The Paradox of Embedded and Open Source

Tim Bird

Senior Software Engineer, Sony Mobile Communications, Inc
Chair, CE Workgroup Architecture Group
(Founder of ELC)
Keynote

- I haven’t keynoted this event for several years
- I don’t really have an action for you...
ELCeption

- In the movie Inception, an international team implants ideas into their subjects minds, by invading their dreams.
- I want to inject an idea into your mind
  - But, I don’t care if you know I’m trying to do it
  - And I don’t have my dream-invading equipment with me...
Outline

- Open Source and network effects
- Embedded
- Internet of Things
- Fragmentation
Open Source

How it works

Developers publish their derivative software
System only works when people do more than publish
They need to contribute

Definition
Open Source software can be freely used, changed, and shared by anyone

Other Licenses
- BSD
- Apache
- MIT
Different Legal Terms

Network Effects

Creates a community

Community size
- Not a single community
- Lots of sub-communities

Open Source and Network Effects
Other developers write software you use
More developers = more value
Ecosystem
- Build services
- Code training
- Jobs, etc.
Definition

Open Source Software can be freely used, changed and shared by anyone

GPL License

Guarantees freedom
Mainstream users
Source code

Other Licenses

Framework
GPL License

Guarantees freedom for downstream users to obtain source code

Legal framework
Other Licenses

- BSD
- Apache
- MIT

Different Legal Terms
Open Source

How it works

Developers publish their derivative software

System only works when people do more than publish

They need to contribute
Creates a community
Network Effects
Network Effects

Economic term for the effect where the value of something increases based on the number of participants or users that adopt it.

Types

Open Source and Network Effects
- Developer writes software for free
- Free developers increase value

Ownership Structure
- Open Source
- Closed Source

Network Effects are Everywhere
- All large companies understand and try to leverage network effects
- First mover advantage
- Companies spend billions to win a format war

Two-sided markets
- Each side attracts more people

Community size
- Not a single component
- Grows if many components grow

Explanatory Power

- Fanboy behavior
  - Convinces people to use your platform in a social behavioral way

- Subsidies to one side of a market
  - TV - relationship of advertisers and viewers

Format wars
- Subsidies to one side of a market
Economic term for the effect where the value of something increases based on the network of participants (users/developers) that adopt it.
Types

Phone network

Format Wars
• VHS vs. Betamax
• HD DVD vs Blu-ray

Operating Systems
• Windows vs. Mac vs. Linux
• Android vs. iOS
Phone network
Operating Systems

- Windows vs. Mac vs. Linux
- Android vs. IOS
Format Wars

- VHS vs. Betamax
- HD DVD vs Blu-ray
Two-sided markets

In 2014 -- still evolving our theories about multi-sided markets
Network Effects are Everywhere

All large companies understand and try to leverage network effects.

3rd parties increase your value!

First mover advantage
3rd parties increase your value!

First mover advantage

Companies spend billions to win a format war
3rd parties increase your value!

First mover advantage

Companies spend billions to win a format war

Applies to anyone who publishes a platform, where other developers or users create value.
Explanatory Power

- Format wars
- Subsidies to one side of a market

TV - relationship of advertisers and viewers

- Fanboy behavior

  Convincing people to use your platform is a rational behavior -- even if irrational arguments are used
Open Source and Network Effects

Other developers write software you use

More developers = more value

Ecosystem

Related services: Books, training, tools, jobs
Community size

- Not a single community
- Lots of sub-communities

Size matters

Some communities are very small

People are in multiple communities simultaneously
Cost to contribute

To build community, must create generalized software

Generalized software costs more to write

Doesn't perform as well

Bigger
Slower
Embedded

Dedicated function

- Router
- TV
- Digital camera
- Set-top box
- Robots

Mobile phones

Tension between generalization and specialization
**History**

From custom to general-purpose OS

- VxWorks
- Nucleus
- pSOS
- VRTX
- LynxOS
- ITRON
- QNX

- Linux
General Purpose Hardware

Tradeoff in development time vs. hardware resources

Modern SOCs have enormous complexity

Cheapest DRAM is 32M

CPU with 9 cores is same cost as one with 3 cores

Get used to wasting silicon!
Andrew Murray's talk on Boot Time
(2010 ELC Europe)
Device Tree

Not a rant
OK - a little bit of a rant

**DT Problems**
- No typing
- Separation of data from code
- Harder to write drivers
- Unfamiliarity
- Multi-node dependencies not easy-to-express

**DT is Hard to specialize**
- Can't reuse a single driver
- Rooted proven tree at bootstrap
- Can't localize, tree optimizations

**DT and network effects**
- Helps build network effects
- Has no single master creating platforms code for no one

**DT and links between platforms**
- Provides smooth links between platforms
- Does not require a single master creating platforms code for no one
DT is Hard to specialize

It's meant to support single image

Kernel parses tree at runtime

Can't do compile-time optimizations

Relate here my long sad story about Link-Time Optimization and how parsed data items are not optimizable by the compiler.
DT and network effects

DT helps build network effects

Has encouraged restructuring platform code for re-use

Exposes IP blocks between platforms
Exposes IP blocks between platforms

Chipidea, is that you?
Internet of Things

IOT
Changes the equation

We want computers in our:
- cars
- appliances
- furniture
- light switches
- clothing

Possibly in our bodies and our food!

In our infrastructure - monitoring environment, water, energy, traffic

Want to run Linux on a 10-cent processor, that runs for years on a single charge

Finally - Linux on a cereal box

Linux Quote

Linus Quote
IOT
Changes the equation

We want computers in our:
- cars
- appliances
- furniture
- light switches
- clothing

Possibly in our bodies and our food!

In our infrastructure - monitoring environment, water, energy, traffic

Want to run Linux on a 10-cent processor, that runs for years on a single charge
Finally - Linux on a cereal box
Linux in IOT

Why Linux?
Do we actually need Linux here?

Streamlining Linux

Modularity
- Dynamic replacement
- Lite packages
- Easy updates
- Modular design

Subtractive Engineering
- Only what is needed
- No extra bloat
- Efficient resource use

Loose community
- Linuxiverse
- Active and responsive
- Community driven development

No network effects
- Open source
- No proprietary lock-in
- Freedom of choice

Reuse
- Pre-built packages
- Large ecosystem
- Community contributions

SOD support
- Secure, optimized, decentralized
- Customizable security features
- Enhanced privacy and anonymity

Big decisions
- Significant impacts on the future of Linux in IOT
- Key considerations for adoption and implementation

Less is more
- Efficient use of resources
- Minimal overhead
- Scalability and performance

Integrated solutions
- Cross-platform compatibility
- Interoperability with various devices
- Flexibility and adaptability

In our bodies
- Biological integration
- Natural processes
- Enhanced health and well-being

in our heads
- Cognitive improvements
- Enhanced mental processing
- Creative thinking
Why Linux?

Do we actually need Linux here?

Hitachi RFID chip
Problem with Linux in IOT

Linux is too:
- Big
- Slow
- Power-hungry
- Insecure

Linux 0.11 system ran in 2MB

More features since then
Streamlining Linux

Modularity/Legos
Lego Model of Software

Subtractive Engineering
Folly of subtractive engineering
As system scales up, it's harder to remove than build from scratch
Nobody wants to remove stuff they don't understand

Just say 'NO' to subtractive engineering

Lose Community
If we slim down Linux, it's not Linux anymore
No network effects

People want to leverage:
- Network stack
- File systems
- USB
- NFC

SOC support
Modularity/Legos

Lego Model of Software

- WiFi stack needs:
  - ICMP
  - crypto lib
  - O(1) Scheduler
  - ...

Ideal - molded plastic

Franken-car

Folly of sub-subs
Lego Model of Software

Franken-car

Wifi-stack needs:
- ICMP
- crypto lib
- O(1) Scheduler
- ....
Franken-car

Wifi-stack needs:
- ICMP
- crypto lib
- O(1) Scheduler
- ...

Ideal - molded plastic
Subtractive Engineering

Folly of subtractive engineering

As system scales up, it's harder to remove than build from scratch

Nobody wants to remove stuff they don't understand

Just say "NO" to subtractive engineering
Lose community

If we slim down Linux, it's not Linux anymore

No network effects
**Reuse**

What re-use are we striving for?

People want to leverage:
- Network stack
- File systems
- USB
- NFC

SOC support
What to do?

Fork!
Fragmentation

Bad Fragmentation

Good Fragmentation

Linus Quote

What to do?

Fork!
Bad Fragmentation
Linus Quote

Linux Mall - Ask Linus, May 2000
We take your questions and submit the most popular and most interesting questions to Linus. Every week we post Linus' answer to one of your questions.

Got a question for Linus? Ask him in the Ask Linus forum.

This Week's Question

Q: Do you think Linux will avoid following the same "fragmentation" route that killed UNIX? Why and how?

A: It will, and it won't.

Let me explain. What made the UNIX fragmentation so bad was that it was an "overlapped" fragmentation - pretty much every single UNIX vendor went after the same market, which meant that every fragment really wanted to do the same things, but because of the lack of openness everybody ended up spending inordinate amounts of energy to re-invent the wheel that somebody else had already done.

And that fragmentation will not happen under Linux. Simply because people do not have to write their own versions of the same thing. They may end up _improving_ on somebody else's version, or writing a new and improved thing that everybody else can use too, and that's all to the good. But we won't have people fighting on basic simple infrastructure issues, simply because all the real infrastructure is under a license that really forces the different companies to share.

But what will happen is that the "market" fragments, as opposed to the technology. Which is good and proper. You'll have different Linux companies going after different markets, and having different priorities. Which means that you won't have any one particular "microsoft of Linux": you'll have dominant players, but they'll be dominant in the areas that they are good at. And nobody ever ends up being the best at everything.

This week's LinuxMall.com's Ask Linus column is sponsored by Team Linux.
At the intersection of Open Source and Embedded
We need a new base camp here!
ELCeption

- Pay attention to network effects
- Forking can equal growth

Thanks for your time