

An aerial night view of the Dubai skyline, featuring the Burj Khalifa and other skyscrapers. A bright blue beam of light originates from the left side of the frame and points towards the center. The city lights are visible in the background, and the foreground shows a mix of modern and older buildings.

Programmable Network Devices: One Vendor's Perspective

Kenneth Duda
Founder, CTO & SVP Software
Arista Networks, Inc.

ARISTA

Data Path Programmability on Network Switches

- What?
 - P4 for ASICs
 - VHDL for FPGAs
- Who?
 - Us! (The switch vendor)
 - Sophisticated customers
 - Development partners
- Why?
 - That's what this talk is about...

The Promise of ASIC Programmability

- More flexible! *Deploy new features into existing infrastructure!*
- Lower cost! *Reduce cost/power/risk of unneeded features!*
- More efficient! *Reconfigure one hardware SKU for many use cases!*
- Faster development! *Reduce new silicon time to market by decoupling features!*
- More strategic! *Customers gain advantage through custom IP!*

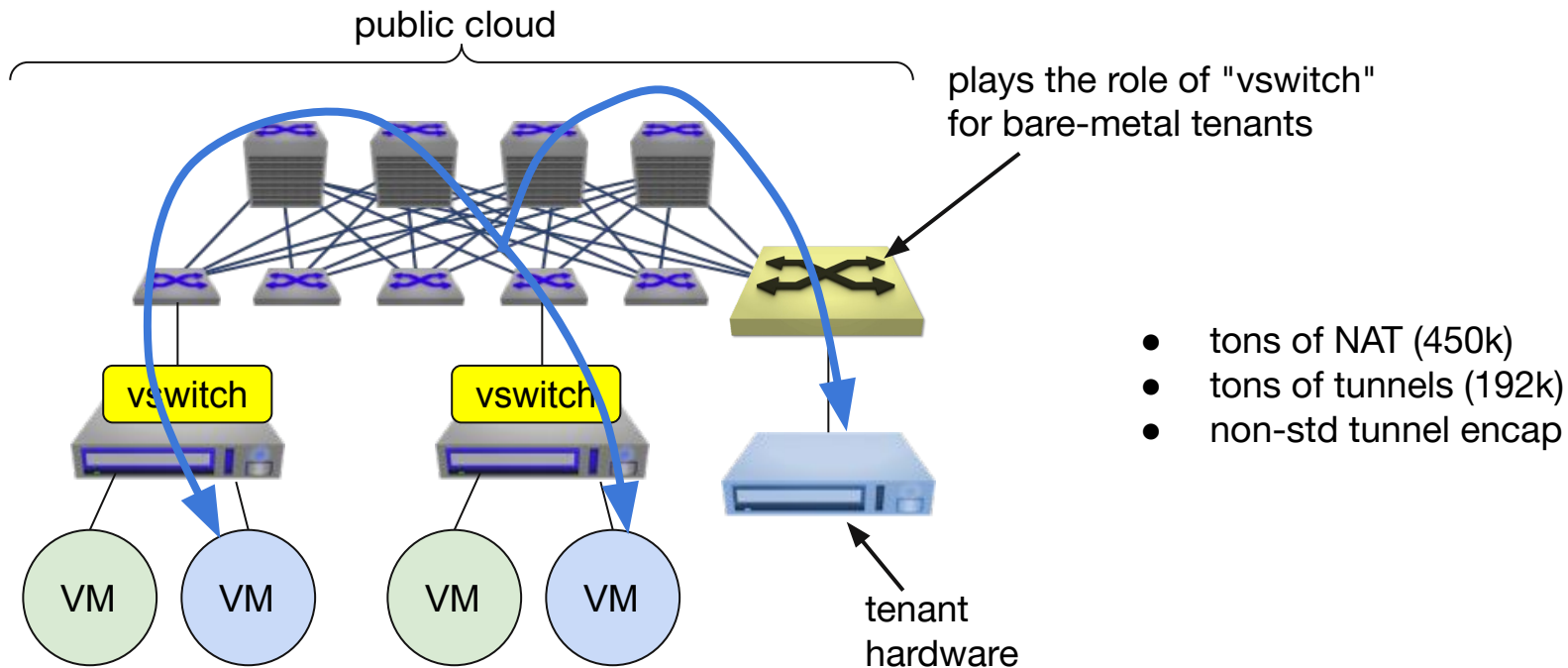
Reality sets in, switch vendor edition

- **More flexible!** Deploy new features into existing infrastructure!
... but it takes years to deploy new network architectures (hardware EOL's)
- **Lower cost!** Reduce cost/power/risk of unneeded features!
... but niche use cases can't drive volume
- **More efficient!** Reconfigure one hardware SKU for many use cases!
... but diverse roles have diverse port configs => SKU diversity either way
- **Faster development!** Reduce new silicon time to market by decoupling features!
... there are other ways to optimize time-to-market for new silicon process
- **More strategic!** Customers gain advantage through custom IP!
... but can they really?

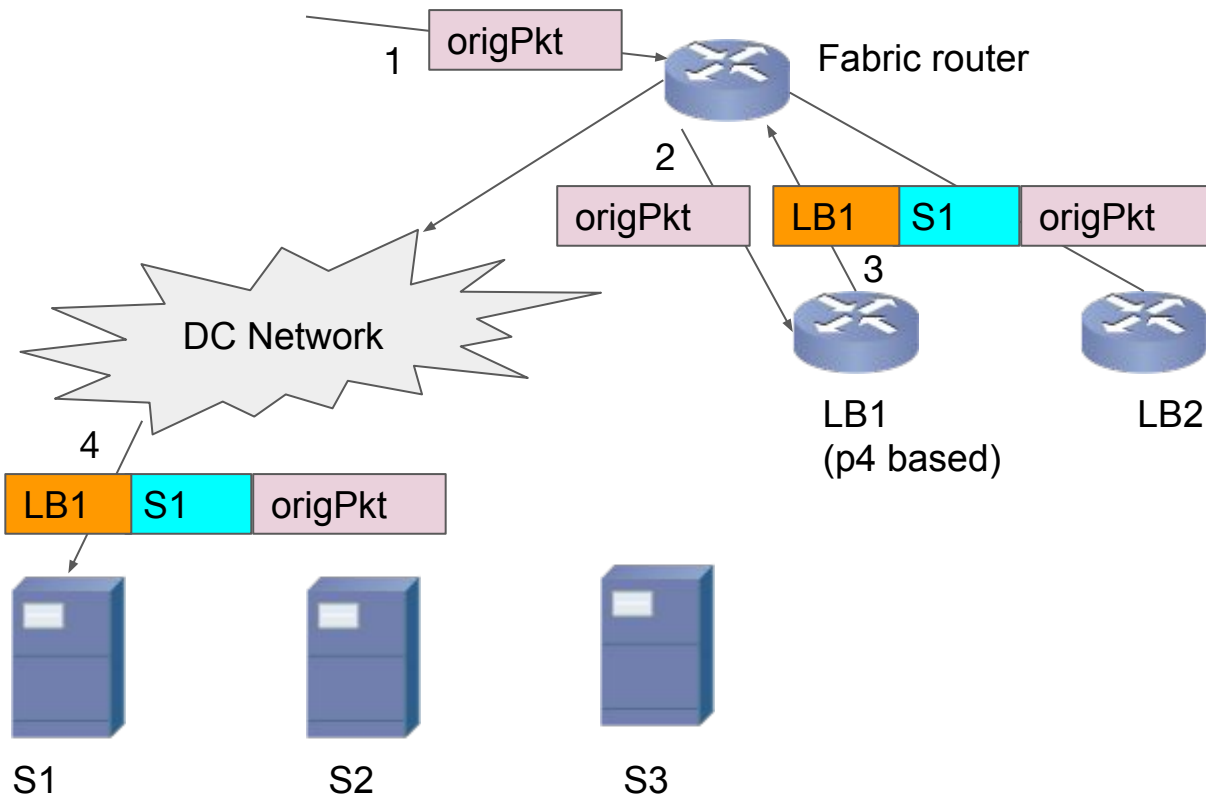
Reality sets in, end user edition

- How do we divide engineering between system vendor and end user?
- Customer wants "standard switch features plus tweaks"
- How can switch vendors enable customers without sharing our p4 source?
- Fitting a p4 program into an actual ASIC is a major challenge
- Compiler targets the whole pipeline; hard to reserve stages for customers
- We feel there's opportunity for improvement here
- "Multics for the Switch ASIC"???

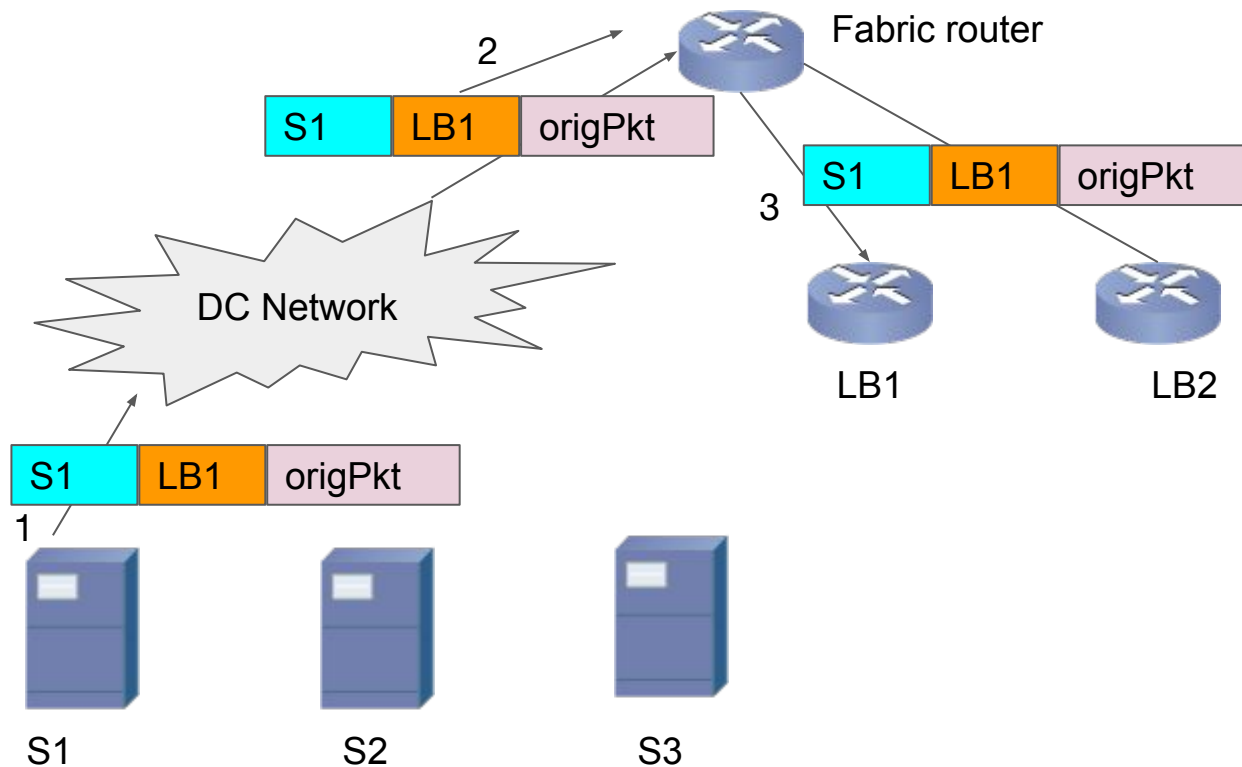
p4 for Bare Metal in the Public Cloud



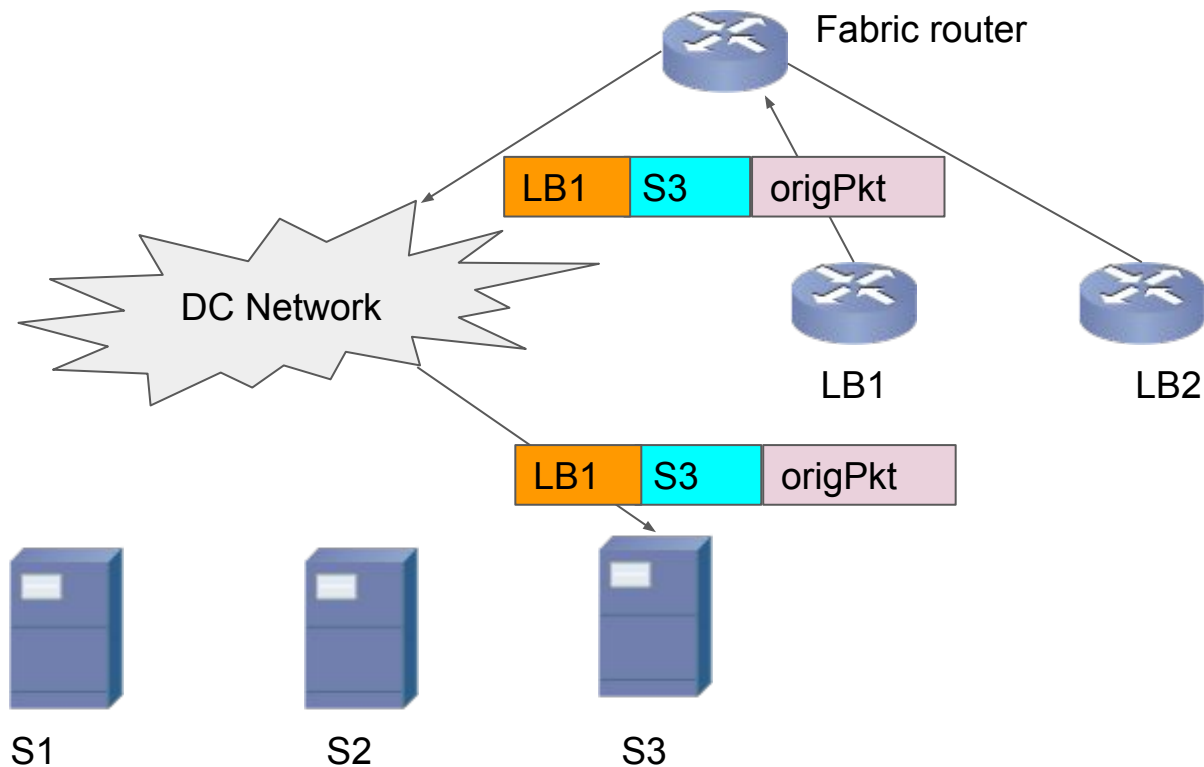
p4 for stateless load balancing



p4 for stateless load balancing



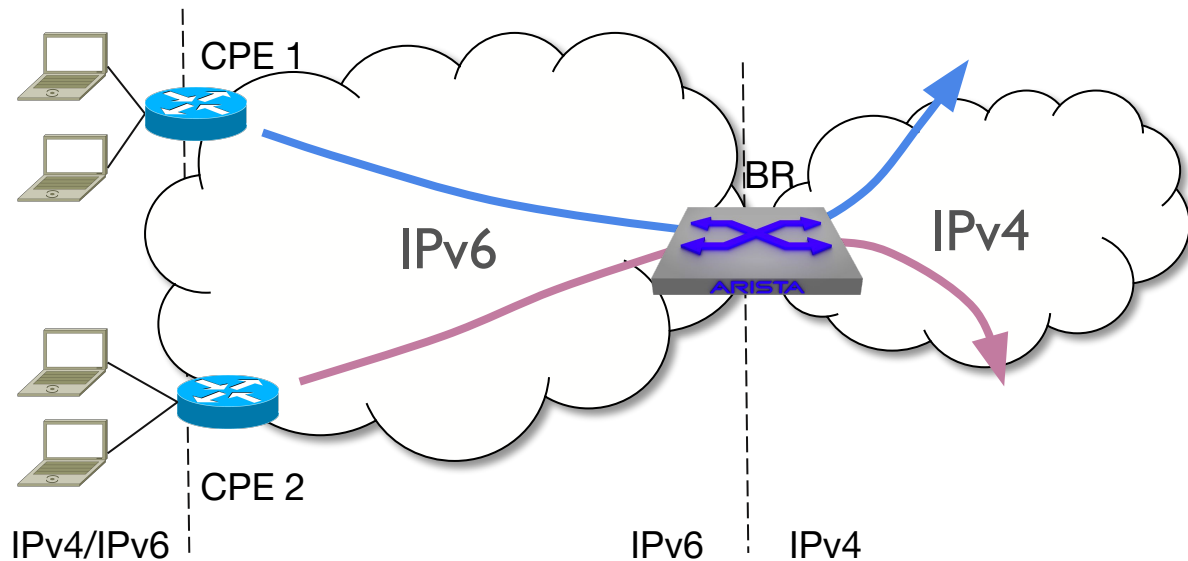
p4 for stateless load balancing



p4 for DDOS Mitigation

- document [here](#)
- cloud titan partner
- replaces A10 + Router
- external system detects ddos attack, informs our box
- 7170 employs two-level policing towards the VIP
 - per-flow
 - aggregate
 - with a permit-list for known-good flows
- 7170 implements TCP SYN challenge
 - simplest: use a bloom filter to drop the first SYN
 - better: syn-cookies in hardware!
- Status: POC complete, customer considering it

p4 for MAP-T

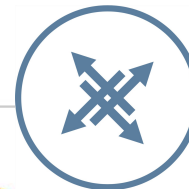
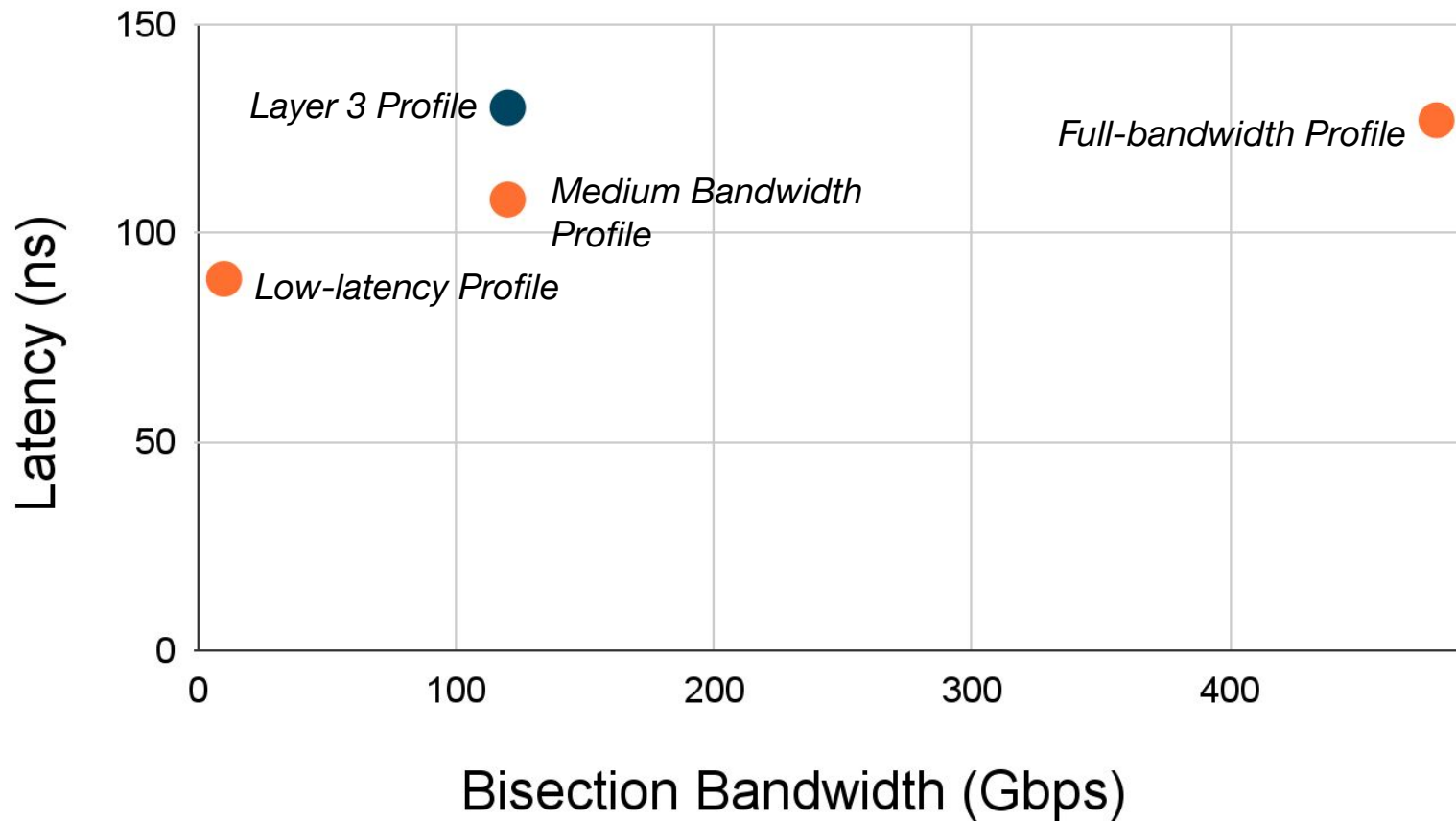


- RFC 7599 — v4-over-v6 via translation (tunneling encoded in DA/SA)
- Need fancy header field extraction, spoof checking, rewrites
- Some MSP's are big fans

FPGA-based Switches

- We provide IP blocks
 - MAC
 - L1 muxing
 - L2 muxing
 - L2/3 switching
- We provide toolchain and SDK
 - CLI library
 - Library for managing state / config
 - Library for talking to FPGA and Switch OS state database
- We provide switch hardware
 - Various FPGAs
 - Various numbers of ports / L1 muxing
 - Some include an L2/3 ASIC too (Jericho2C)

Arista SwitchApp Profiles

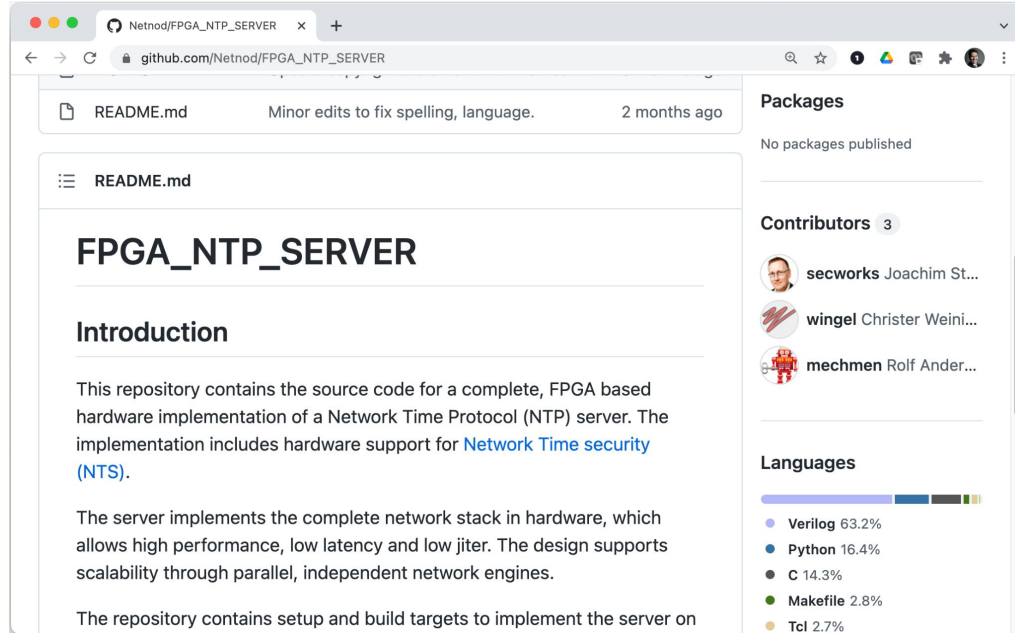


Customers Have Done Stuff

- introduce their own risk management controls
- deploy their own trading algorithm
- we built a multi-FPGA box for more elaborate trading strategies
- there are apparently federal applications as well

Example Customer Use Case: Netnod

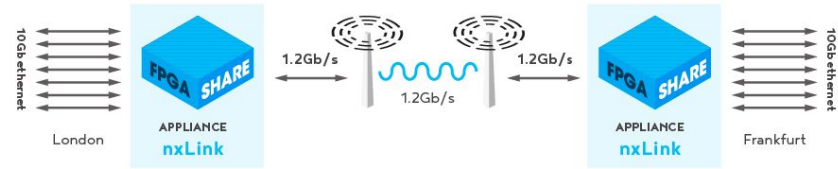
- FPGA-based, secure NTP server, Swedish timing service
- Open-source
- We sell devices to Netnod, and they run their application.
- They use our timing IP.



The screenshot shows a GitHub repository page for 'FPGA_NTP_SERVER'. The main content area displays the repository name and an introduction. The introduction text reads: 'This repository contains the source code for a complete, FPGA based hardware implementation of a Network Time Protocol (NTP) server. The implementation includes hardware support for Network Time security (NTS). The server implements the complete network stack in hardware, which allows high performance, low latency and low jitter. The design supports scalability through parallel, independent network engines. The repository contains setup and build targets to implement the server on'. On the right side, there are sections for 'Packages' (No packages published), 'Contributors' (3 contributors: secworks Joachim St..., wingel Christer Weini..., mechmen Rolf Ander...), and 'Languages' (Verilog 63.2%, Python 16.4%, C 14.3%, Makefile 2.8%, Tcl 2.7%).

So Have Third Party Engineering Partners

- Enyx nxLink: Microwave link management
 - muxing with a backup fiber
 - finance-specific compression
 - space for custom processing logic



- Nextera: Software Defined Digital Video
 - transcoding
 - picture-in-picture
 - scaling
 - compression
 - color correction
 - SRT-to-SMPTE-2110 gateway
 - integrated with L2/L3 switching/routing



Takeaways

- Innovating in the core L2/3 data path is hard and slow going
- Sometimes you can get a jump on things (e.g., MAP-T) but that's rare
- Programmable silicon more useful for L4-7 / niche use cases
- We lack a model for joint development that preserves L2/3 features
- Success with 3rd parties programming FPGAs suggests the market size is greater than zero!

Thank You

ARISTA