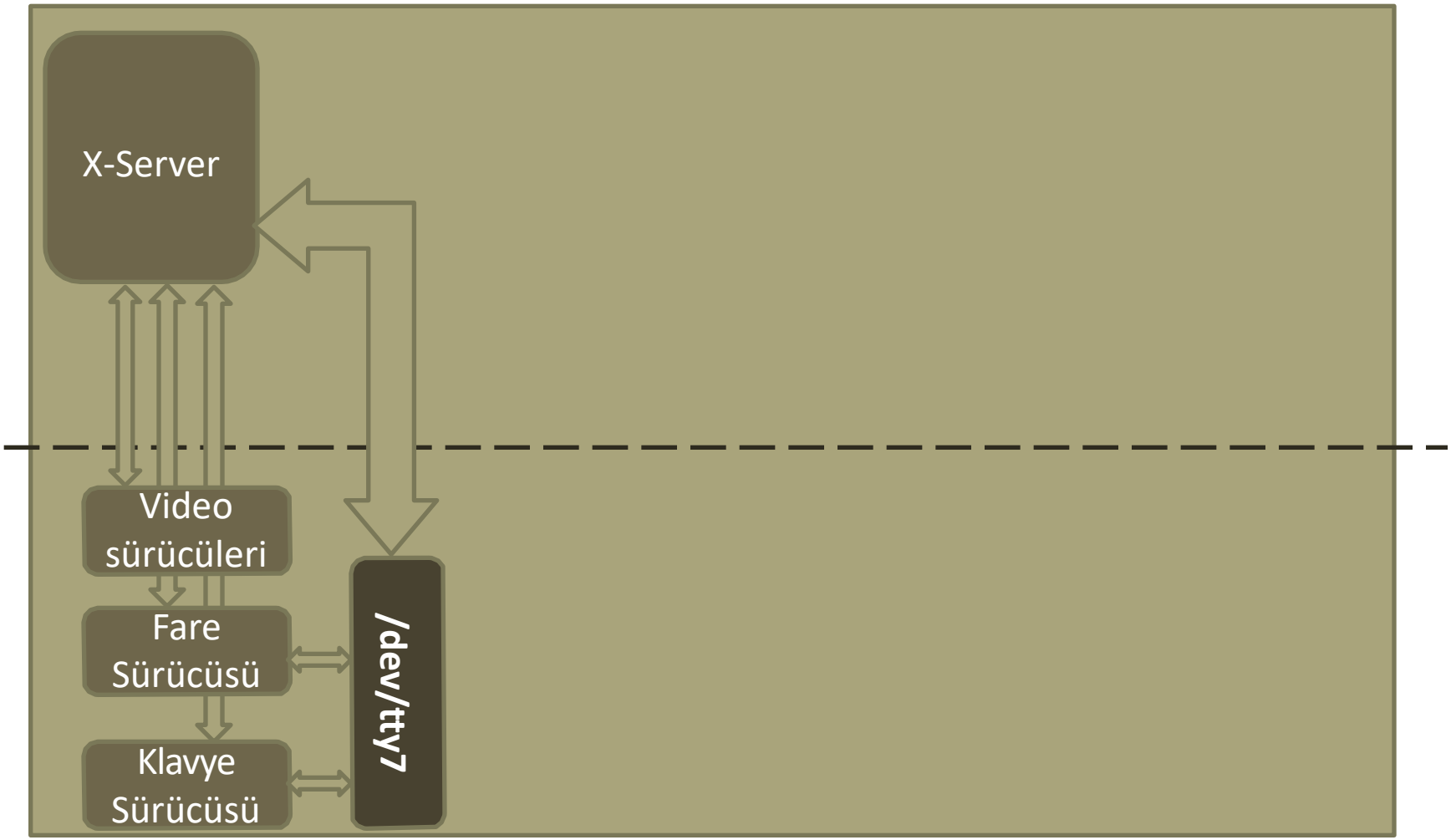


$ command

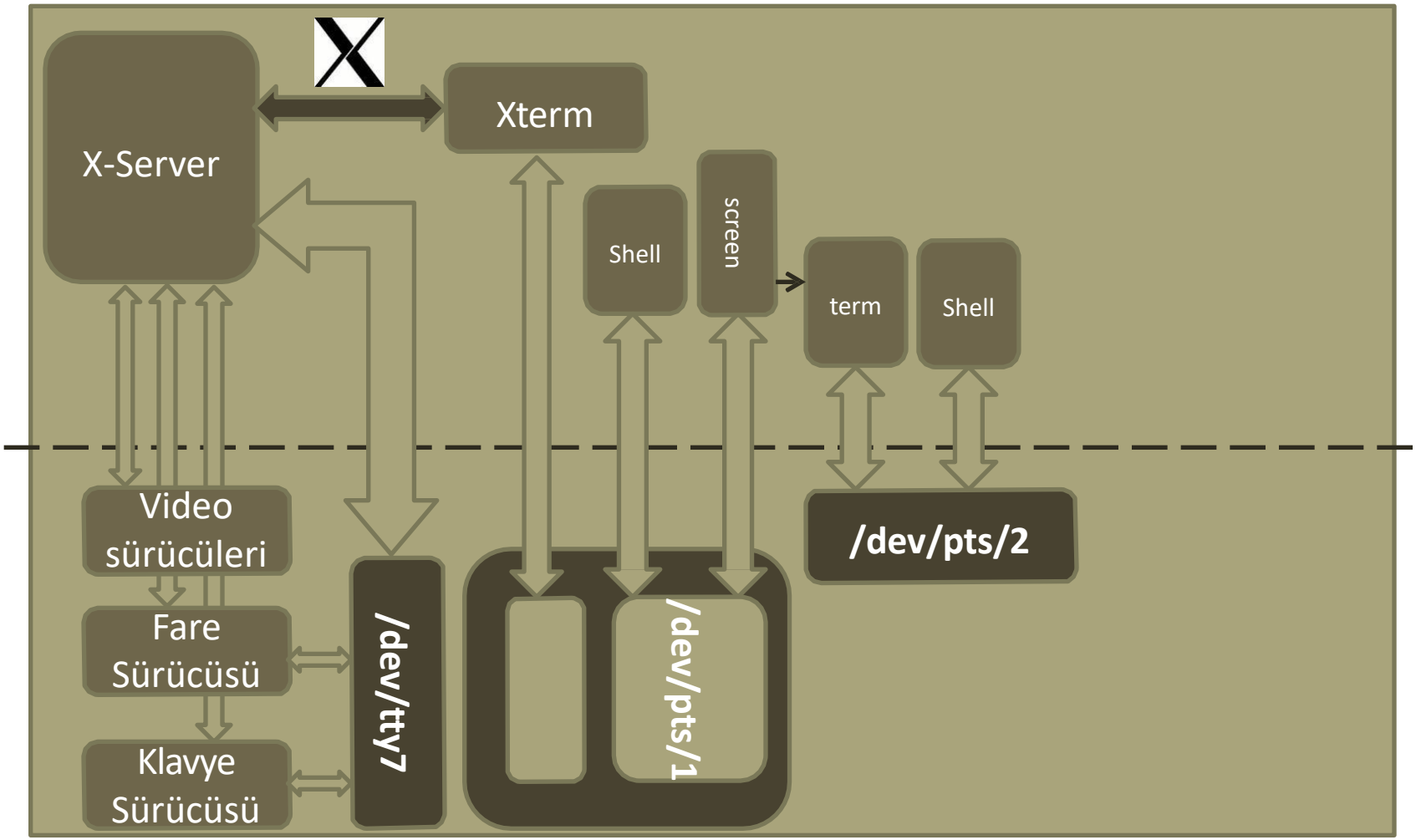
Ekran komutlarıyla çalışmak



Ekran komutlarıyla çalışmak

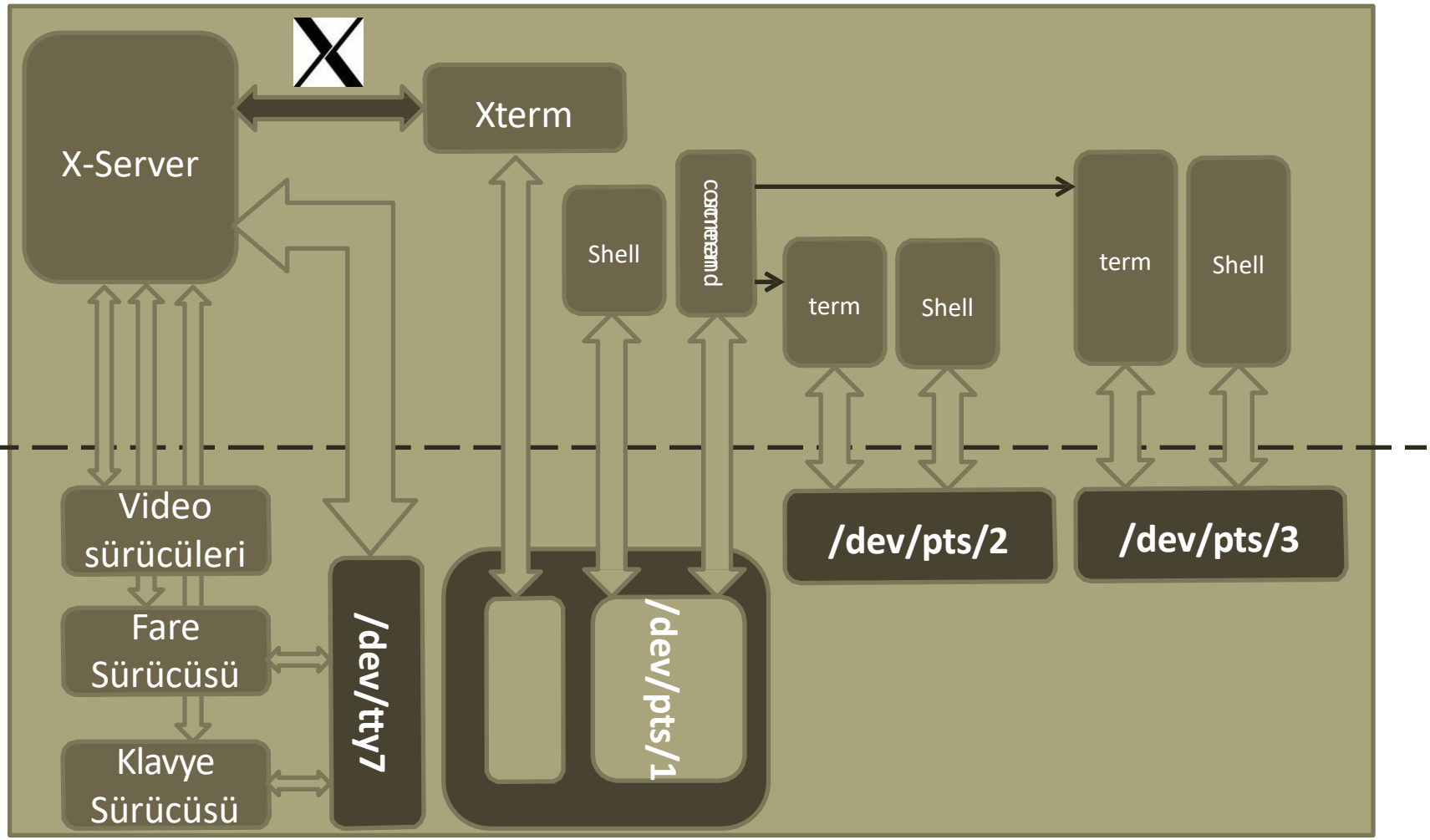


Ekran komutlarıyla çalışmak



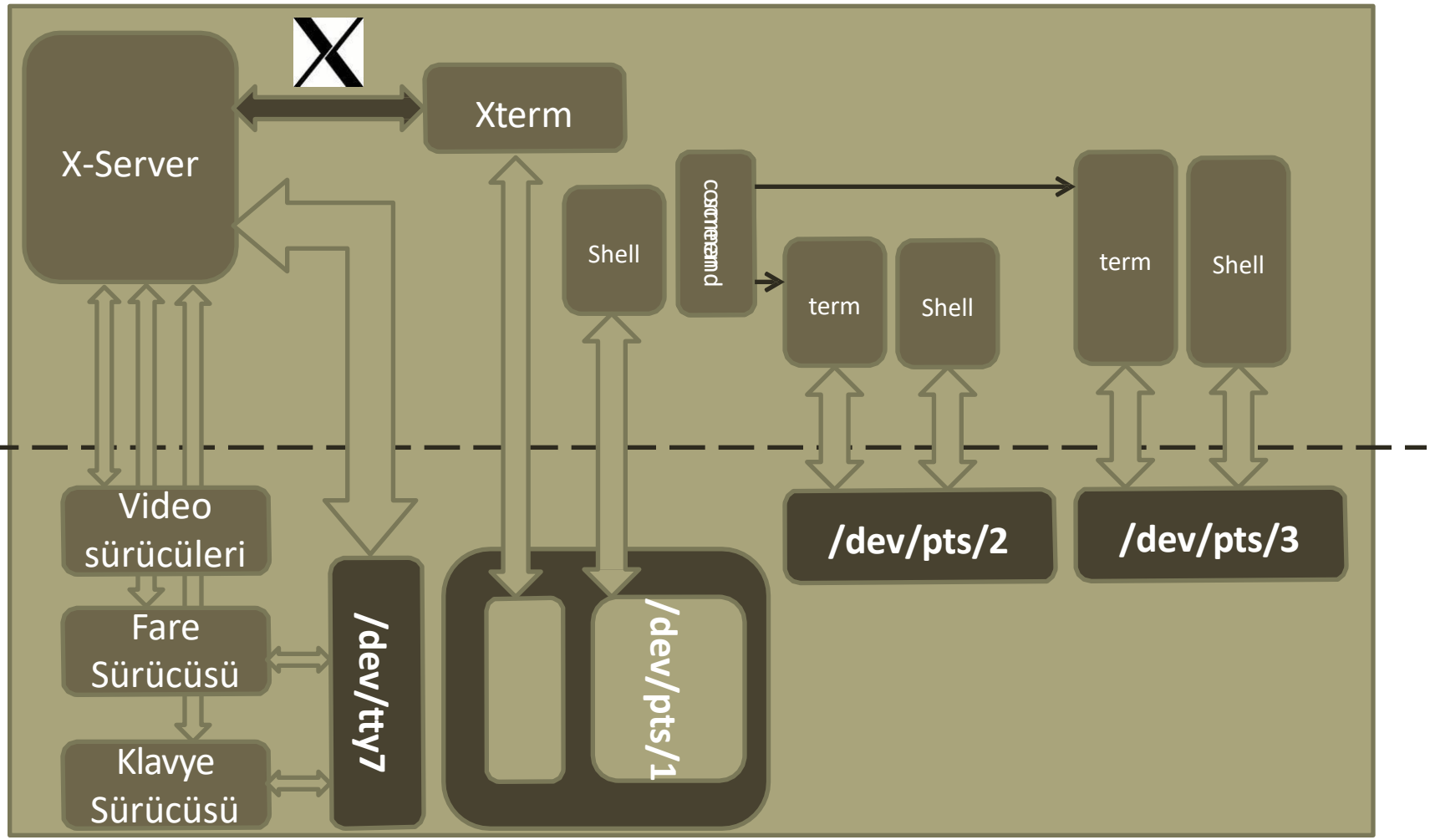
$ screen

Ekran komutlarıyla çalışmak



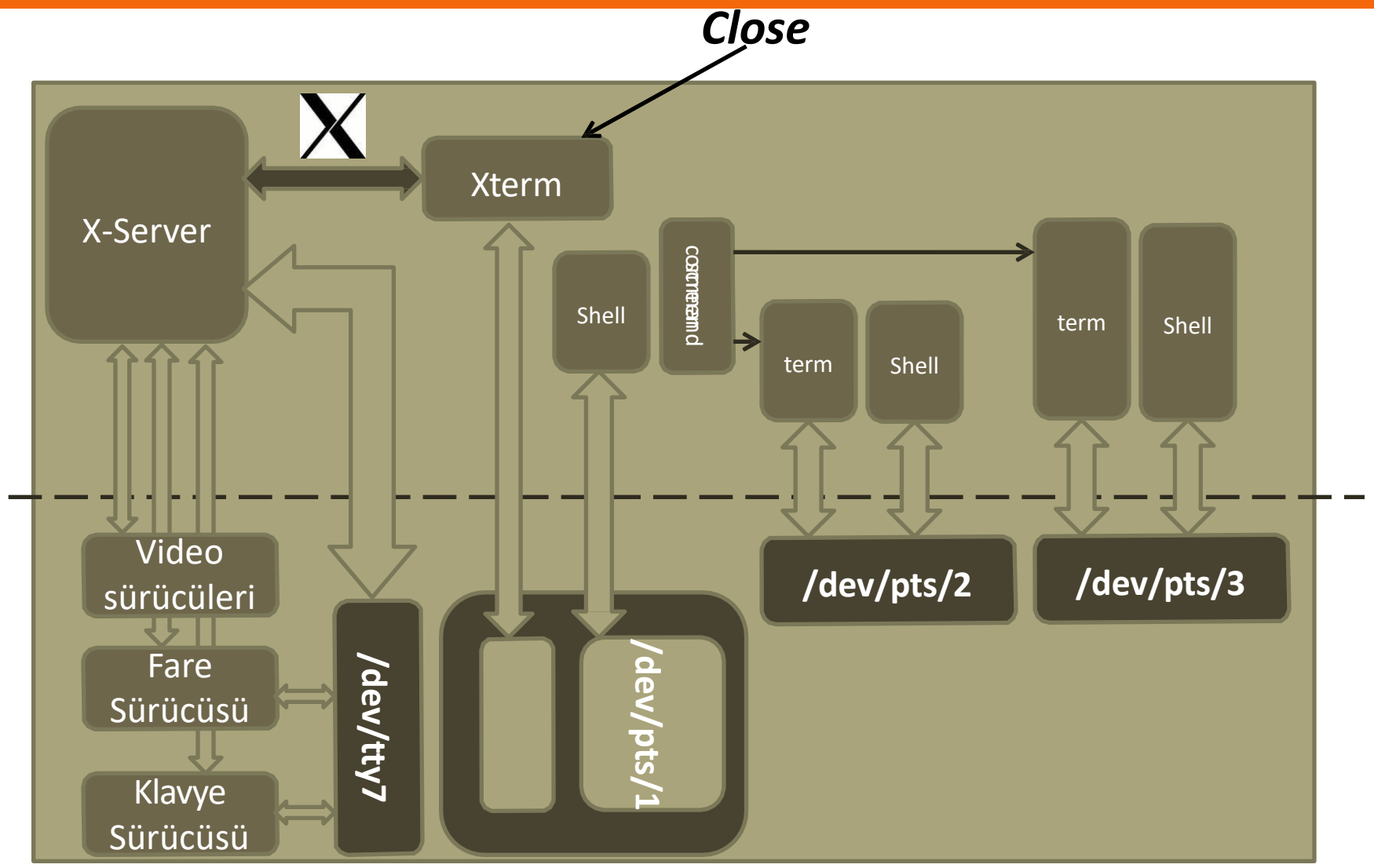
Ctrl-A c

Ekran komutlarıyla çalışmak

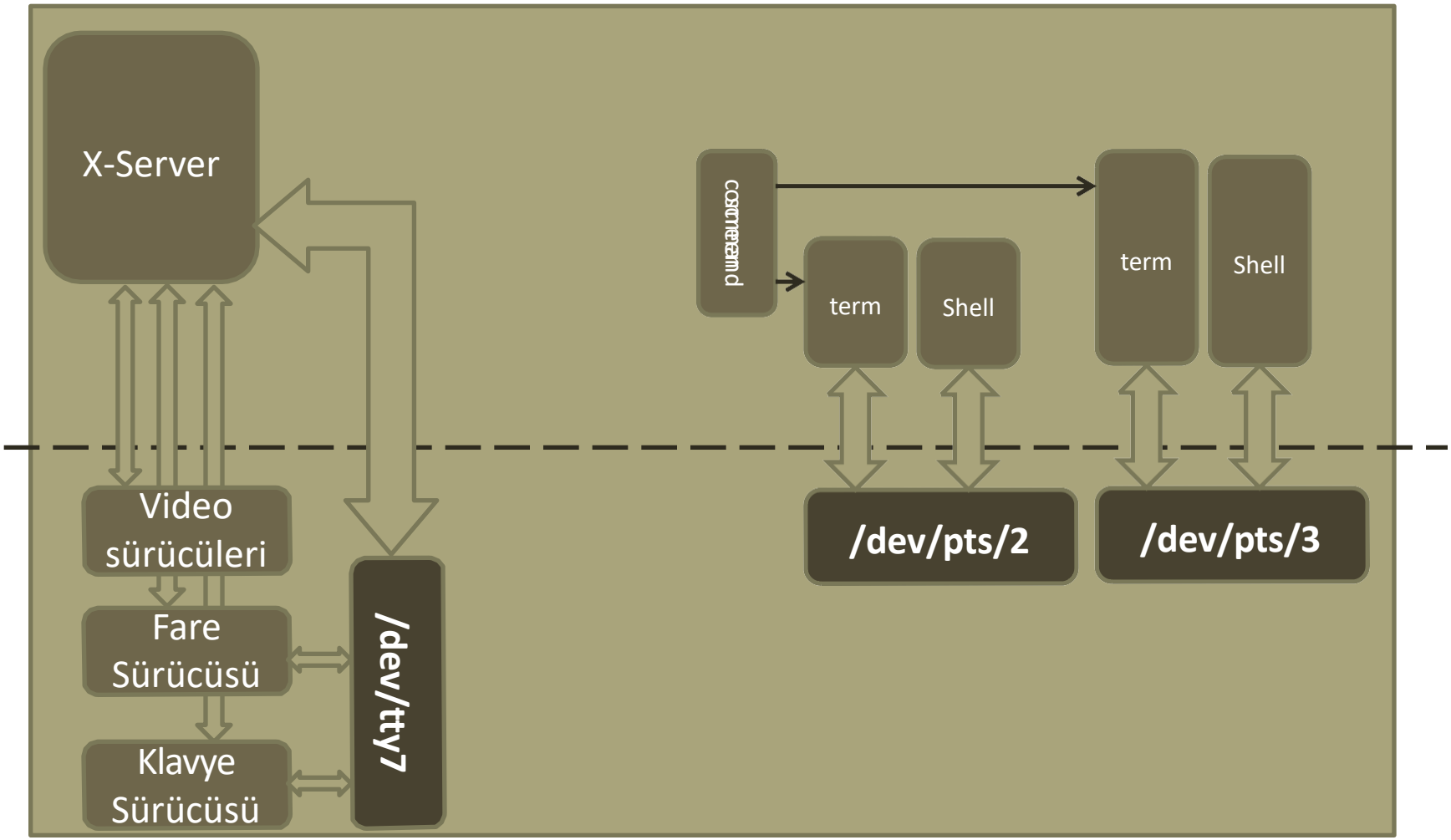


Ctrl-A d

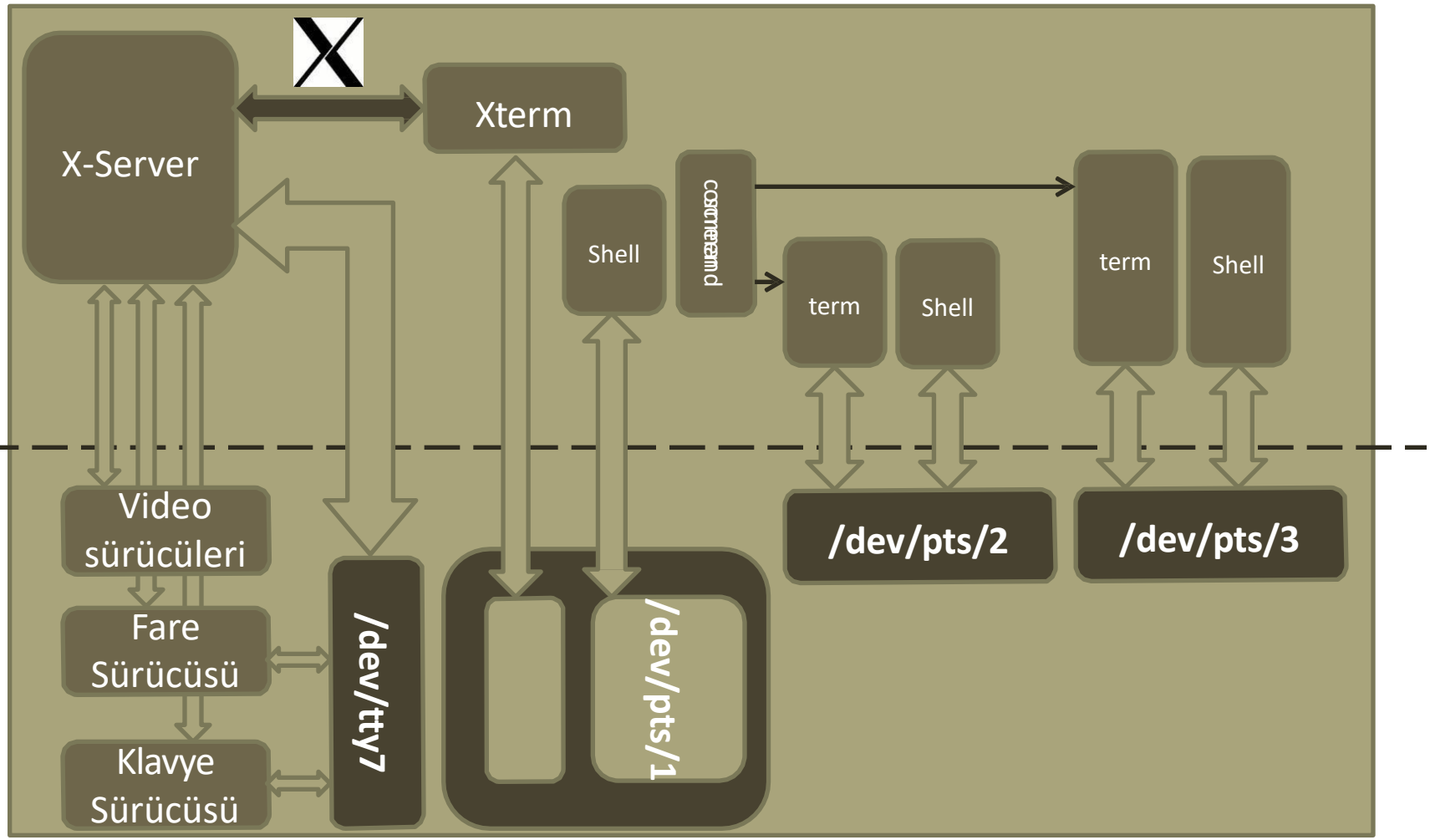
Ekran komutlarıyla çalışmak



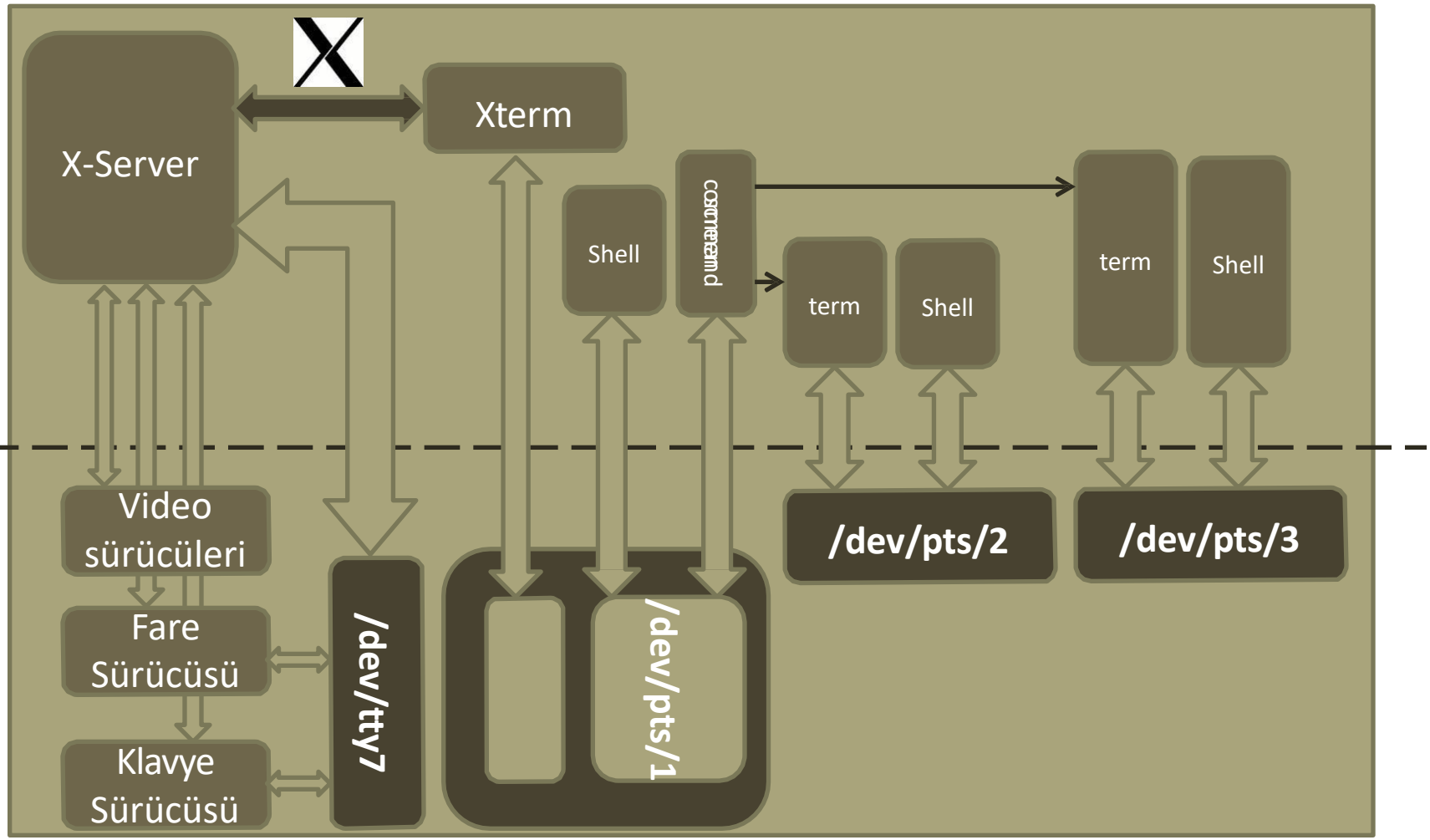
Ekran komutlarıyla çalışmak



Ekran komutlarıyla çalışmak



Ekran komutlarıyla çalışmak



$ screen -r

Komutlarla Taklit Oturum Yönetimi

$ screen

$ screen –r

- Komut ekranı taklit (sanal) oturumun yöneticisidir.
- Kullanıcının birden çok taklit terminal oturumu (***Ctrl-A c*** yeni bir oturum oluşturmak için) oluşturmasını sağlar
- Kullanıcının farklı oturuma gidebileceğini sağlar

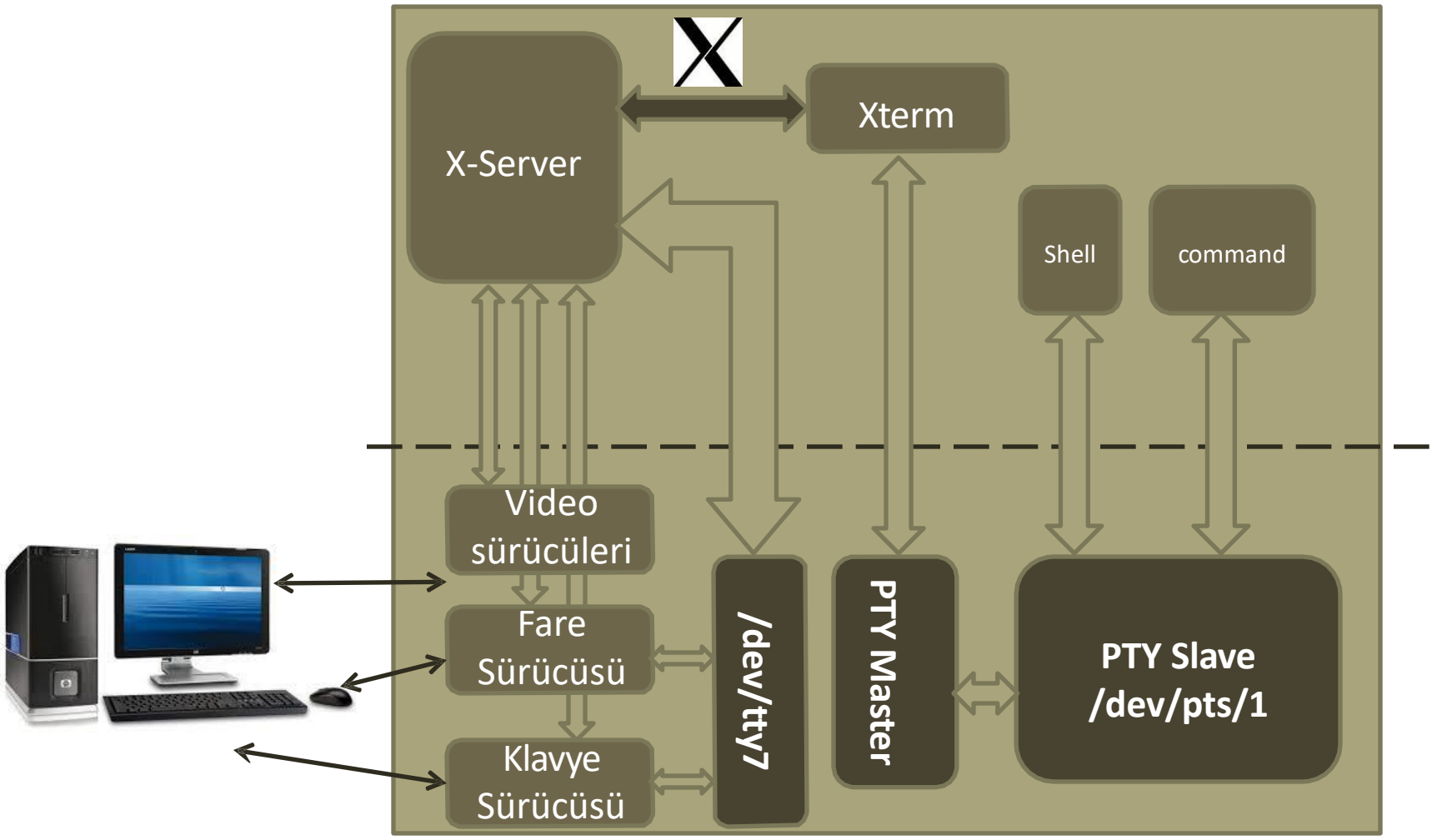
  (sonraki oturuma gitmek için ***Ctrl-A n***)
- Kullanıcı ekran komutunu kontrolünden çıkarabilir

  TTY (***Ctrl-A d***).
- Bu çok kullanışlı bir araçtır, şimdi kontrol terminali ekran altında çalışan oturumları etkilemeksizin kapanabilir
- Ekran oturumlarının kontrolünü tekrar kazanmak için

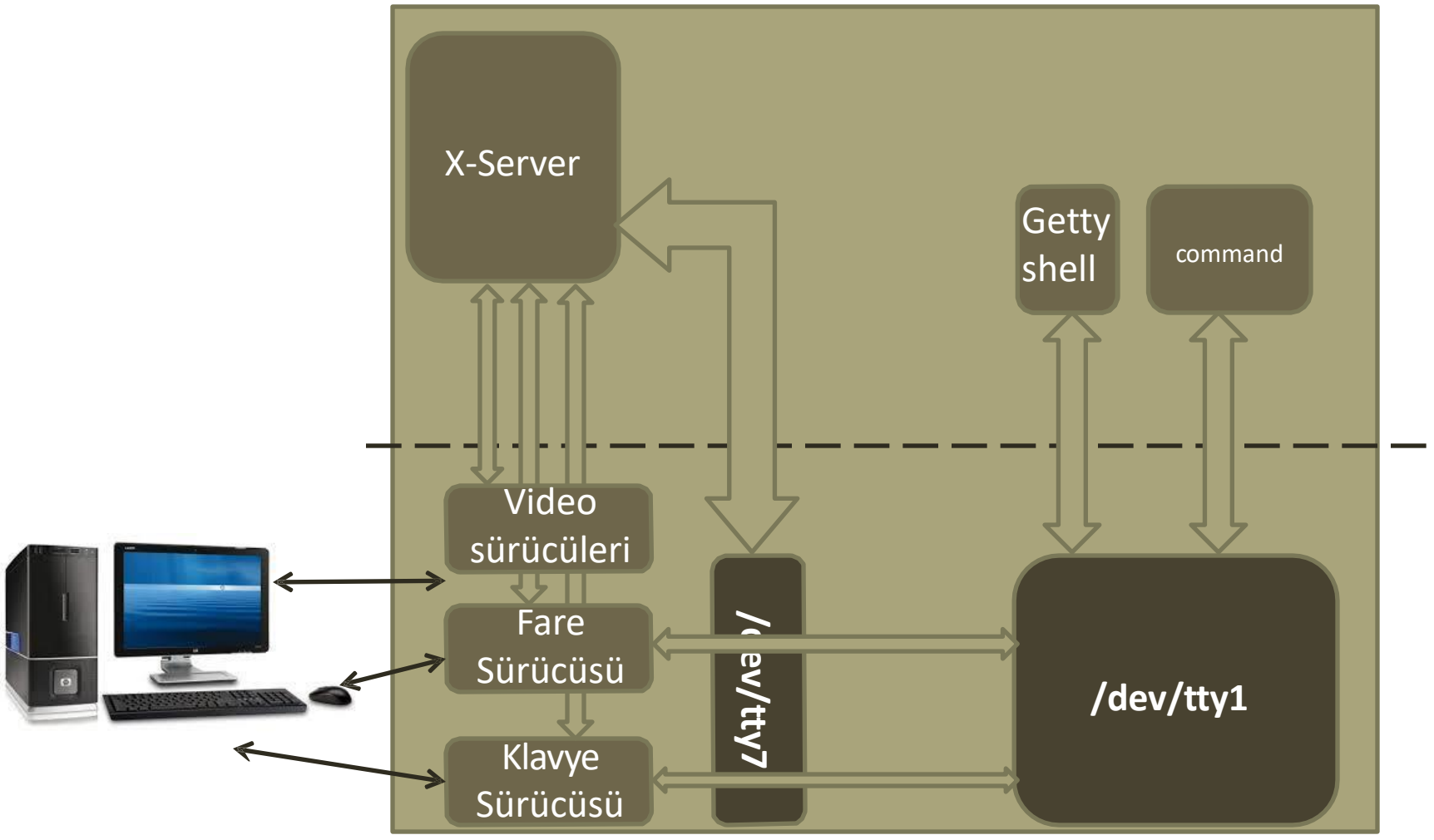
  $ screen -r

Kabuk

Kabuk Komutu Çalıştırmak

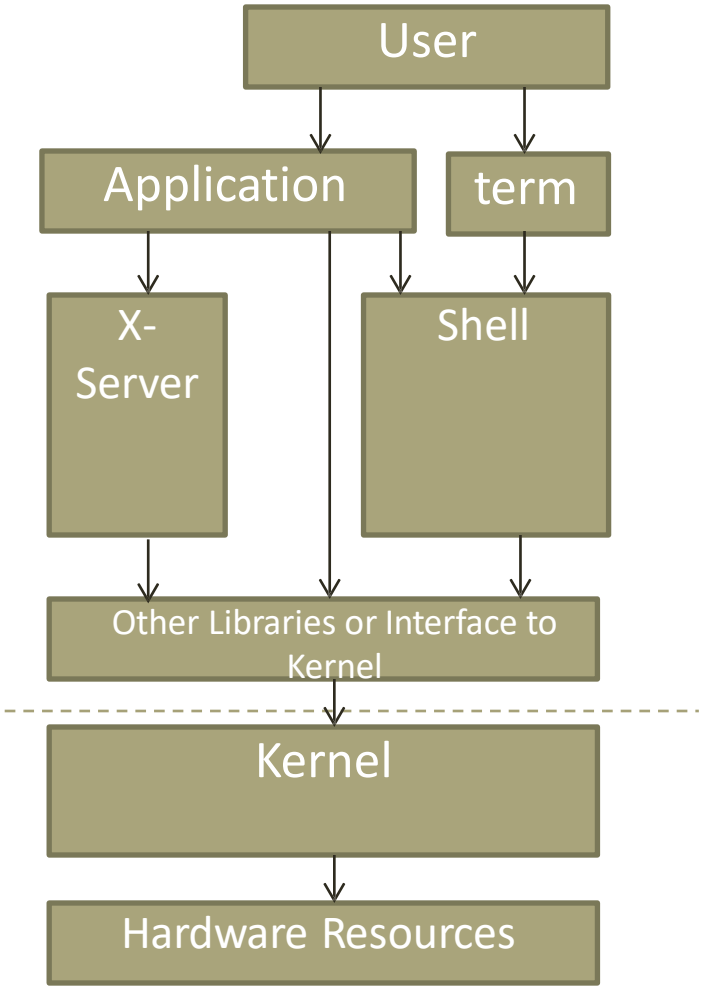


Kabuk Komutu Çalıştırmak



Kabuk(shell) Nedir?

- Shell, metin girişi kabul eden bir “user space”programıdır.
- Ayrıştırma ve girişin biraz genişletilmesini gerçekleştirir.
- Metin ayrıştırması için ***readline*** kütüphanesini kullanır.
- Bir terminal veya taklit(emulated) terminal programı aracılığıyla erişilir.
- Daha sonra denetimi uygun işlevlere geçirir (Kabuk içinde veya dışında).
- Bir kabuk sistem ve kullanıcıları arasındaki etkileşimi yönetir.
- Bir kabuk bazı yerleşik işlevlerle gelecektir, diğer işlevler ayrı ikili dosyalar tarafından sağlanacaktır



Kabuk Türleri

- İlk Kabuk, Bourne kabuğu olarak da bilinen sh kabuğudur.
  (Şu anda artık kullanılmıyor)
- Bunu takip eden csh kabuğudur(c-benzeri kabuk),
  (bu da artık kullanılmıyor)
- tcsh var ama bu da çok popüler değil

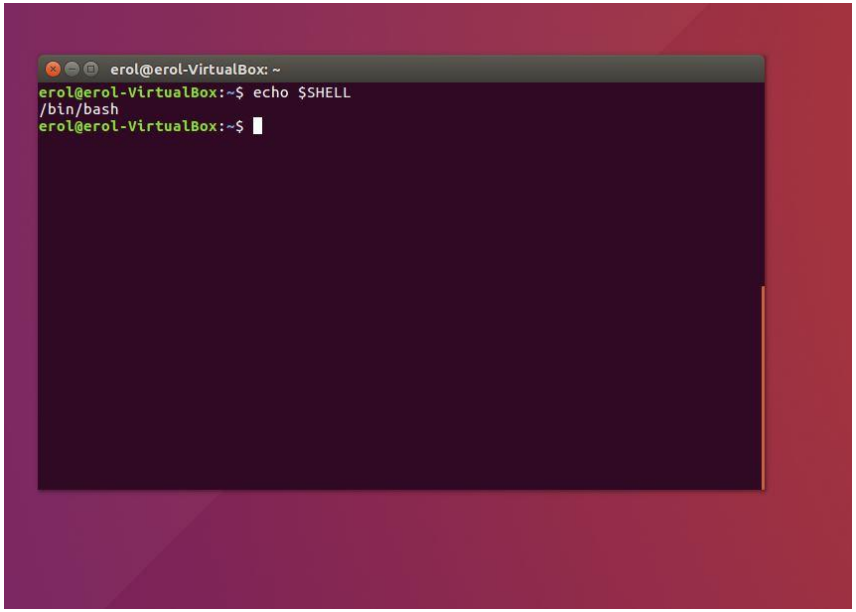
- En popüler kabuk (ve kullanacağımız kabuk),
  bash shell'dir. GNU projesinin bir parçasıdır

Kabuk Türleri

- Desteklenen kabuklar / etc / shells dosyasında listelenir
- Kullanıcı için varsayılan kabuk / etc / passwd'da belirtilecektir.
- Farklı bir kabuğa geçmek için kendi ikili dosyasını çağırın, örneğin farklı bir kabukla çalışıyorsanız ve bash kullanmak istiyorsanız,
- ***$ bash***

  Ardından, orijinal kabuğa,

  $ exit
- Hangi kabuğu kullandığınızı bilmek için,
- ***$ echo $SHELL***



Komut Kategorileri

Komutlar aşağıdakilerden biri olabilir

1. Kabuk programı içinde yerleşik komut(‘***cd***’ gibi)
2. Kabuk programı tarafından çağrılan sistemdeki bir ikili veya çalıştırılabilir. Bu ikili erişilebilir olabilmek için PATH içinde olmalıdır (normalde / usr / bin'de bulunurlar)
3. Başka bir komuta Alias (nasıl yaptığımızı göreceğiz)
4. Yada bir kabuk fonksiyonu olabilir (detaylı olarak bash scripting konusunda anlatılacak)

Komut kategorisi tanımlama

$ Yaz <Komut>

<Komut> komut kategorisini tanımlar

- Örnekler:

  $ type cd

  cd is a shell builtin

  $ type rm

  rm is /bin/rm

  $ type ls

  ls is aliased to `ls --color=auto`

Yerleşik Komutlar

- Bunlar kabuk ikili dosyasında uygulanan işlevselliklerdir.
- Bunun için ayrı bir ikili yoktur.
- Çok sınırlı set ve yalnızca çok basit komutlar içindir.

- Örnekler,

$ cd

$ pwd

Ayrı bir ikili olmadığından, bunu yapamazsınız

$ cd --help

Bunun yerine yerleşik komuta komutunu kullanabilirsiniz

$ help cd

Ayrı İkili Komutlar

- Bu, Linux dizin yapısında yer alan ayrı bir programdır.
- Çoğu komut bu kategoriye aittir.
- Her komutun kendi sürümü vardır.

  $ rm --version

- Komut kullanımını almak için

  $ rm --help

- Komut için ikili bulmak için

  $ which rm

Aliases

- Aliases başka bir komutun bir kısaltmasıdır (seçenekler / bağımsız değişkenlerle)

- Aliases komutunu yapmak için
- ***$ alias yenikomut='uzun komut'***

  Örneğin

  $ alias ll='ls -al'

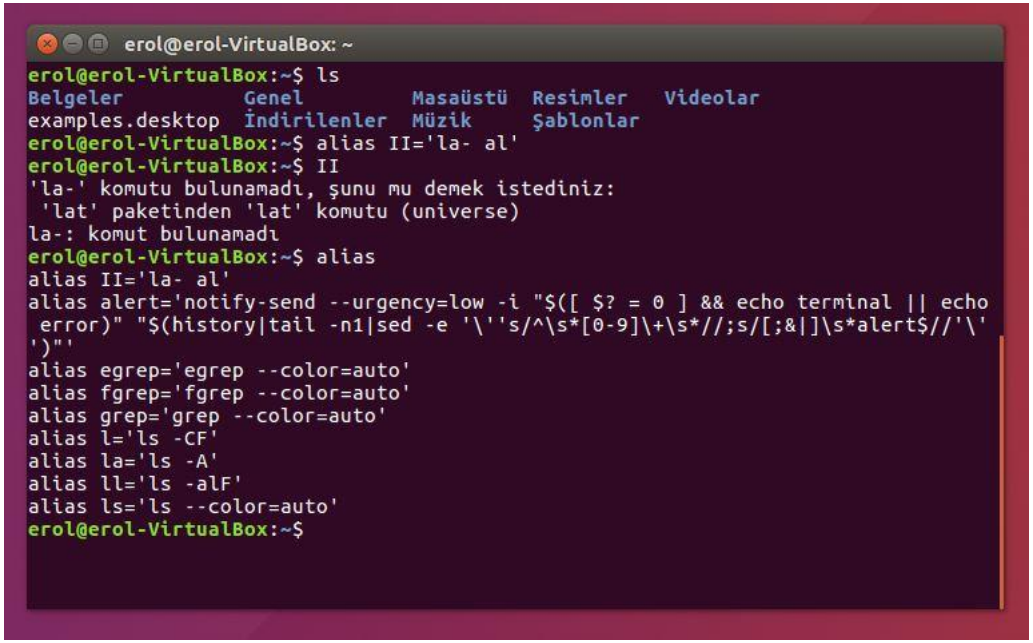
- Bir Aliases komutu kaldırmak için

  $ unalias <alias command>

  Örneğin:

  $ unalias ll

- Tüm Aliasesları listelemek için
- ***$ alias***



Aliases

- Yeni komut kullanılmış bir komut ise, yine de çalışacaktır (yeni komut eskisini **geçersiz** kılacaktır)

  $ alias ls= 'ls --color=auto'

- Bunu yanlışlıkla yapmadığınızdan emin olun, takma ad vermeden önce bir komutun varlığını takma adla kontrol edin

  $ type <alias candidate>

  (Sana bulamadığını söylemeli)

Komut Geçmişi

- Bir komutu girdiğinizde, Linux bunu bir geçmiş dosyasına depolar ***~/.bash-history***
- Komut geçmişine göz atmak için

  $ history
- Şimdi şunları yapabilirsiniz:

  $!! (son komutu girmek için)

  $! <n> (geçmişte # n komutunu girmek için

  $! abc (‘abc' ile başlayan son komutu girmek için)

  $ ^abc ^def (son komutu girin, ancak 'abc' yerine 'def' yazın)

  $ command2 !* (command2 yi son komuttan gelen tüm bağımsız değişkenlerle çalıştırın)

  $ command2 !$ (command2'yi son argümanla son komuttan çalıştırın

  $ command2 !^ (command2yi son komuttan yalnızca ilk argümanla çalıştırın)

Komutları bir dosyaya yazma (script Command)

$ script <file>

Komutları ve çıktılarını bir dosyaya yazmak
için

$ script file

$ script -a file (Varolan dosyayı ekle)

$ script -t file (Her komutun önüne zaman damgası koyar)

$ script -f file (Her komutun ardından boşaltın)

Kabuk Türleri

- Kabuk iki kategoriye aittir
  - Giriş olan kabuğu
    - Başlatmadan önce giriş yapmayı gerektiren kabuk
  - Giriş olmayan kabuğu
    - Giriş yapmayı gerektirmeyen kabuklar
    - Giriş kabuklarının girişleridir
- Kabuktan çıkmak için,
  - Giriş olan kabuğu için

    $ logout
  - Giriş olmayan kabuğu için

    $ exit

Neden kabuk(shell) diyoruz???

Kabuk Başlangıç-Kabuklar Girişi

Giriş Kabuğu başlatıldığında, uygun ortamı ayarlamak için bu adımlar geçer

- System Wide Configurations
  - Genel yapılandırmalar için ***/ etc / profile***'i okuyor (tüm kullanıcılar için)
- User Specific Configurations
  - Ardından, kullanıcıya özel olarak aşağıdakilerden birini okur extensions/overrides(ilk dosyadan başlayarak dosyalardan birini bulursa, dosyayı okuyacak ve kalan dosyayı atlayacak)

    ~/.bash-profile

    ~/.bash-login

    ~/.profile
  - Bu dosyaların normalde ***~ / .bashrc***'yi dahili olarak çağırdığını unutmayın

Kabuk başlangıcı– giriş olmayan kabuklar

Giriş Olmayan Kabuklar çevreyi aşağıdaki gibi oluşturur;

- İlk önce, çevreyi parent giriş olan kabuklardan devralırlar.
- Bunun üzerine, aşağıdaki yapılandırma dosyalarını okur;
  - Genel ayarlar için dosyaları okurlar:

    /etc/.bashrc

    /etc/bash.bashrc
  - Kullanıcıya özel ayarlar için,

    ~/.bashrc

~ / .bashrc güncellemesi

- Her kullanıcı kendi ayarlarını ~ / .bashrc içine koyabilir orneğin
  - Ortam değişkenlerini ayarlama
  - Aliases komut ayarlama
  - Kabuk fonksiyonlarını tanımlama
- ***~ / .bashrc***'deki yeni ayarlar yalnızca kabuk başlangıçta okunduğu için şimdiki kabukta etkili olmayacaktır
- çözüm,
  - Yeni bir kabuk başlat
  - ***~ / .bashrc*** okumasını manuel olarak ayarla

~/.bashrc Script Çalıştırma (Kaynak komutu)

- Normalde, komut dosyaları komut isteminden çağırarak çalıştırılır

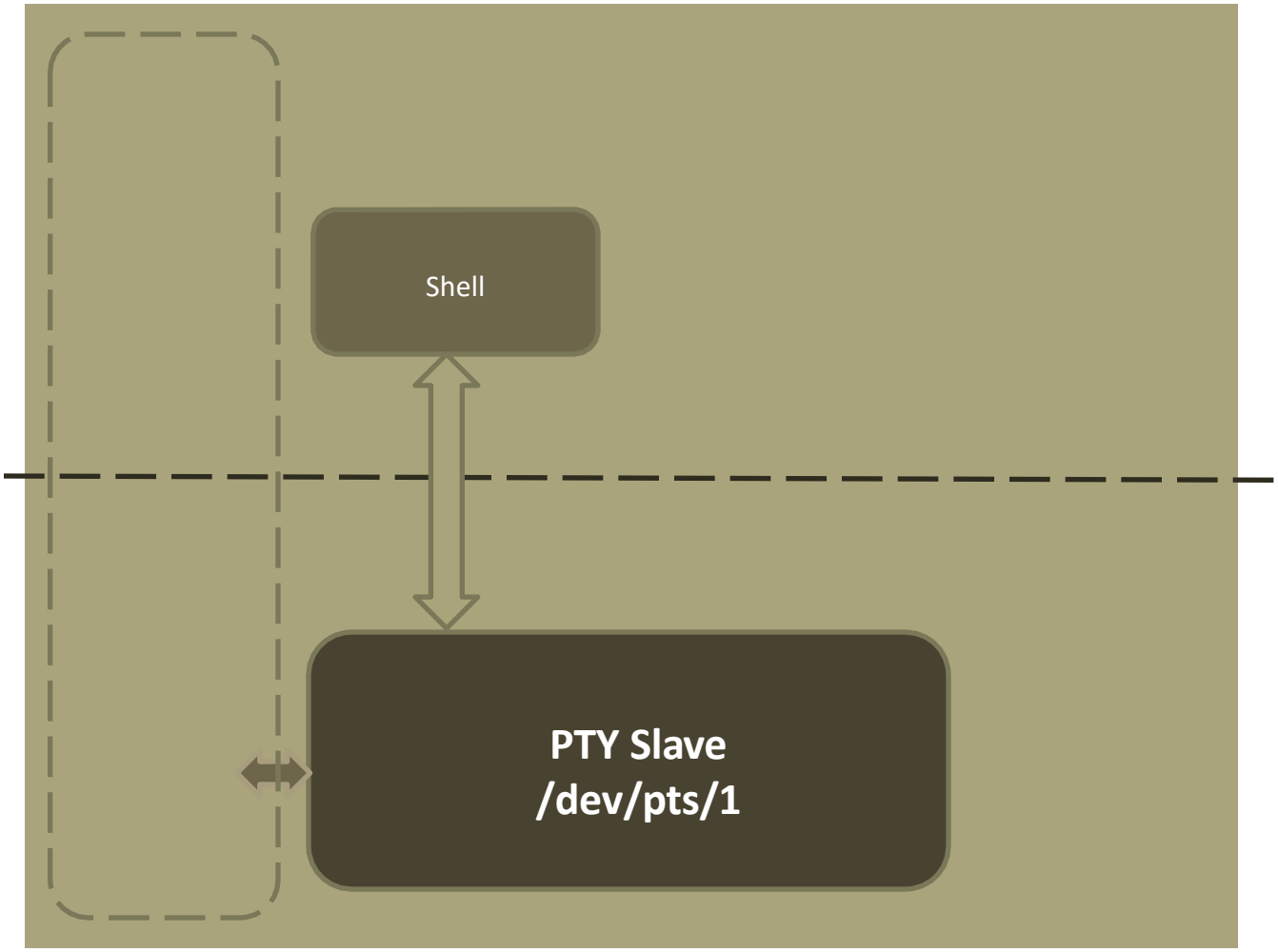
  $ <script name>

- ~ / .bashrc komut dosyasını normal komut dosyası gibi çağıramayız ... NİYE ??
  - Bir komut dosyası çalıştırıldığında, bir alt kabukta çalışır
  - Komut dosyası tamamlandığında alt kabuk kapanır ve denetim orijinal kabuğa geri döner
  - komut dosyasında ayarlanan herhangi bir şey alt kabuk için geçerli olacağı anlamına gelir ve kapatıldığında bu ayarlar kaybolur
- Komut dosyasını bir kabuk kabuğunda değil de geçerli kabukta çalıştırmaya zorlamanın yeni bir yoluna ihtiyacımız var, bu nedenle ayarlar geçerli kabuk için geçerli olacak

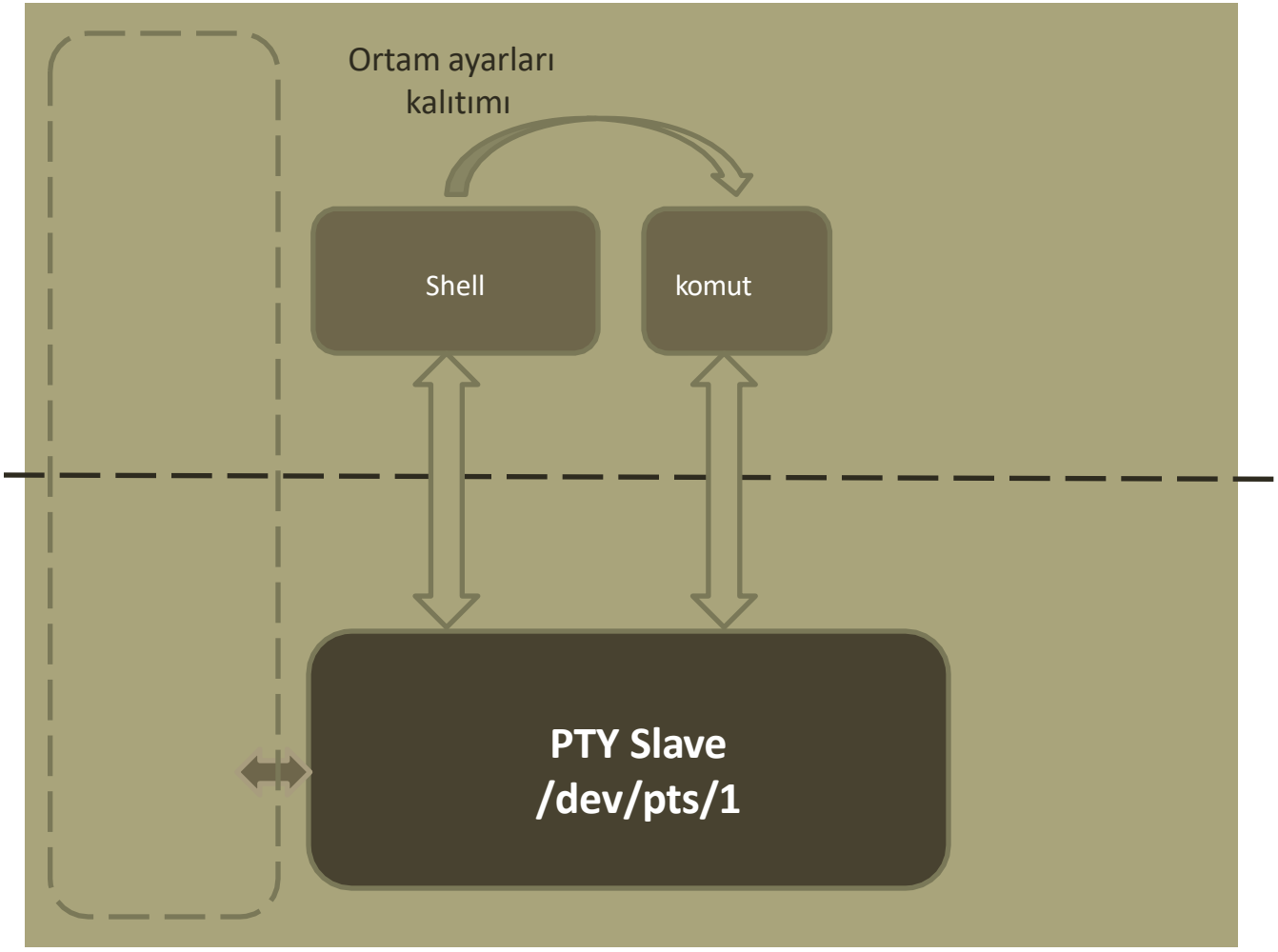
  $ source .bashrc

  $ . .bashrc

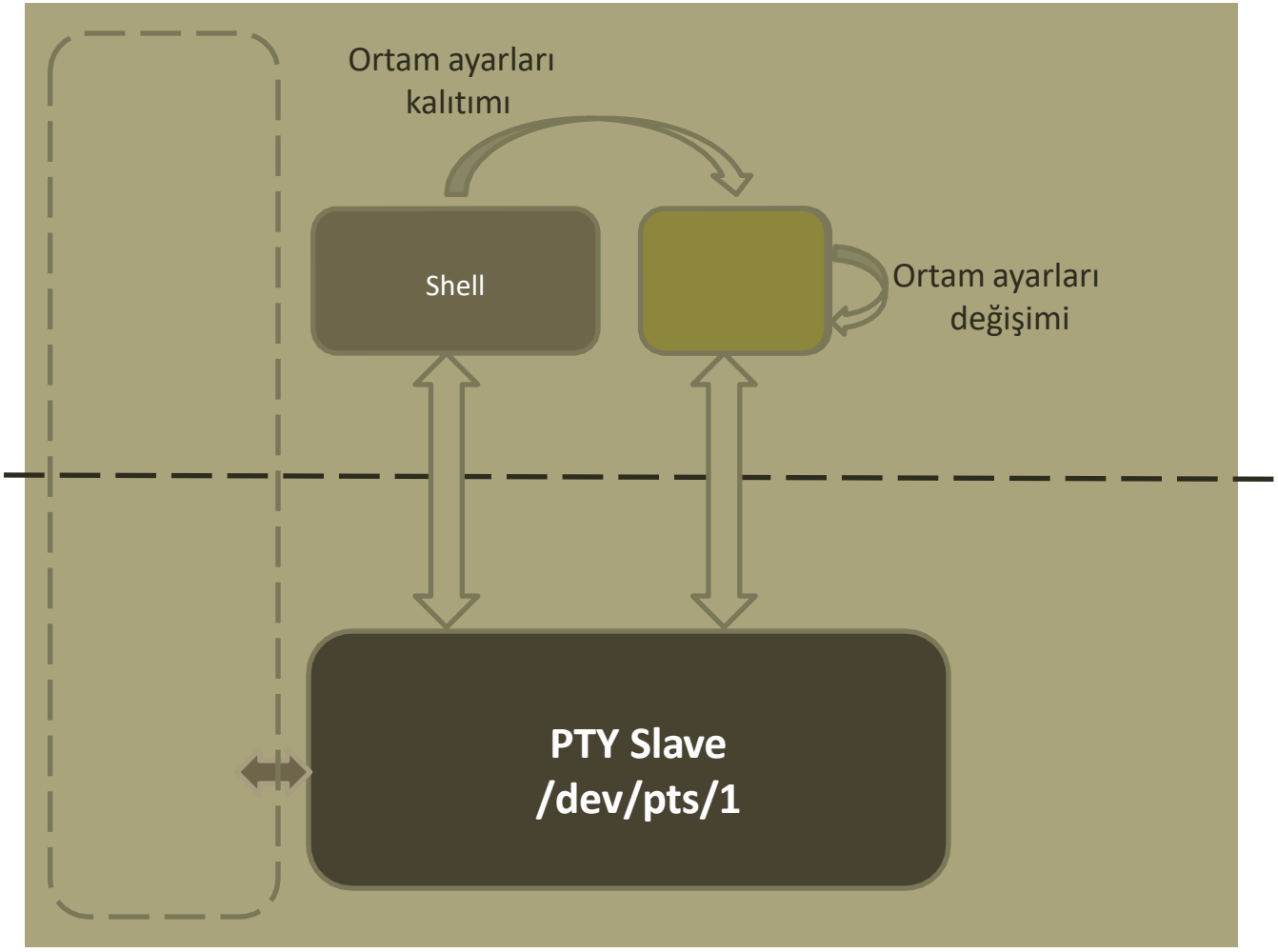
Kabuk Komut Çalıştırma



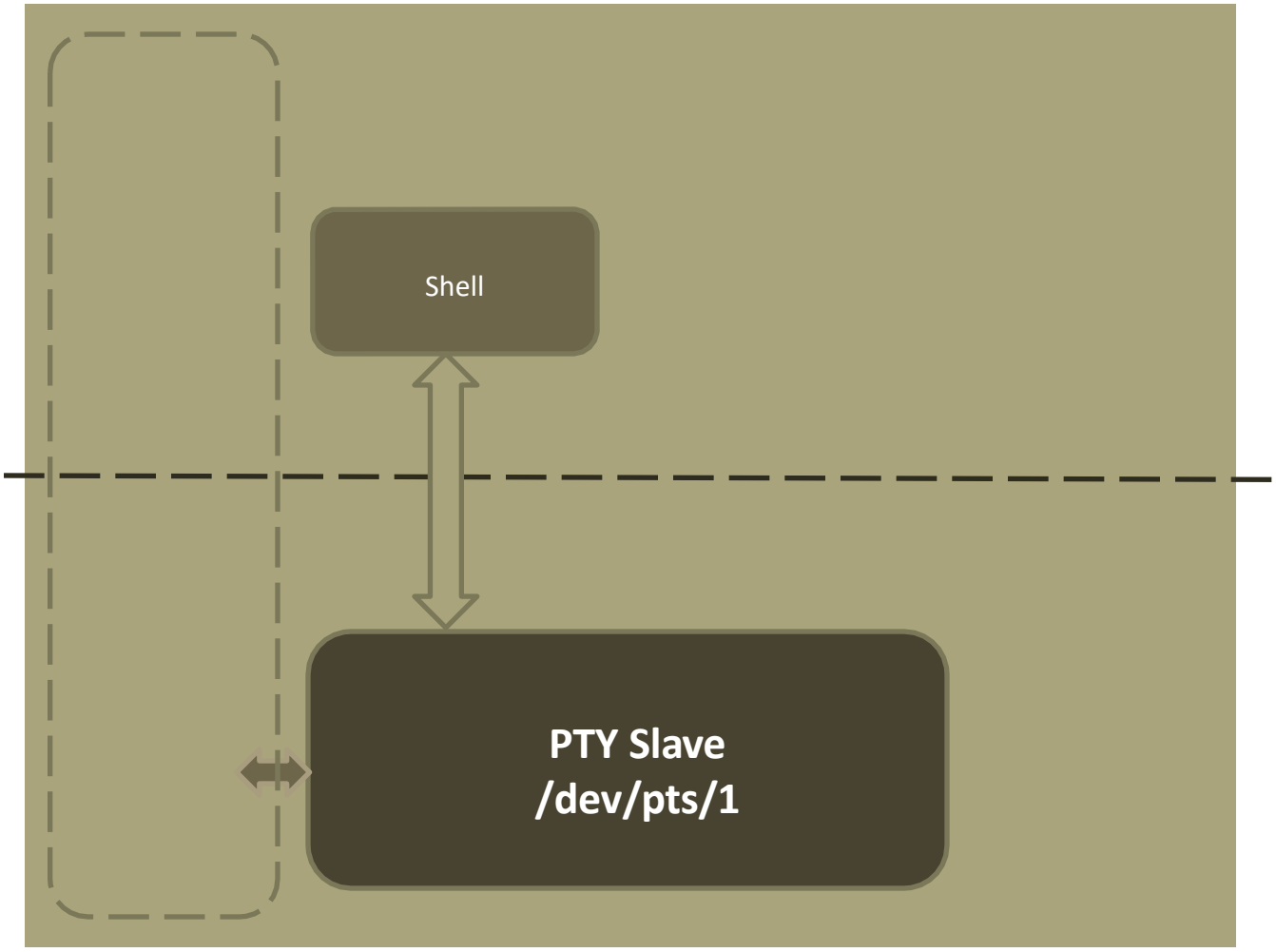
Kabuk Komut Çalıştırma



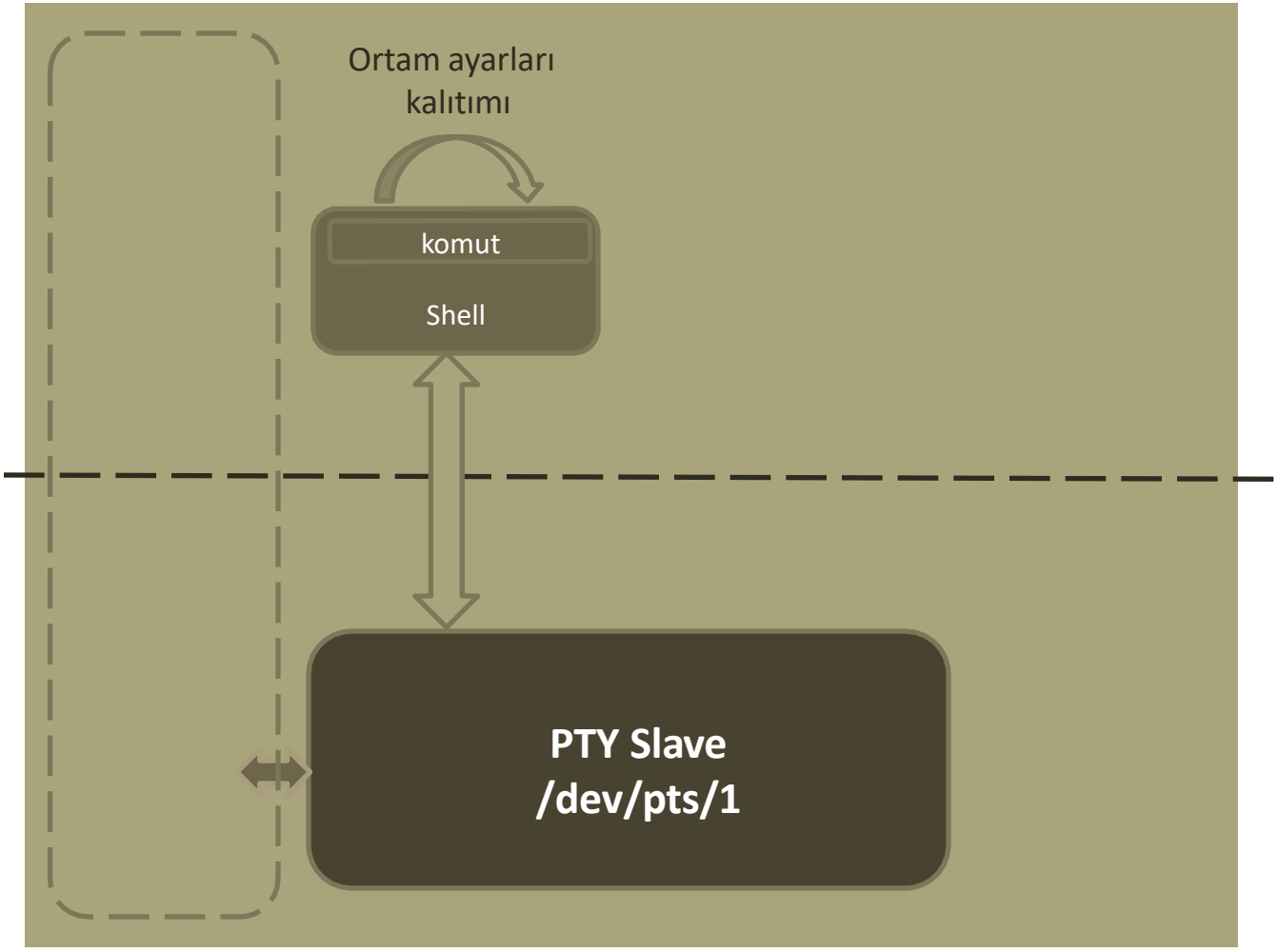
Kabuk Komut Çalıştırma



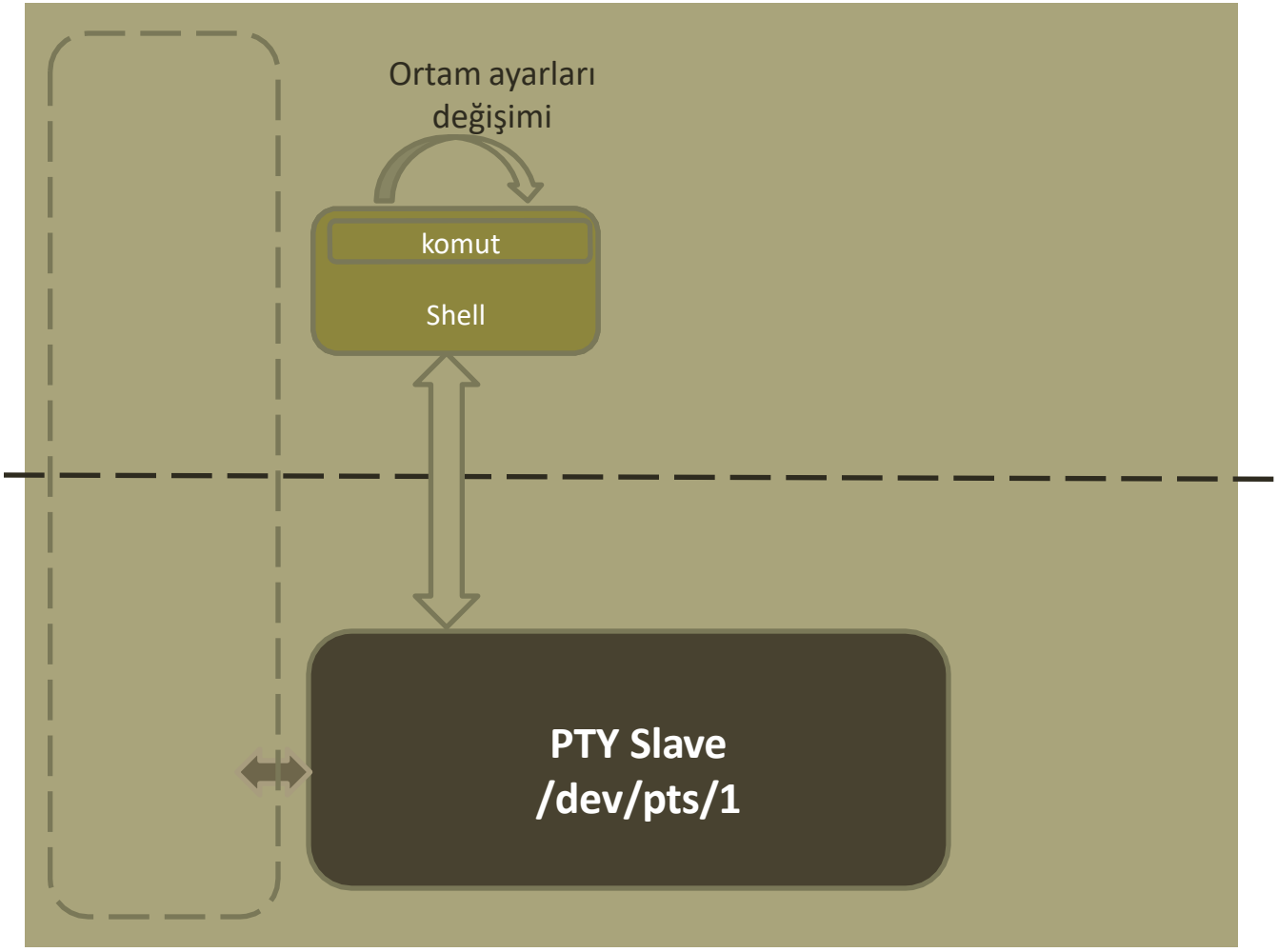
Kabuk Komut Çalıştırma



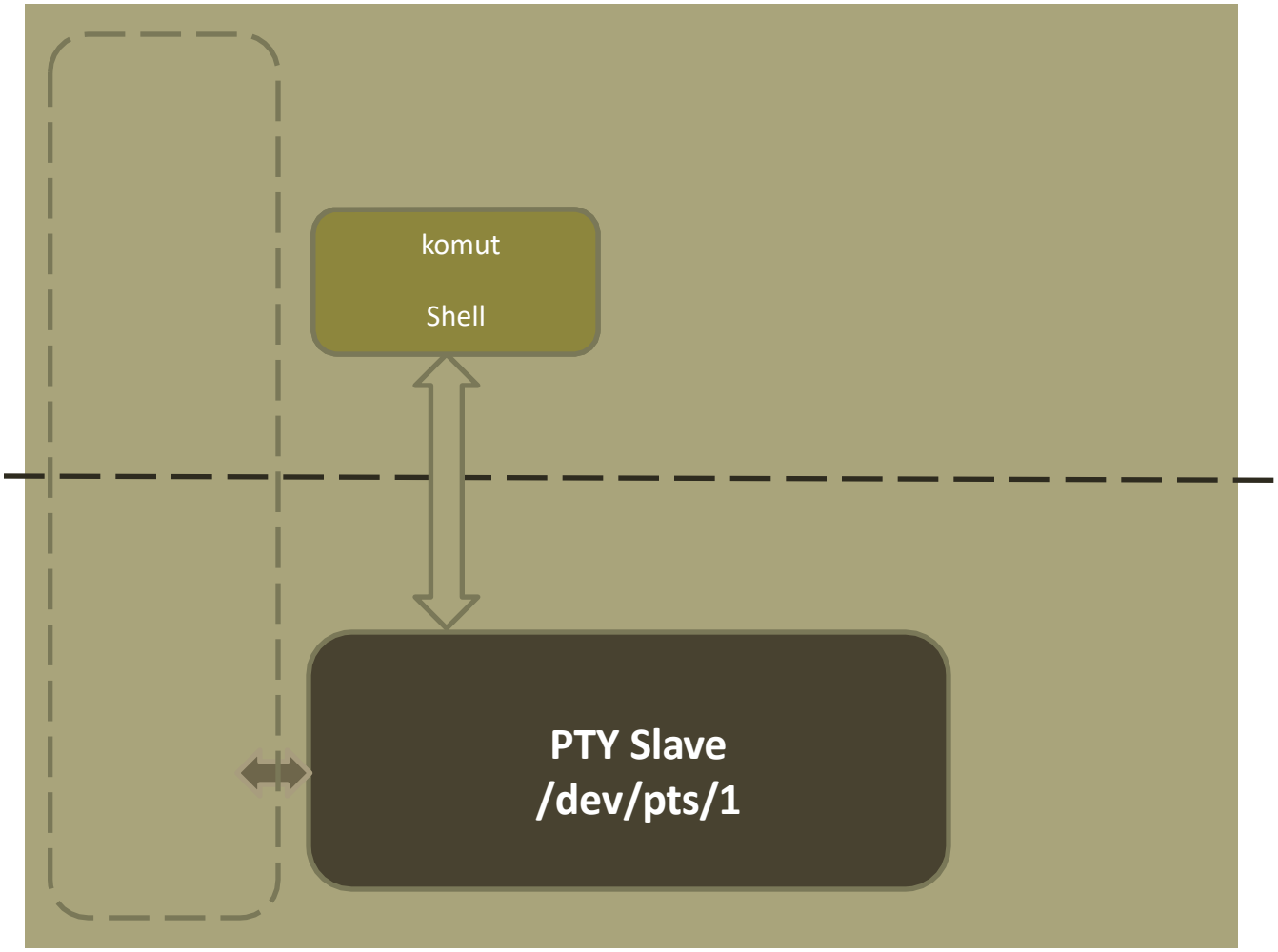
Kabuk Komut Çalıştırma



Kabuk Komut Çalıştırma



Kabuk Komut Çalıştırma



~ / .bashrc'de kullanışlı ayarlar

Ortak Ayarlar

- Aliases ayarları
  - Aliases ayarlamak için iyi bir ***yer ~ / .bashrc*** içindedir.
  - Bu takma adların yalnızca o kullanıcı için geçerli olduğunu unutmayın! Dolayısıyla kök olarak çalıştırırsanız bu takma adları ayarlamazsınız.
- Kullanıcı için Ortam Değişkenlerini Ayarlama
- Kabuk işlevlerini tanımlama

Dosya Üzerine Yazma'dan Koruma (Setting noclobber)

- Bir dosyayı tıkmak, bir dosyanın üzerine yazılması (normalde kasıtsız olarak)
- çıktı yönlendirmesiyle sıklıkla olur

  $ echo "Good Morning" > file.txt

- Bunu önlemek için, ***~/.bashrc*** içindeki noclobber ayarlarını yapalım

  örnek:

  $ set -o noclobber (dosyaların üzerine yazılmasını önleyecektir)

  $ set +o noclobber (koruma kaldırır)

- Not
  - Dosyalar aşırı yazmaya karşı korunursa, üstüne yazmaya zorlayabilirsiniz

    $ echo "Good Morning" >| file.txt

Kaynakça

- Ahmed ElArabawy, Linux for Embedded Systems for Arabs

Teşekkürler.

Dersin Sonu

Kocaeli Üniversitesi Bilgisayar Mühendisliği
Yapay Zeka ve Benzetim Sistemleri Ar-Ge Lab.
http://yapbenzet.kocaeli.edu.tr/