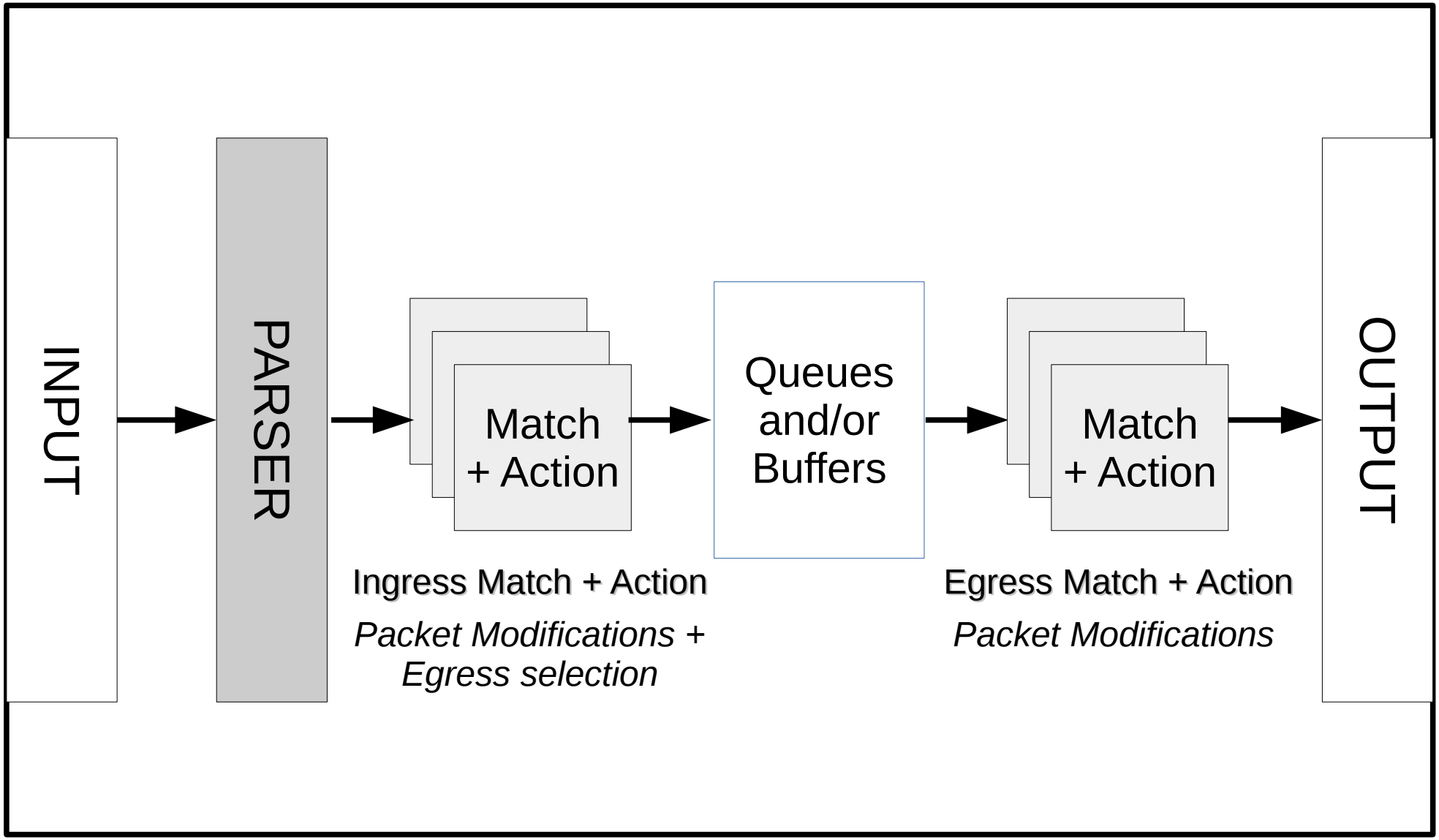
Abstract Dataplane

29-04-2015

P4 language

- Programming Protocol-Independent Packet Processors
- High-level
- More flexible as OpenFlow
- http://p4.org/

P4 model


INPUT
PARSER
Match
+ Action
Queues
and/or
Buffers
Match
+ Action
OUTPUT
Ingress Match + Action
Packet Modifications +
Egress selection
Egress Match + Action
Packet Modifications

P4 source (match +action)

```
control ingress {
    apply(table1);
    apply(table2);
}
```

Sequential execution

```
control egress {
    apply(table3);
    apply(table4);
}
```

Sequential execution

P4 source (match +action)

```
control ... {
    if cond {...}
    else {...}
    ...
    apply (...) {
      apply_match {
          ...
      }
    }
}
```

Control can be more difficult...

Control based on table matching or the applied function

P4 source (match + action)

```
table table_name {
    reads {
        metadata.field : exact;
        header.field : lpm;
    }
    actions {
        action1;
        action2;
    }
    size : TABLE_SIZE;
}
```

Matching type
(exact, terbary, lpm, range, valid)

Fields used by matching

P4 source (match + action)

```
table table_name {
    reads {
        metadata.field : exact;
        header.field : lpm;
    }
    actions {
        action1;
        action2;
    }
    size : TABLE_SIZE;
}
```

Allowed action functions

P4 source (match + action)

```
action fname(params) {
    primitive_action1(params1);
    primitive_action2(params2);
}
```

Simultaneous execution
(according to the specification)

Primitive actions
(from an extensible set)

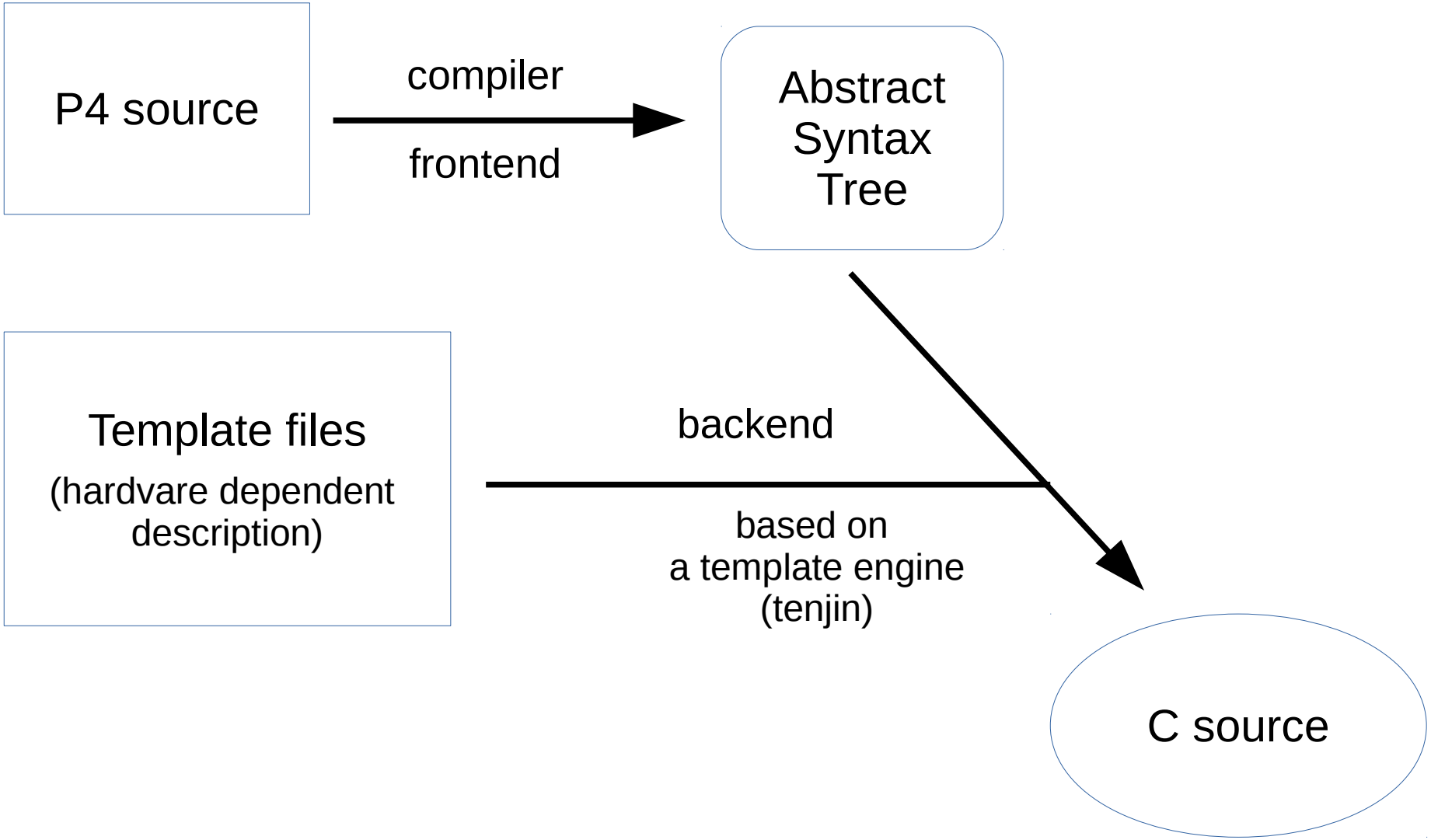
Currently available

- Compiler frontend
- Behavioral compiler backend

  (created by Barefoot Networks, Inc.)
- Some example for the compiler

  (created by Barefoot Networks, Inc.)

Current solution



Template files

```
//:: for x in list_calc():
int function_${x}(params) {
  ...
  //:: ...
 }
```

Python script
(using the AST)

C like code as text

- fast prototype
- very hard to maintain
- no validation (e.g. typecheck)