pcube: Primitives for network data Plane Programming

Rinku Shah

Aniket Shirke

Akash Trehan

Mythili Vutukuru

Purushottam Kulkarni

Department of Computer Science & Engineering

Indian Institute of Technology Bombay

P4EU 2018

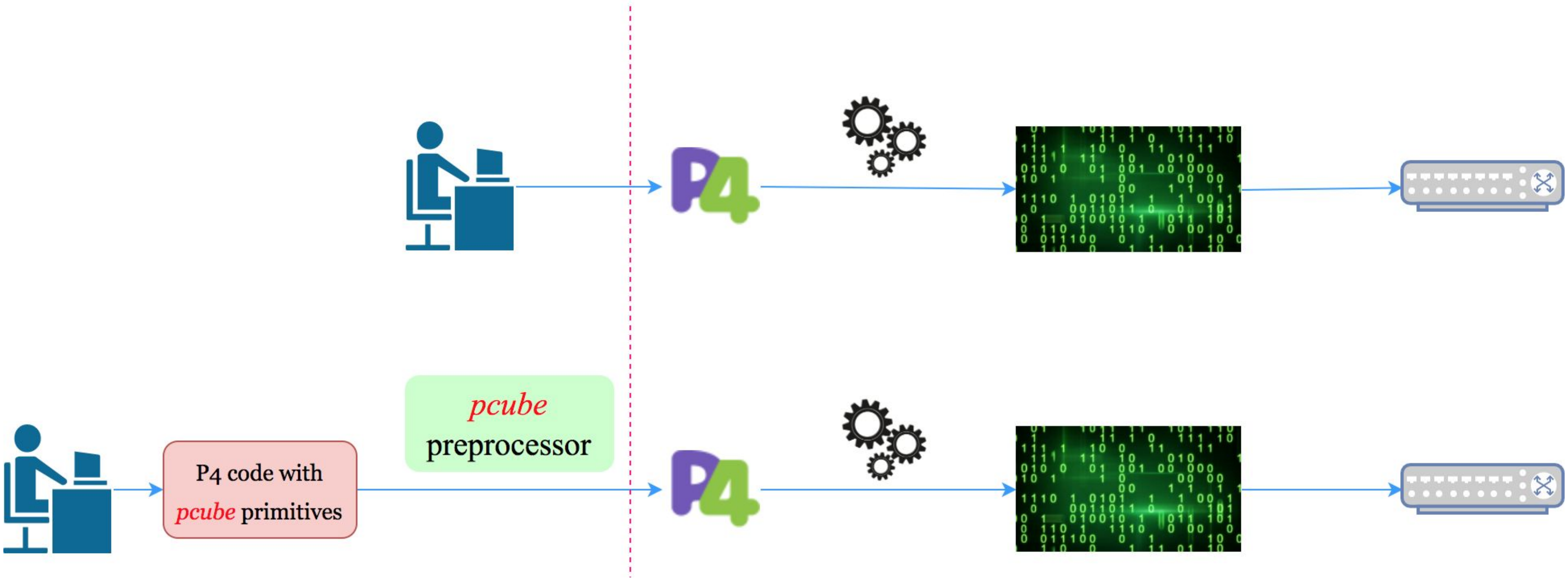
24th September, 2018

P4: The Story So Far

- Revolutionary
- Freedom to have all kinds of features
- Unconstrained by a fixed hardware

Image Source: https://stratumproject.org/wp-content/uploads/2018/03/p4-logo.png

pcube



P4 code with *pcube* primitives

pcube preprocessor

Image Source: https://stratumproject.org/wp-content/uploads/2018/03/p4-logo.png

Pcube primitives

code, code, code!

```
table get_server_flow_count_table{
    actions{
        get_server_flow_count;
    }
}

table update_min_flow_len1_table1 {
    actions{
        update_min_flow_len1;
    }
    size: 1;
}

table update_min_flow_len2_table1 {
    actions{
        update_min_flow_len2;
    }
    size: 1;
}

table set_probe_bool_table {
    actions{
        set_probe_bool;
    }
    size: 1;
}

table set_switch1_dest_port_table{
    actions{
        set_switch1_dest_port;
    }
    size:1;
}

table set_switch2_dest_port_table{
    actions{
        set_switch2_dest_port;
    }
    size:1;
}

table set_switch3_dest_port_table{
    actions{
        set_switch3_dest_port;
    }
    size:1;
}

table update_map_table {
    actions {
        update_map;
    }
}
```

```
control ingress {
    if (load_balancer_head.preamble == 1){
        apply(mirror_info1_table);
    }
    else {
    if(load_balancer_head.syn == 1) {
    if(meta.server_flow1 < 5 or meta.server_flow2 < 5){
    apply(set_server_dest_port_table);
    if ((meta.server_flow1 + meta.server_flow2)*100 > (meta.upper_limit * 2 * 5)){
            apply(set_probe_bool_table);
        }
    }
    else{
        apply(get_switch_flow_count_table);
    if (meta.flow1 >= 2*5 and meta.flow2 >= 2*5 and meta.flow3 >= 2*5){
        apply(drop_table);
    }
    else {
        if(meta.flow1 <= meta.flow2 and meta.flow1 <= meta.flow3) {
            apply(set_switch1_dest_port_table);
        }
        else if(meta.flow2 <= meta.flow1 and meta.flow2 <= meta.flow3) {
            apply(set_switch2_dest_port_table);
        }
        else if(meta.flow3 <= meta.flow1 and meta.flow3 <= meta.flow2) {
            apply(set_switch3_dest_port_table);
        }
    }
    apply(update_map_table);

    if(load_balancer_head.fin == 1) {
        apply(clear_map_table);
        apply(update_flow_count_table);
    if ((meta.server_flow1 + meta.server_flow2)*100 < (meta.lower_limit * 2 * 5)){
        if(meta.routing_port == 2 or meta.routing_port == 3){
        apply(sync_info2_table);
        }
    }
    }
    }
}
```

```
action get_limits(upper_limit, lower_limit){
    modify_field(meta.upper_limit, upper_limit);
    modify_field(meta.lower_limit, lower_limit);
}

action get_server_flow_count(){
        register_read(meta.server_flow1,
total_flow_count_register, 1 - 1);
        register_read(meta.server_flow2,
total_flow_count_register, 2 - 1);
}

action update_switch_flow_count() {
    register_write(total_flow_count_register,
standard_metadata.ingress_port - 2, sync_info._0);
}
```

```
action set_server_dest_port(flow_count,flow_dest){
    register_write(reg, flow_dest - 2, flow_count + 1);
    modify_field(standard_metadata.egress_spec, flow_dest);
}
action set_probe_bool(){
    modify_field(meta.probe_bool, 1);
}
action get_switch_flow_count(){
        register_read(meta.flow1, reg, 1 + 3 - 2);
        register_read(meta.flow2, reg, 2 + 3 - 2);
        register_read(meta.flow3, reg, 3 + 3 - 2);
}
```

The pcube_for Primitive

```
action write_reg1(){
    register_write(reg,1,meta.f1 + 1);
}
action write_reg2(){
    register_write(reg,2,meta.f2 + 1);
}
action write_reg3(){
    register_write(reg,3, meta.f3 + 1);
}
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_for Primitive

Before pcube

```
action write_reg1(){
    register_write(reg,1,meta.f1 + 1);
}
action write_reg2(){
    register_write(reg,2,meta.f2 + 1);
}
action write_reg3(){
    register_write(reg,3, meta.f3 + 1);
}
```

After pcube

```
@pcube_for (i) (1,4,1)
    action set_port$i(){
        register_write(reg,$i,meta.f$i + 1);
    }
@pcube_endfor
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_for Primitive

Before pcube

```
action write_reg1(){
    register_write(reg,1,meta.f1 + 1);
}
action write_reg2(){
    register_write(reg,2,meta.f2 + 1);
}
action write_reg3(){
    register_write(reg,3, meta.f3 + 1);
}
```

After pcube

Loop variable

```
@pcube_for (i) (1,4,1)
    action set_port$i(){
        register_write(reg,$i,meta.f$i + 1);
    }
@pcube_endfor
```

Accessing loop variable

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_for Primitive

Before pcube

```
action write_reg1(){
    register_write(reg,1,meta.f1 + 1);
}
action write_reg2(){
    register_write(reg,2,meta.f2 + 1);
}
action write_reg3(){
    register_write(reg,3, meta.f3 + 1);
}
```

After pcube

Start Index

End Index

Step size

```
@pcube_for (i) (1,4,1)
    action set_port$i(){
        register_write(reg,$i,meta.f$i + 1);
    }
@pcube_endfor
```

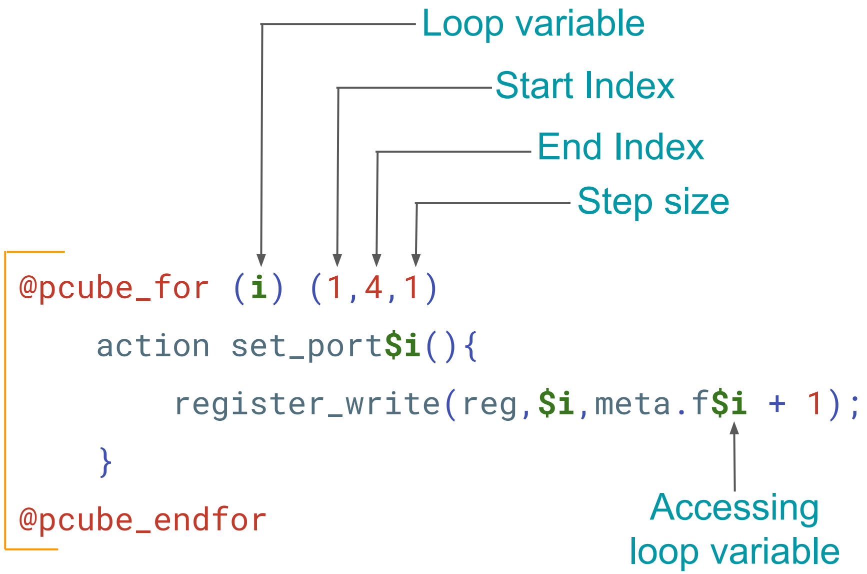
Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_for Primitive

Before pcube

```
action write_reg1(){
    register_write(reg,1,meta.f1 + 1);
}
action write_reg2(){
    register_write(reg,2,meta.f2 + 1);
}
action write_reg3(){
    register_write(reg,3, meta.f3 + 1);
}
```

After pcube



```
@pcube_for (i) (1,4,1)
    action set_port$i(){
        register_write(reg,$i,meta.f$i + 1);
    }
@pcube_endfor
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_minmax primitive

Before pcube

```
if(var1 <= var2 and var1 <= var3){
    apply(tab_server1);
}
else if(var2 <= var1 and var2 <= var3){
    apply(tab_server2);
}
else if(var3 <= var1 and var3 <= var2){
    apply(tab_server3);
}
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_minmax primitive

Before pcube

```
if(var1 <= var2 and var1 <= var3){
    apply(tab_server1);
}
else if(var2 <= var1 and var2 <= var3){
    apply(tab_server2);
}
else if(var3 <= var1 and var3 <= var2){
    apply(tab_server3);
}
```

After pcube

```
@pcube_minmax (<=)
    @pcube_case var1:
        apply(tab_server1);
    @pcube_endcase
    @pcube_case var2:
        apply(tab_server2);
    @pcube_endcase
    @pcube_case var3:
        apply(tab_server3);
    @pcube_endcase
@pcube_endminmax
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_minmax primitive

Before pcube

```
if(var1 <= var2 and var1 <= var3){
    apply(tab_server1);
}
else if(var2 <= var1 and var2 <= var3){
    apply(tab_server2);
}
else if(var3 <= var1 and var3 <= var2){
    apply(tab_server3);
}
```

After pcube

Relational Op

```
@pcube_minmax (<=)
    @pcube_case var1:
        apply(tab_server1);
    @pcube_endcase
    @pcube_case var2:
        apply(tab_server2);
    @pcube_endcase
    @pcube_case var3:
        apply(tab_server3);
    @pcube_endcase
@pcube_endminmax
```

Case variable

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

Primitives can be nested!

Before pcube

```
if(var1 <= var2 and var1 <= var3){
    apply(tab_server1);
}
else if(var2 <= var1 and var2 <= var3){
    apply(tab_server2);
}
else if(var3 <= var1 and var3 <= var2){
    apply(tab_server3);
}
```

After pcube

Relational Operator

```
@pcube_minmax (<=)
    @pcube_for (i) (1,4,1)
        @pcube_case var$i:
            apply(tab_server$i);
        @pcube_endcase
    @pcube_endfor
@pcube_endminmax
```

Case variable

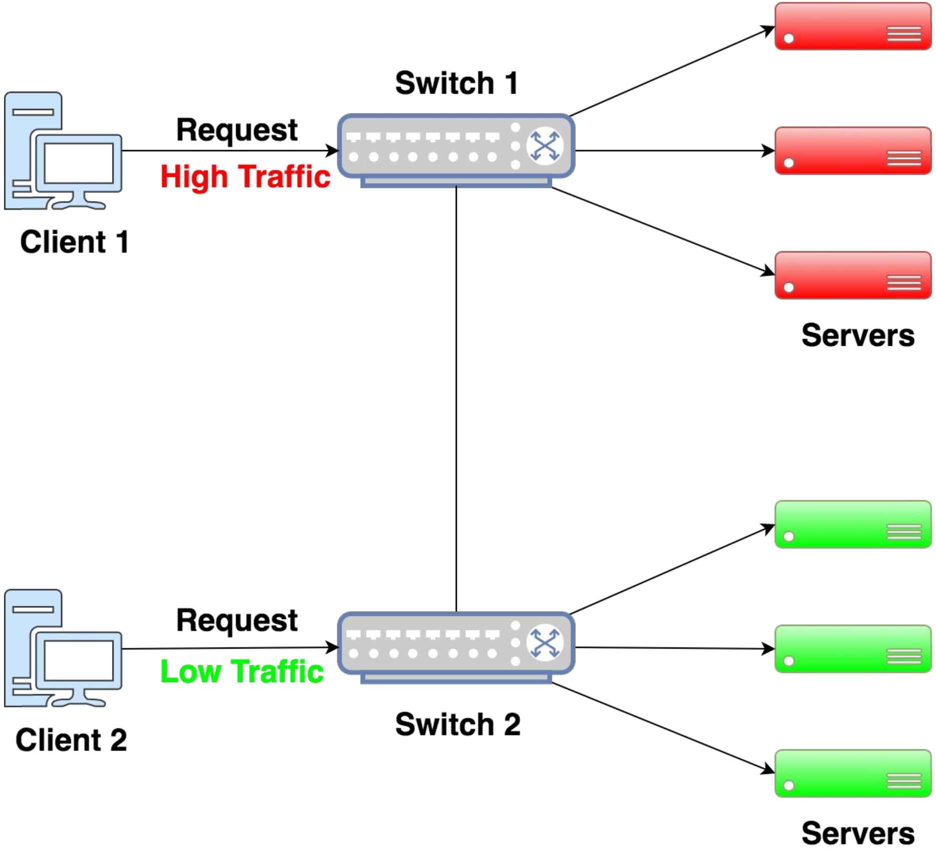
Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

Summary of basic primitives

Type	Annotation	Purpose
Loop	`@pcube_for`	Iterate over indexed code
Minmax	`@pcube_minmax`	Determine the minimum or maximum value from an input list and choose corresponding action
Summation	`@pcube_sum`	Summation over indexed variables
Conditional	`@pcube_cmp`	Conditional test over indexed variables

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

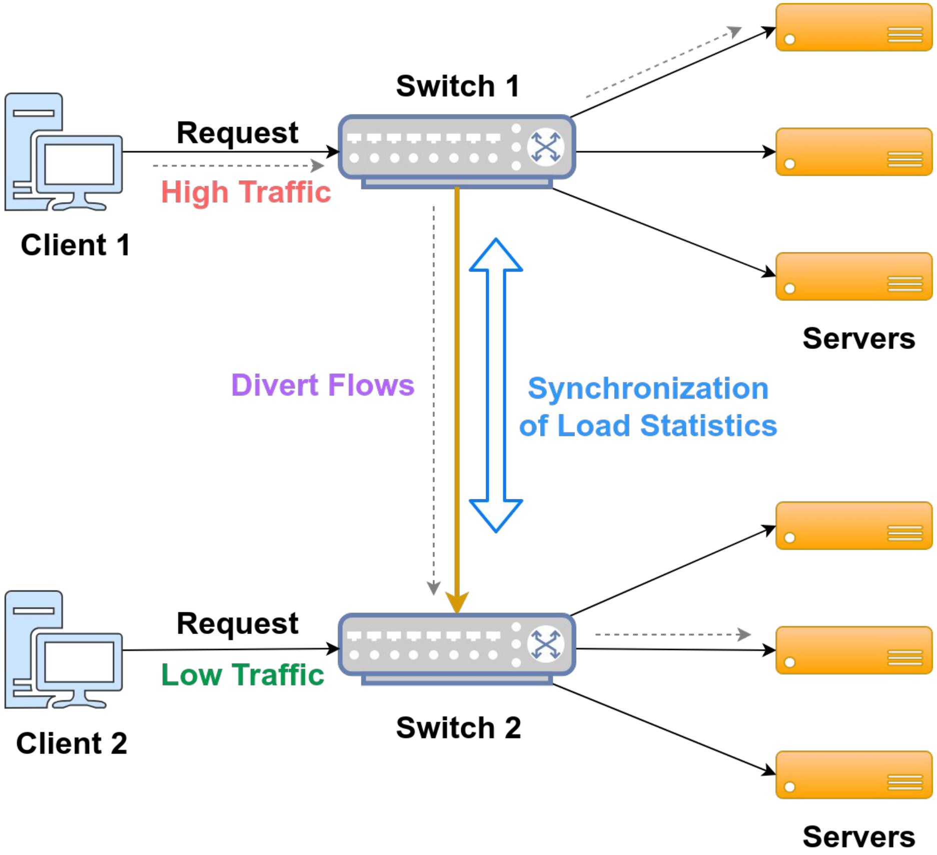
pcube in Distributed Data plane applications



Local decisions

made by switches

pcube in Distributed Data plane applications



Optimal decision

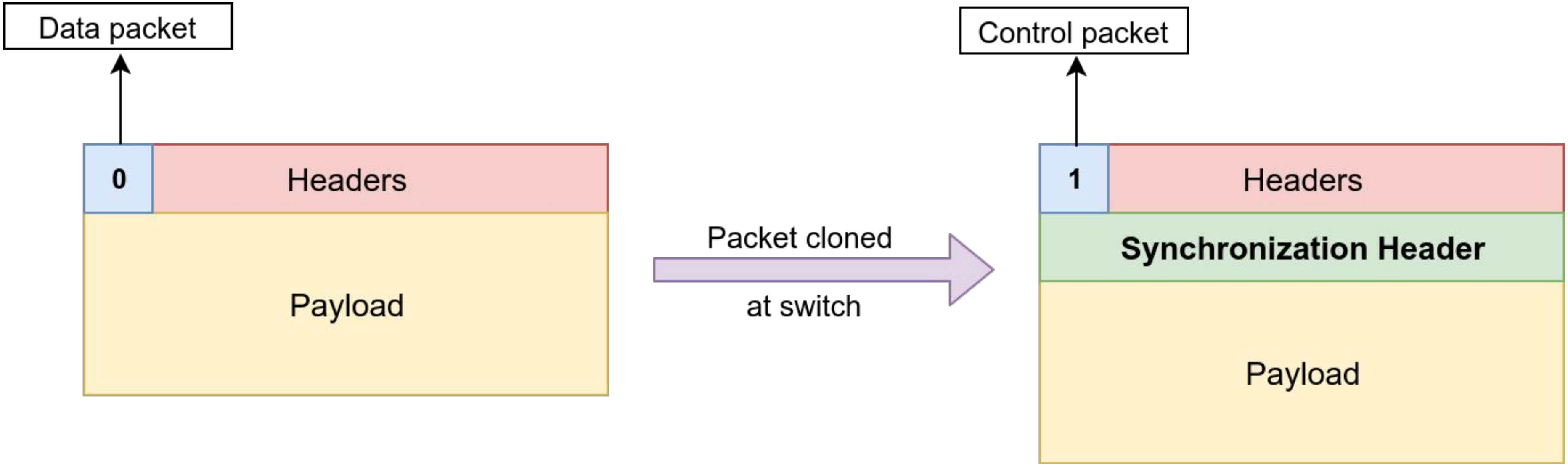
made by switches

Based on global state

Need for state

synchronisation

Switches do not generate packets!



Synchronization with P4 can be cumbersome

```
//Need to define a header
header_type sync_info_t {
    fields{
        _0 : 32;
        _1 : 32;
    }
}
header sync_info_t
sync_info;
```

And this is not even the complete code!

Code size increases with increase in topology size!

```
//Need to define a
table
table sync_info_table {
    actions{
        sync_info;
    }
    size: 1;
}
```

```
//Control logic
if( condition ){
    apply(sync_info_table);
}
```

```
//Need to define an action
action sync_info() {
      clone_ingress_pkt_to_egress(standard_metadata.egress_spec,meta_list);
      modify_field(head.preamble,1);
      modify_field(sync_info._0,info.field1);
      modify_field(sync_info._1,info.field2);
      add_header(sync_info);
      modify_field(intrinsic_metadata.mcast_grp, 2);
}
```

```
//Packet mirroring commands
mirroring_add 1 1
mirroring_add 2 2
mirroring_add 3 3
mirroring_add 4 4
```

```
//Multicast Group Creation Commands
mc_mgrp_create 1
mc_node_create 0 4
mc_node_associate 1 0
```

```
//Command for table
table_set_default sync_info_table sync_info
```

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_sync primitive

After pcube

```
if(condition){
    @pcube_sync(head.preamble,1)
         info.field1
         info.field2
     @pcube_endsync
}
```

Synchronization Header

Match action tables

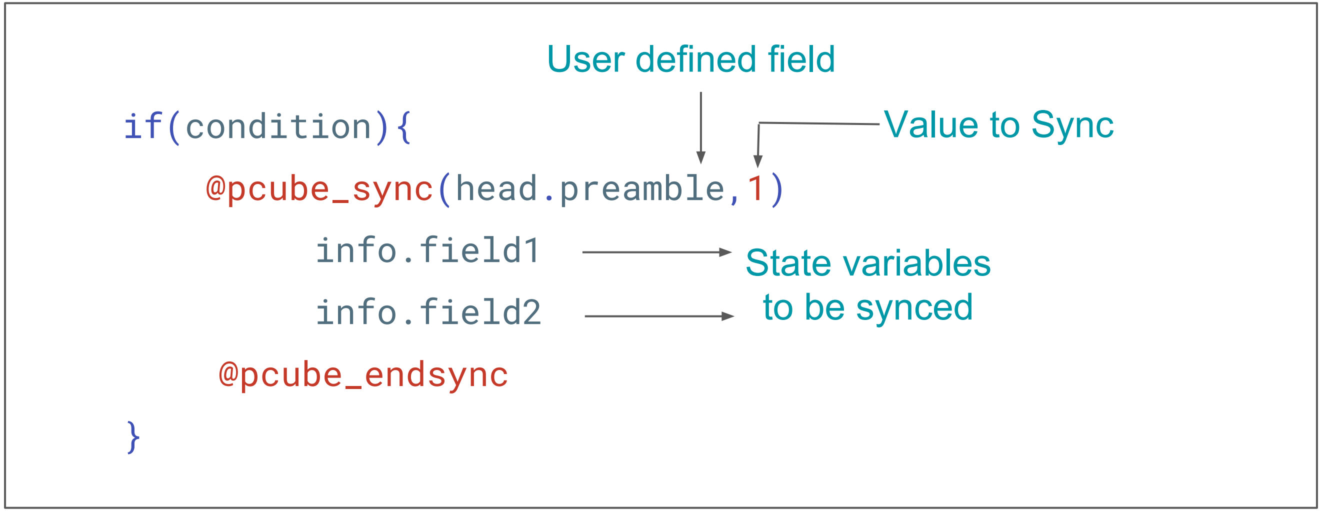
Control plane commands

6 lines of *pcube* code does the job !

Same pcube code for different topology sizes!

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The pcube_sync primitive



```
if(condition){
    @pcube_sync(head.preamble,1)
        info.field1
        info.field2
    @pcube_endsync
}
```

User defined field

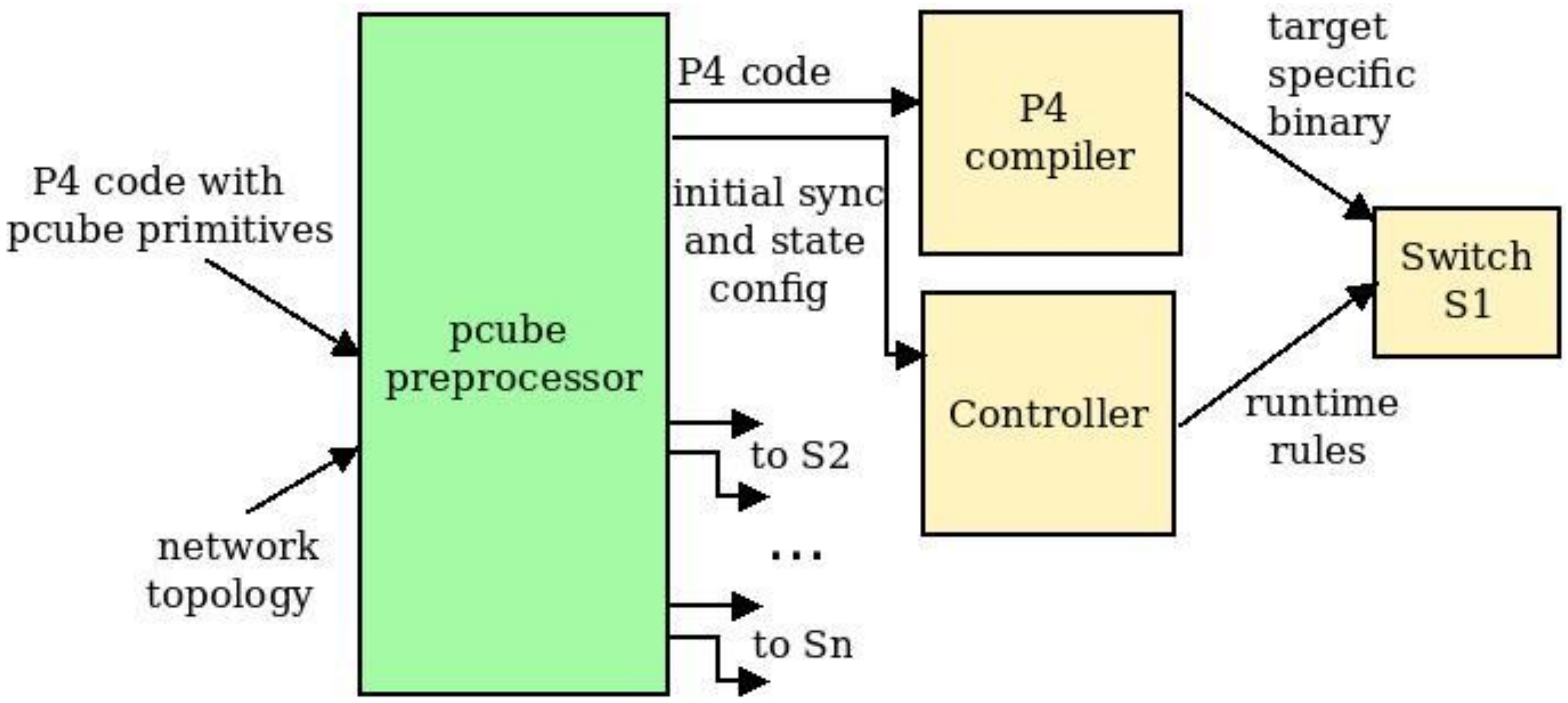
Value to Sync

State variables to be synced

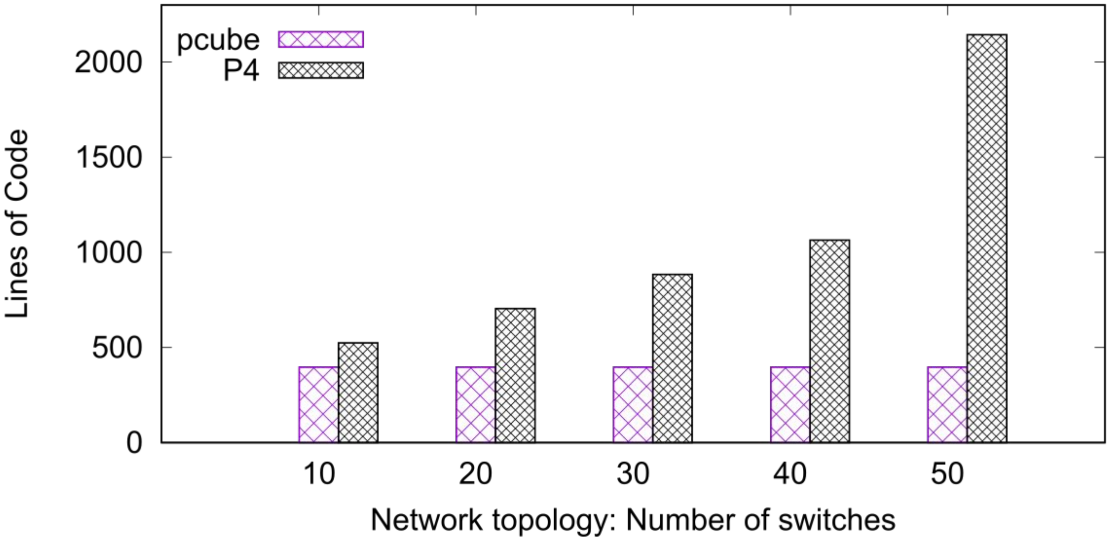
1. User defined header field to identify the synchronisation packets
2. State variables to be synchronised with neighboring switches

Code highlight: https://romannurik.github.io/SlidesCodeHighlighter/

The Implementation



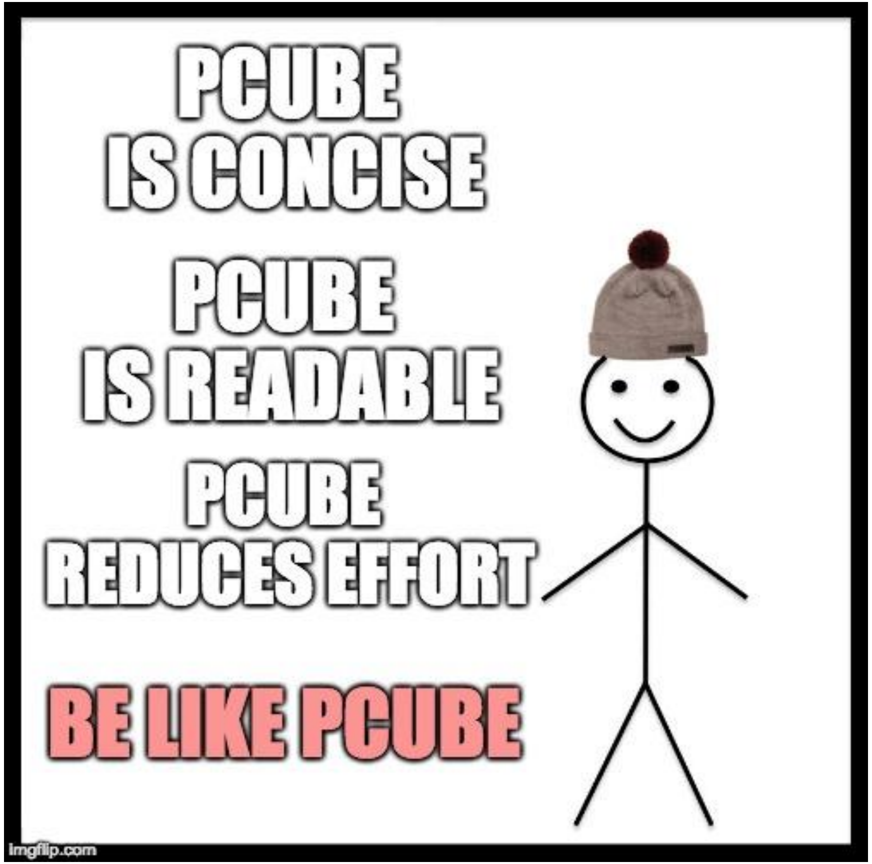
pcube delivers . . .



Application: Distributed Stateful Load Balancer

- Reduced programmer effort in terms of lines of code
  - By 80% in case of Load Balancer with 50 switches

Reduction in LOC by 70% in case of Heavy Hitter with bloom filter bucket size of 50



pcube source code is available at https://github.com/networkedsystemsIITB/pcube