



# Exposing Data Plane Programmability on Turn-Key Network Devices

Opportunities, challenges, and options

Mario Baldi

# Programmable Switch Deployment Flavors



Whitebox



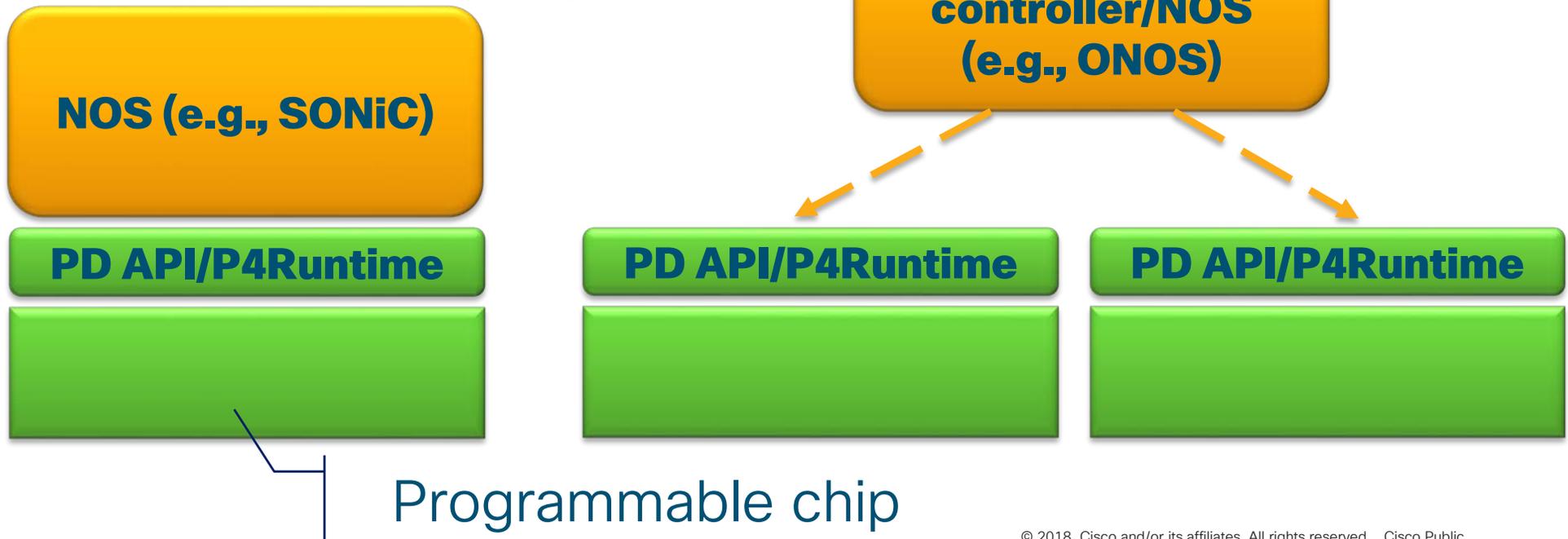
Turn-key



Hybrid

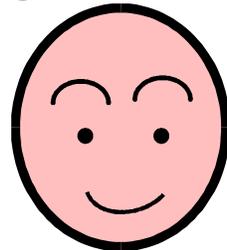
# Whitebox Deployment

- Maximum flexibility 
- Maximum disruption/risk 
- Significant barrier
  - Who can code in P4 today?

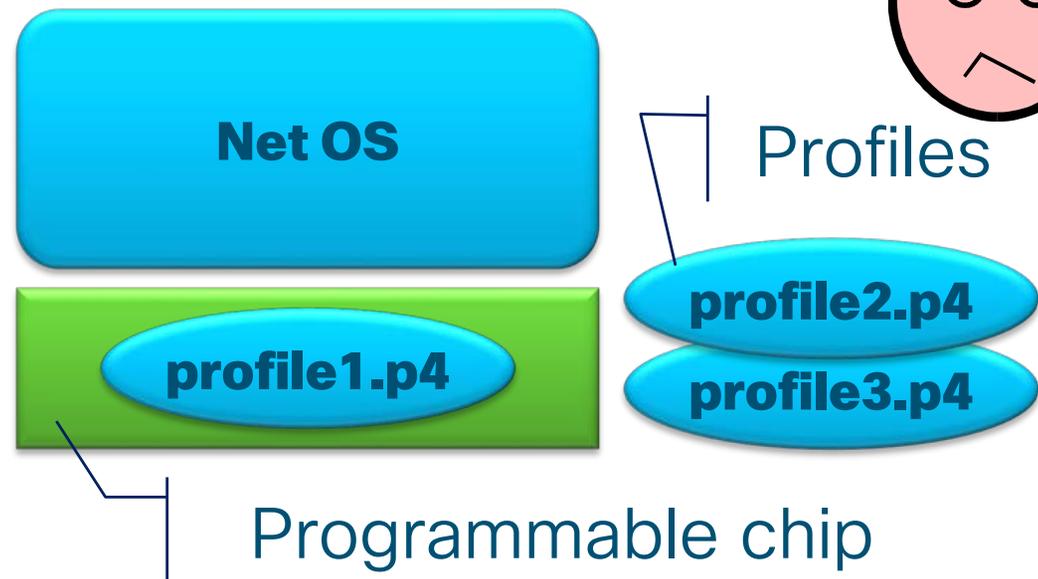
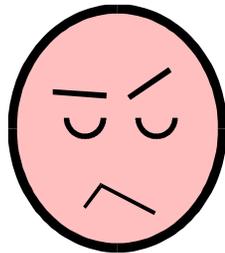


# Turn-key Deployment

- Deployment as usual
  - Familiar features and interfaces
- Resource optimization
- Future proof
- Feature agility
- Streaming telemetry

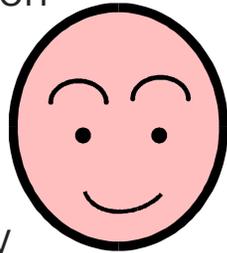


- No flexibility
  - No custom features and protocol support



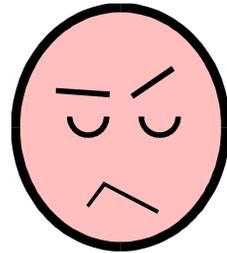
# Open Platform

- Deployment as usual
  - Familiar features and interfaces
- Resource optimization
- Future proof
- Feature agility
- Streaming telemetry



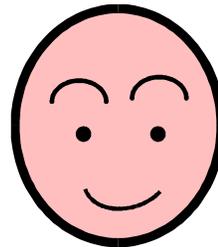
Same as  
Turn-key

- No flexibility
  - No custom features and protocol support

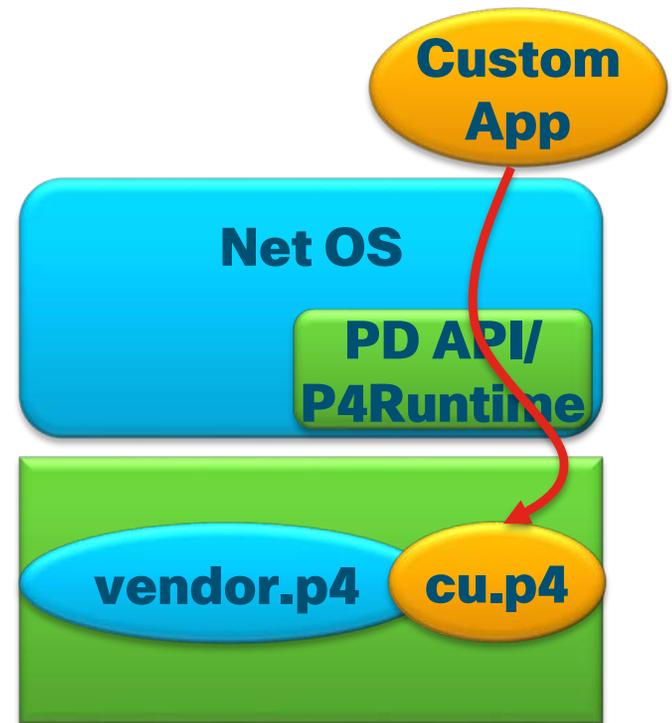


# Hybrid Deployment

- Best of breed
- Deployment as usual
  - Familiar features and interfaces
- And also flexibility



**Without the  
added risk!**



# Hybrid Deployment Challenges

## Do not break what works

- Vendor data plane code is well tested
- ... and we don't want to need regression testing

## Don't want to show, don't want to see

- Vendor code and custom code may be confidential
- Not practical to familiarize with a lot of vendor code to just write a few lines

## Resource availability

- Still "limited" on current chips

## Data/control plane dependence

- Net OS should keep working
- Net OS should not be aware of custom data plane functions

In a nutshell

**P4 and its ecosystem were not  
designed for  
*incremental programming***



Single  
programmer

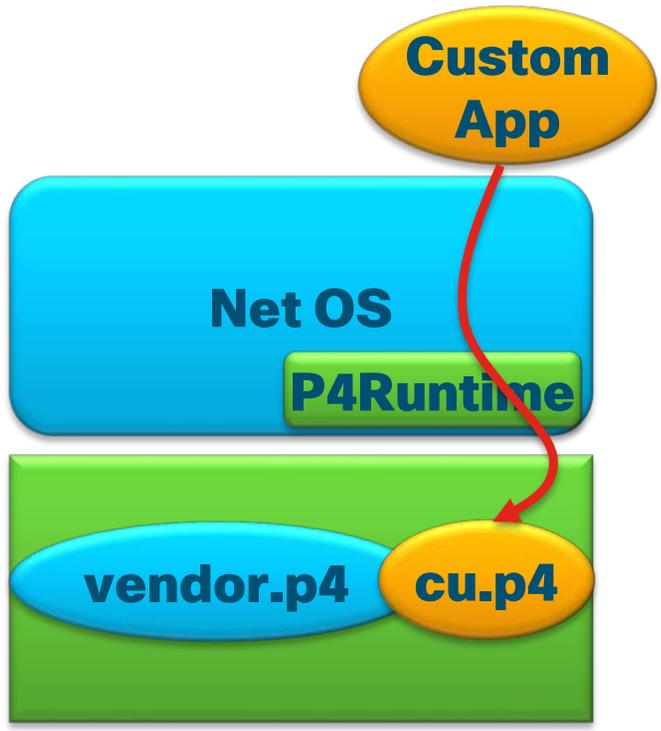
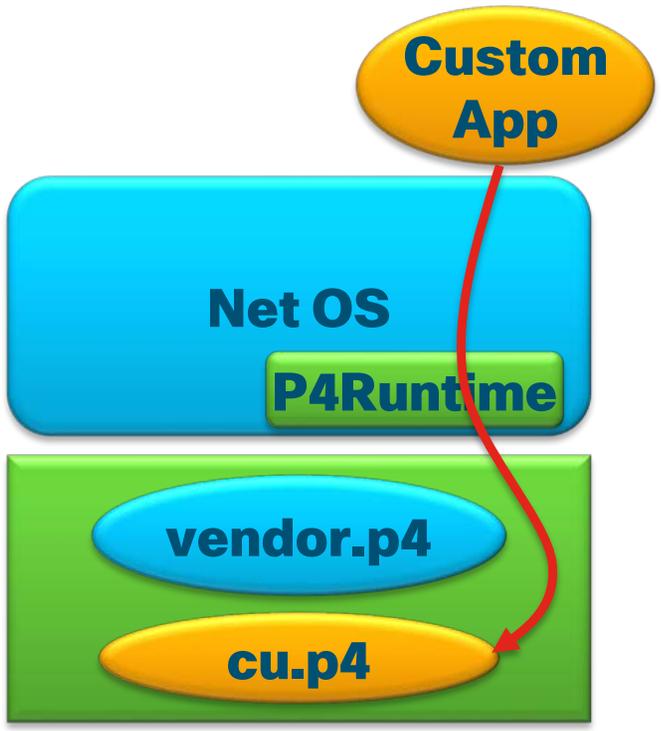


Single source  
code

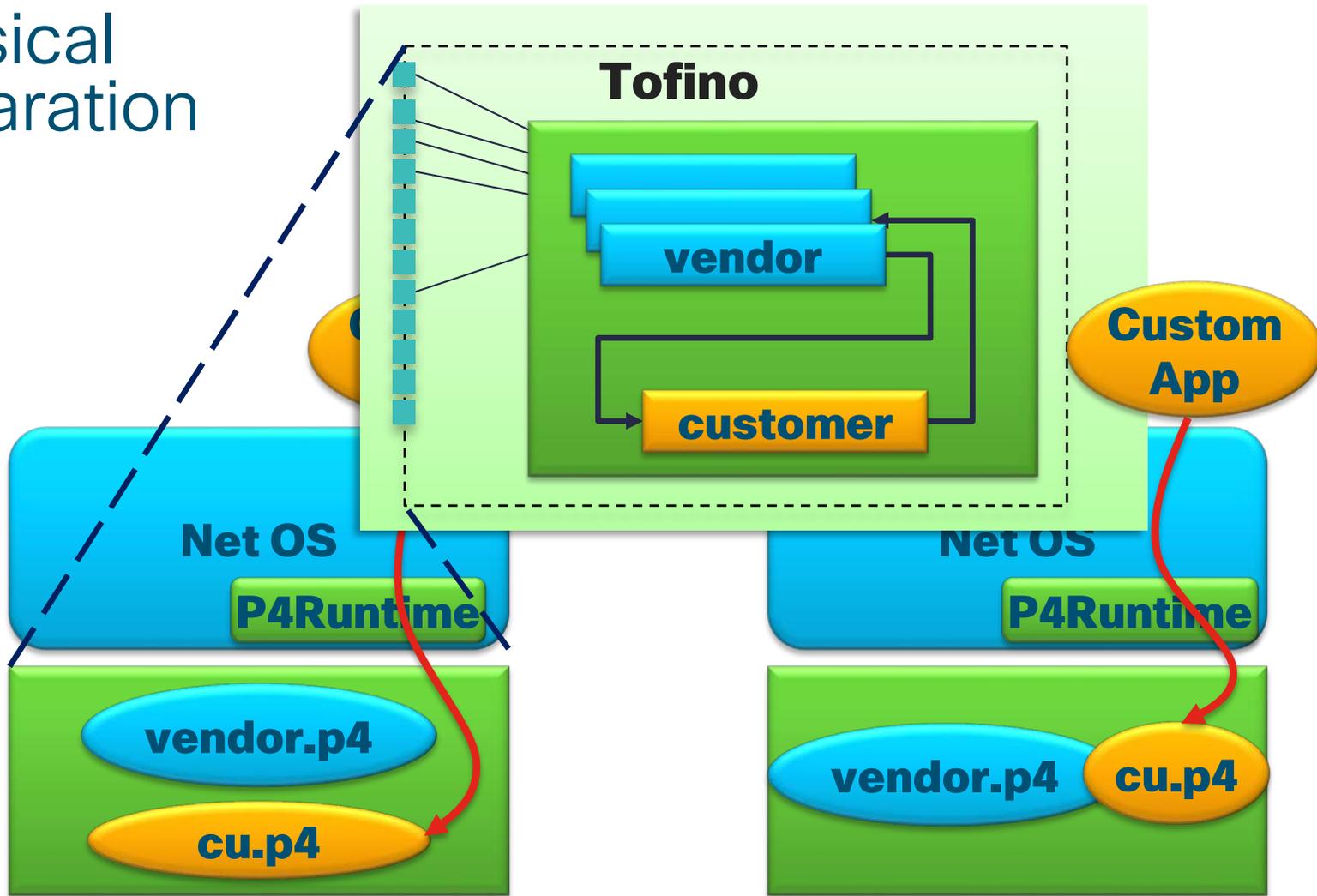


Single control  
plane

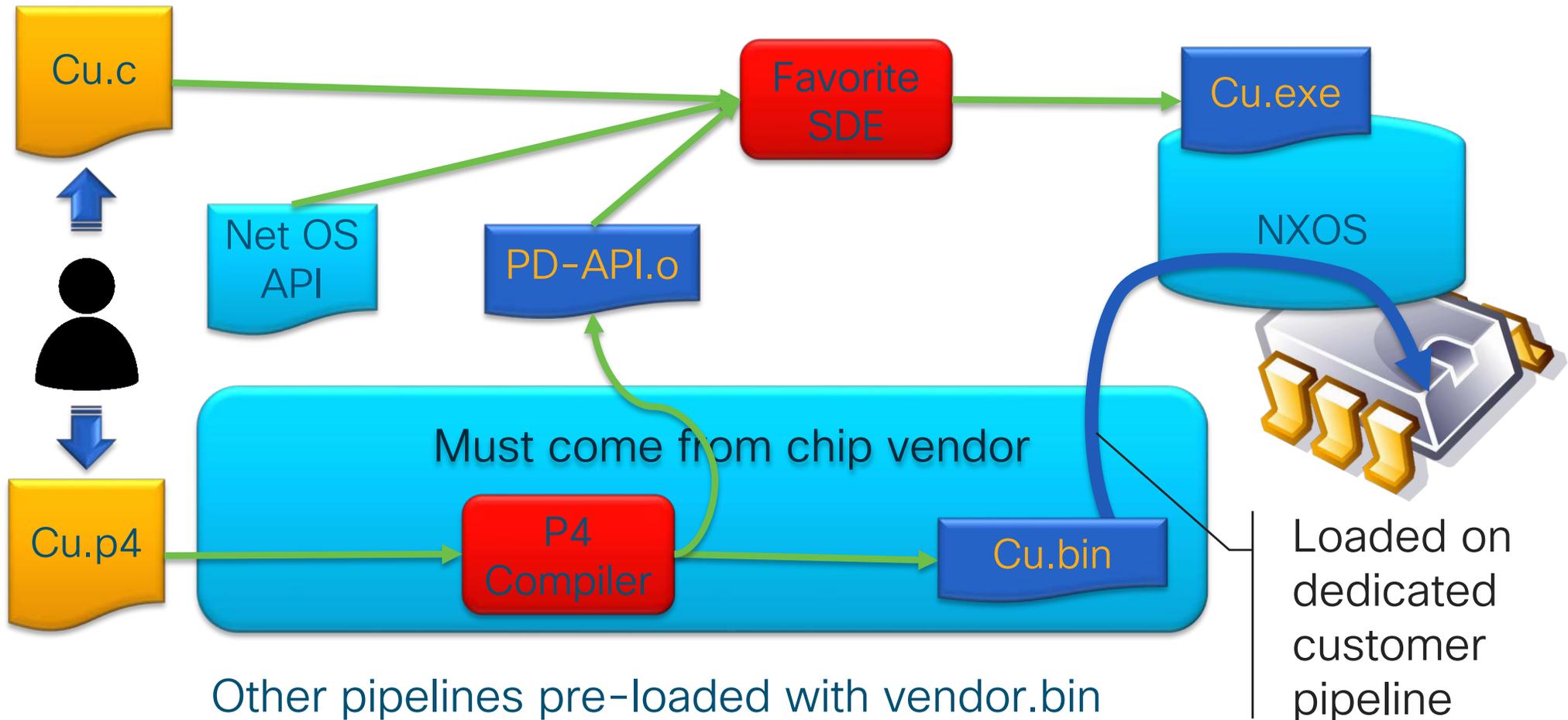
# Possible Options



Physical Separation

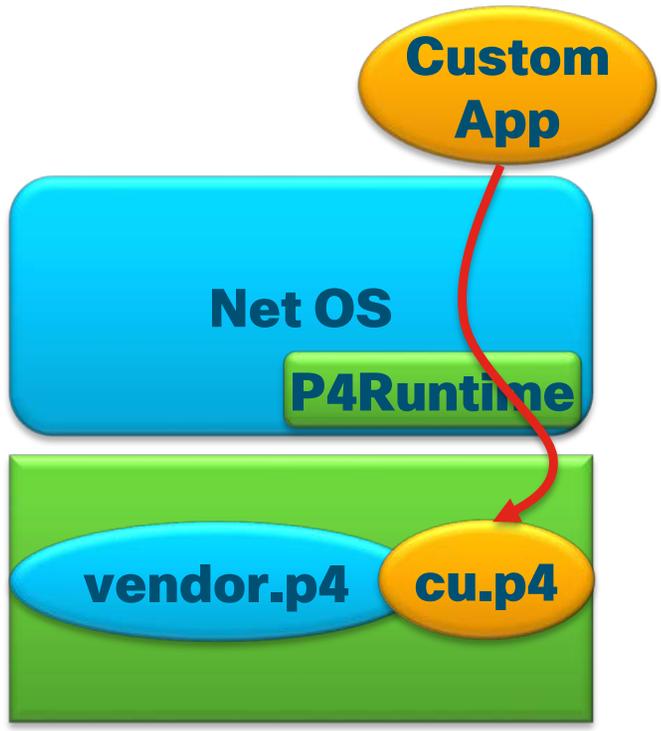
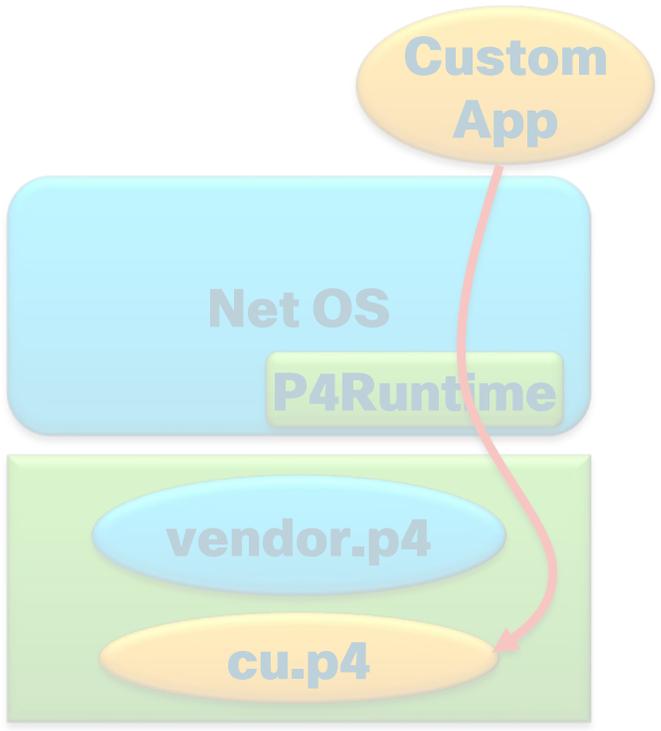


# Incremental Programming Workflow

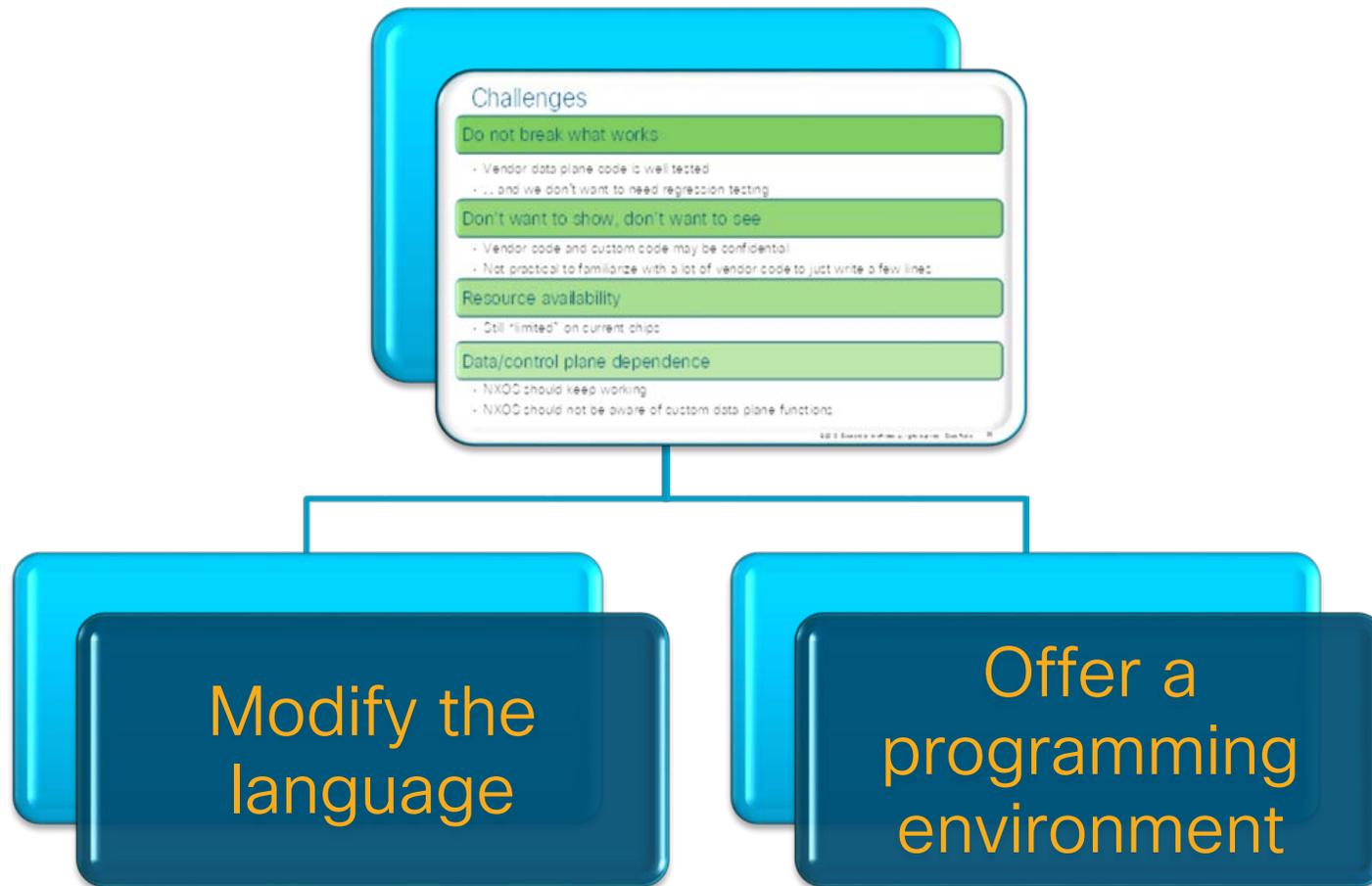


# Software Solution

- Platform vendor (Cisco)
- Chip vendor (Barefoot)
- Customer/open source



# What about the challenges we mentioned earlier?



# Language Design Working Group

## Modularity can help with incremental programming

Sub-working group to introduce modularity in P4

- March 2018

Started focusing on polymorphism

- Generic data type
- Generic function type

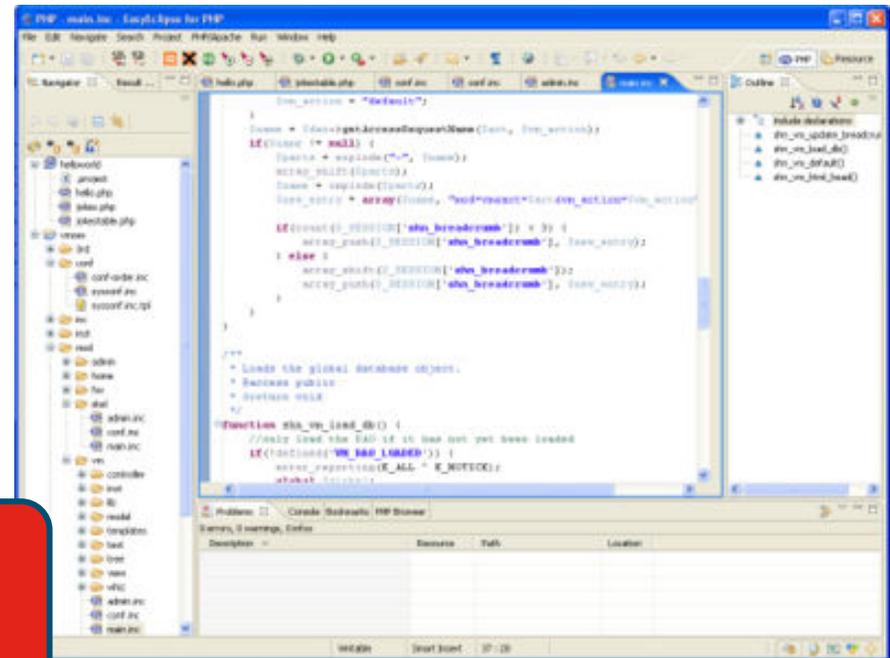
Intent to focus on modularity for incremental programming

# daPIPE

## Data Plane Incremental Programming Environment

Identify constraints  
on new code

Impose those  
constraints on  
the program



The screenshot shows the daPIPE development environment. The main window displays a C code file named 'main.c'. The code includes a function 'main()' that calls 'get\_args\_and\_reqs()' and 'get\_args\_and\_reqs()' to process arguments and requests. It also includes a function 'whv\_hw\_load()' that loads the global hardware object. The code is written in C and uses standard C syntax for loops, conditionals, and function calls. The environment includes a file explorer on the left showing a project structure with folders like 'lib', 'src', and 'include'. A 'Problem' window at the bottom shows a warning about a missing header file.

```
int main() {
    int argc = 0;
    char **argv = NULL;
    int ret = 0;

    if (argc < 2) {
        printf("Usage: %s <command> <args>\n", argv[0]);
        return -1;
    }

    if (strcmp(argv[1], "help") == 0) {
        printf("Usage: %s <command> <args>\n", argv[0]);
        return -1;
    }

    if (strcmp(argv[1], "load") == 0) {
        ret = whv_hw_load();
        if (ret != 0) {
            printf("Failed to load hardware object\n");
            return -1;
        }
    }

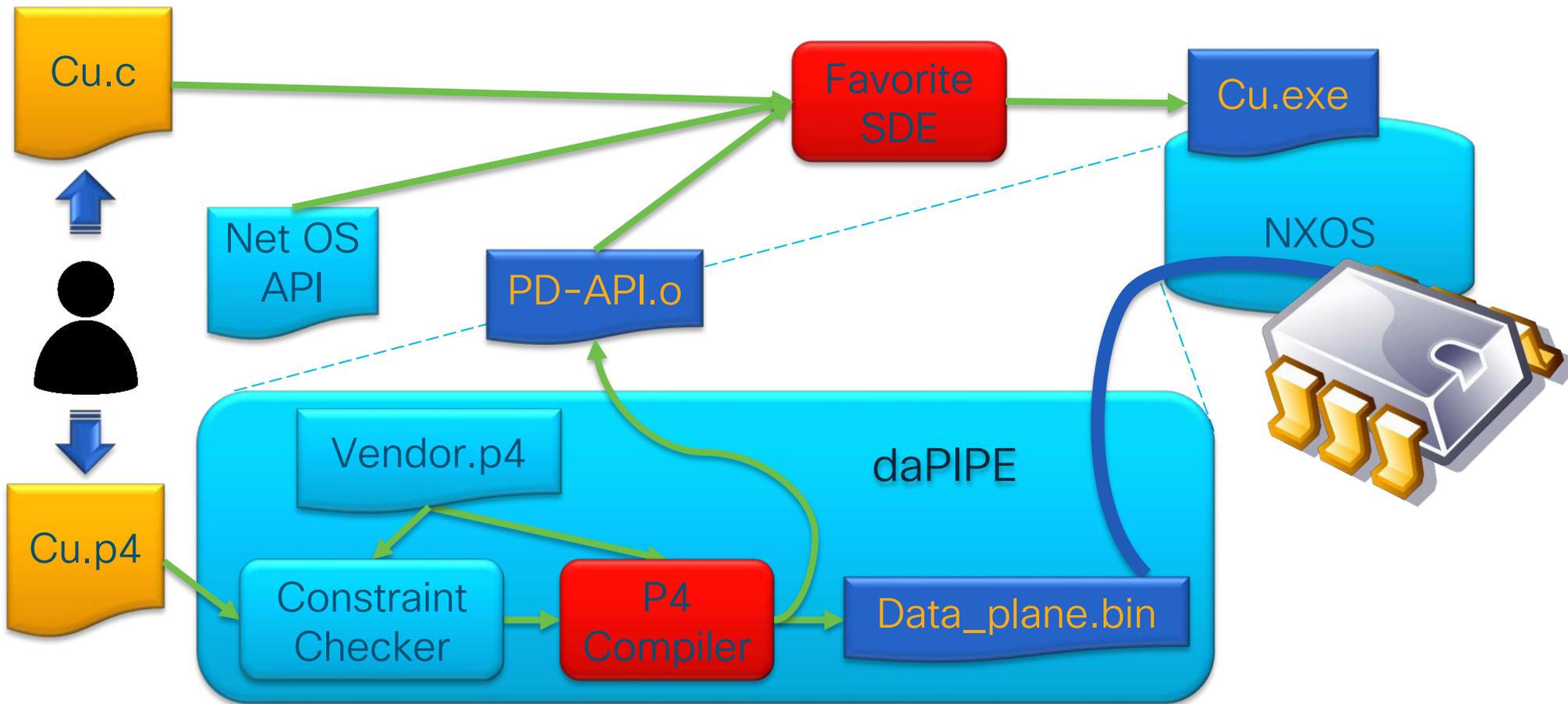
    if (strcmp(argv[1], "run") == 0) {
        ret = get_args_and_reqs(argv[2], argv);
        if (ret != 0) {
            printf("Failed to get arguments and requests\n");
            return -1;
        }
    }

    return ret;
}

/**
 * Loads the global hardware object.
 * Success: 0
 * Failure: -1
 */
int whv_hw_load() {
    // Only load the HW if it has not yet been loaded
    if (!g_hw_loaded) {
        return whv_hw_load_impl();
    }
    return 0;
}
```

*Support* developers and  
*streamline* their task (while  
enforcing needed constraints)

# Customer Programming Workflow





Thank you

Any questions?