Lyra: A Cross-Platform Language and Compiler for Data Plane Programming on Heterogeneous ASICS

Jiaqi Gao, Ennan Zhai, Hongqiang Harry Liu, Rui Miao, Yu Zhou
Bingchuan Tian, Chen Sun, Dennis Cai, Ming Zhang, Minlan Yu
Programmable switches gain significant traction

Fixed-Function ASIC

Programmable ASIC
Programmable switches gain significant traction

Fixed-Function ASIC

Parser → L2 Table → IPv4 Table → IPv6 Table → ACL Table

Programmable ASIC

Parser → Memory → ALU → Memory → ALU → Memory → ALU

Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering

SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs

NetChain: Scale-Free Sub-RTT Coordination

P4: Programming Protocol-Independent Packet Processors

HPCC: High Precision Congestion Control


Alibaba Group*, Harvard University*, University of Cambridge*, Massachusetts Institute of Technology*
Programmable switches gain significant traction

Fixed-Function ASIC

Parser → L2 Table → IPv4 Table → IPv6 Table → ACL Table

Programmable ASIC

Parser → Memory ALU → Memory ALU → Memory ALU → Memory ALU

Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering

SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs

NetChain: Scale-Free Sub-RTT Coordination

P4: Programming Protocol-Independent Packet Processors

HPCC: High Precision Congestion Control


Alibaba Group*, Harvard University*, University of Cambridge*, Massachusetts Institute of Technology*
Data plane programming is still at an early stage …
Data plane programming is still at an early stage …

Chip specific languages ~ Assembly languages
Running example: A network sequencer
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Running example: A network sequencer

Linearizing the transactions for consensus protocols such as Paxos

Add sequence header to selected flows and drop others
Problem 1: Portability
Low level, chip-specific languages
Problem 1: Portability

Low level, chip-specific languages
Problem 1: Portability
Low level, chip-specific languages

```
// P4_14
action a_shift_switch_id(server_id) {
    shift_left(sequence.seq, server_id, 28);
}

table shift_switch_id {
    reads {
    ipv4.src_ip: exact;
    }
    actions {
    a_shift_switch_id;
    }
}

action a_and_timestamp() {
    bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}

table and_timestamp {
    actions {
    a_and_timestamp;
    }
    default_action: a_and_timestamp();
}

action a_sequence_header() {
    add_header(sequence);
    bit_and(sequence.seq, sequence.seq, ig_ts);
}

table add_sequence_header {
    actions {
    a_sequence_header;
    }
    default_action: a_sequence_header();
}
```
Problem 1: Portability
Low level, chip-specific languages

```c
// P4_14
action a_shift_switch_id(server_id) {
    shift_left(sequence.seq, server_id, 28);
}
table shift_switch_id {
    reads {
        ipv4.src_ip: exact;
    }
    actions {
        a_shift_switch_id;
    }
}

action a_and_timestamp() {
    bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
    actions {
        a_and_timestamp;
    }
    default_action: a_and_timestamp();
}
action a_sequence_header() {
    add_header(sequence);
    bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
    actions {
        a_sequence_header;
    }
    default_action: a_sequence_header();
}
```
Problem 1: Portability
Low level, chip-specific languages

```c
// P4_14
action a_shift_switch_id(server_id) {
    shift_left(sequence.seq, server_id, 28);
}
table shift_switch_id {
    reads {
        ipv4.src_ip: exact;
    }
    actions {
        a_shift_switch_id;
    }
}

action a_and_timestamp() {
    bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
    actions {
        a_and_timestamp;
    }
    default_action: a_and_timestamp();
}
action a_sequence_header() {
    add_header(sequence);
    bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
    actions {
        a_sequence_header;
    }
    default_action: a_sequence_header();
}
```

Chip Vendors

- Broadcom
- Cisco
- Barefoot Networks
- Mellanox
Problem 1: Portability
Low level, chip-specific languages

Chip Vendors
- BROADCOM
- CISCO
- Barefoot Networks
- Mellanox Technologies

Languages
- NPL

Languages

Chip Vendors

Add ID
Add Timestamp

// P4_14
action a_shift_switch_id(server_id) {
   shift_left(sequence.seq, server_id, 28);
}
table shift_switch_id {
   reads {
      ipv4.src_ip: exact;
   }
   actions {
      a_shift_switch_id;
   }
}

action a_and_timestamp() {
   bit_and(ig_ts, ig_ts, 0x0FFFFFFF);
}
table and_timestamp {
   actions {
      a_and_timestamp;
   }
   default_action: a_and_timestamp();
}
action a_sequence_header() {
   add_header(sequence);
   bit_and(sequence.seq, sequence.seq, ig_ts);
}
table add_sequence_header {
   actions {
      a_sequence_header;
   }
   default_action: a_sequence_header();
}
Problem 2: Extensibility

Cannot program across switches
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
300 entries

None

filter (300)

routing (500)
Problem 2: Extensibility

Cannot program across switches

Aggregation
1000 entries

Top of Rack
1000 entries

Filter table
300 entries

None

Filter (300)
Routing (500)

Filter (800)
Routing (500)

800 entries
Problem 2: Extensibility

Cannot program across switches

- **Aggregation**
  - 1000 entries

- **Top of Rack**
  - 1000 entries

- **Filter table**
  - 300 entries

- **300 entries**
  - filter (300)
  - routing (500)

- **800 entries**
  - filter (800)
  - routing (500)

- **1300 entries**
  - filter (1000)
  - routing (500)
Problem 3: Composition

Co-locate with other programs
Problem 3: Composition

Co-locate with other programs

Add an ARP table with 300 entries to the ToR switch

Aggregation
1000 entries

Top of Rack
1000 entries
Problem 3: Composition

Co-locate with other programs

Add an ARP table with 300 entries to the ToR switch

Aggregation
1000 entries

Top of Rack
1000 entries
Data plane programming is still at an early stage

- **Portability**: Migrate program across switches
  - Different languages
  - Different architectures
  - Different ASIC models

- **Extensibility**: Distribute program across multiple switches
  - Different roles (In-band network telemetry)
  - Expand memory or computation (Sequencer)

- **Composition**: Fit multiple programs into one switch
  - Function overlapping
Data plane programming is still at an early stage

Portability
- Migrate program across switches
  - Different languages
  - Different architectures
  - Different ASIC models

Extensibility
- Distribute program across multiple switches
  - Different roles (In-band network telemetry)
  - Expand memory or computation (Sequencer)

Composition
- Fit multiple programs into one switch
  - Function overlapping

C language lets you get close to the machine, without getting tied up in the machine
  — Dr. Brian Kernighan
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Lyra compiler

NPL (Trident-4)

P4 (Tofino 32Q)

P4 (Tofino 64Q)

P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Portability: Language synthesizer

Lyra compiler

- NPL (Trident-4)
- P4 (Tofino 32Q)
- P4 (Tofino 64Q)
- P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Portability: Language synthesizer

Topology-aware code allocation

Lyra compiler

- NPL (Trident-4)
- P4 (Tofino 32Q)
- P4 (Tofino 64Q)
- P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

- Portability: Language synthesizer
- Topology-aware code allocation
- Chip-specific constraint encoding

Lyra compiler

- NPL (Trident-4)
- P4 (Tofino 32Q)
- P4 (Tofino 64Q)
- P4 (Silicon One)
Lyra language: One-big-pipeline model
Lyra language: One-big-pipeline model

Per-switch model

One big switch

One-big-switch model
Lyra language: One-big-pipeline model

Per-switch model

One-big-pipeline model

One-big-switch model
// Sequencer.lyra
header_type sequence_t{
   bit[32] seq;
}
parser_node parse_sequence{
   extract_fields(sequence);
}
namespace[SEQ] {sequencer};

algorithm sequencer {
   add_sequence();
   routing();
}

func add_sequence() {
   if (ipv4.src_ip in filter) {
      add_header(sequence);
      sequence.seq = (filter[ipv4.src_ip] << 28) \
         & (ig_ts & 0xFFFFFFFF);
   } else {
      drop();
   }
}
Lyra language: Program

```lyra
// Sequencer.lyra
header_type sequence_t{
  bit[32] seq;
}
parser_node parse_sequence{
  extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
  add_sequence();
  routing();
}

func add_sequence() {
  if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) & (ig_ts & 0x0FFFFFFF);
  }
  else {
    drop();
  }
}
```

- Packet
  - Header
  - Parser
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}

pipeline[SEQ] {sequencer};

algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \ 
            & (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
// Sequencer.lyra
header_type sequence_t{
    bit[32] seq;
}
parser_node parse_sequence{
    extract_fields(sequence);
}
pipeline[SEQ] {sequencer};

pipeline[SEQ] {sequencer};
algorithm sequencer {
    add_sequence();
    routing();
}

func add_sequence() {
    if (ipv4.src_ip in filter) {
        add_header(sequence);
        sequence.seq = (filter[ipv4.src_ip] << 28) \n
            & (ig_ts & 0xFFFFFFFF);
    } else {
        drop();
    }
}
// Sequencer.lyra
header_type sequence_t{
  bit[32] seq;
}
parser_node parse_sequence{
  extract_fields(sequence);
}
pipeline[SEQ] {sequencer};
algorithm sequencer {
  add_sequence();
  routing();
}
func add_sequence() {
  if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n      & (ig_ts & 0xFFFFFFFF);
  } else {
    drop();
  }
}
Lyra: A high-level data plane language & compiler

One-big-pipeline model

Lyra program

Lyra compiler

Parser
Preprocessor
Analyzer
Frontend

Language synthesizer
Chip-specific constraint
Extensibility
 Backend

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Lyra compiler

Parser
Preprocessor
Analyzer
Frontend

Language synthesizer
Chip-specific constraint
Extensibility
Backend

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n        & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \n    & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```

**Lyra program**

```c
p1 = ipv4.src_ip in filter;
add_header(sequence);
tmpl1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmpl1 & tmp2;
p2 = !(ipv4.src_ip in filter);
drop();
```

**Intermediate representation**
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) & (ig_ts & 0x0FFFFFFF);
} else {
    drop();
}
```

Lyra program

```
pl = ipv4.src_ip in filter;
add_header(sequence);
tmp1 = filter[ipv4.src_ip] << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmp1 & tmp2;
p2 = !(ipv4.src_ip in filter);
drop();
```

One big pipeline model

No complex statements
No branches
Has statement dependencies

Intermediate representation
```c
extern list<bit[32] ip>[300] filter;
if (ipv4.src_ip in filter) {
    add_header(sequence);
    sequence.seq = (filter[ipv4.src_ip] << 28) \ 
        & (ig_ts & 0x0FFFFFFF);
}
else {
    drop();
}
```

Lyra program

- One big pipeline model
- No complex statements
- No branches
- Has statement dependencies

Intermediate representation
Lyra: A high-level data plane language & compiler

One-big-pipeline model

Lyra program

Lyra compiler

Frontend
- Parser
- Preprocessor
- Analyzer

Backend
- Extensibility
- Chip-specific constraint
- Language synthesizer

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Silicon One)
P4 (Tofino 64Q)
Lyra: A high-level data plane language & compiler

- Language synthesizer
- Chip-specific constraint
- Extensibility
- Backend
Lyra: A high-level data plane language & compiler

Language synthesizer
Chip-specific constraint
Extensibility
Backend

Portability: Compile Lyra into structured low-level languages
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**
- **Backend**

**Portability** I Compile Lyra into structured low level languages

**Composition** I Checks whether the program can fit into the switch
Lyra: A high-level data plane language & compiler

- **Portability**: Compile Lyra into structured low level languages
- **Composition**: Checks whether the program can fit into the switch
- **Extensibility**: Correctly deploy the program in the scope
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

- **Portability**
  Compile Lyra into structured low level languages

- **Composition**
  Checks whether the program can fit into the switch

- **Extensibility**
  Correctly deploy the program in the scope
Languages have different programming paradigms

Table oriented programming

Procedural oriented programming
Languages have different programming paradigms

Table oriented programming

Procedural oriented programming

P4 Program
Languages have different programming paradigms

**Table oriented programming**

- **P4 Program**
  - table 1
  - table 2
  - table 3
  - table 4

**Procedural oriented programming**

- **P4 Table**
  - action 1
  - statement 1
  - statement 2
  - Parallel
  - action 2

**NPL**
Predicate block

Group of statements that has the same predicate and no dependency

\[ p1 = \neg (ipv4\.src\_ip in filter) \]
\[ \text{add\_header(sequence)}; \]
\[ \text{tmp1} = \text{filter}[ipv4\.src\_ip] \ll 28; \]
\[ \text{tmp2} = \text{ig\_ts} \& 0x0FFFFFFF; \]
\[ \text{sequence.seq} = \text{tmp1} \& \text{tmp2}; \]
\[ p2 = !(ipv4\.src\_ip in filter); \]
\[ \text{drop}(); \]
Predicate block

Group of statements that has the same predicate and no dependency

**filter table 1**

**action 1 (input: server_id)**

```
add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
```

**match src_ip**

**action 2**

```
drop();
```

**filter table 2**

**Dependent**

```
sequence.seq = tmp1 & tmp2;
```

**Mutual exclusive**
Lyra: A high-level data plane language & compiler

- **Language synthesizer**
- **Chip-specific constraint**
- **Extensibility**

- **Portability** → Compile Lyra into structured low level languages
- **Composition** → Checks whether the program can fit into the switch
- **Extensibility** → Correctly deploy the program in the scope
Target-specific resource encoding: RMT

**filter table 1**

- **match src_ip**
  - **action 1 (input: server_id)**
    - \texttt{add\_header(sequence);}
    - \texttt{tmp1 = server\_id \ll 28;}
    - \texttt{tmp2 = ig\_ts & 0x0FFFFFFF;}
   
  - **action 2**
    - \texttt{drop();}

**filter table 2**

- **action 1**
  - \texttt{sequence.seq = tmp1 & tmp2;}

---

**Code Examples**

- \texttt{tmp2 = ig\_ts \& 0x0FFFFFFF;}
- \texttt{sequence.seq = tmp1 \& tmp2;}
Target-specific resource encoding: RMT

```
tmp2 = ig_ts & 0x0FFFFFFF;
sequence.seq = tmp1 & tmp2;
add_header(sequence);
drop();
```

```
filter table 1

match src_ip

action 1 (input: server_id)

add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFFF;
```

```
filter table 2

action 1

sequence.seq = tmp1 & tmp2;
```

```
S_{filter_1} < S_{filter_2}
```

S_{filter_1} < S_{filter_2}
Target-specific resource encoding: RMT

filter table 1

match src_ip

action 1 (input: server_id)
add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0xFFFFFFFF;

action 2
drop();

filter table 2

action 1
sequence.seq = tmp1 & tmp2;

$S_{filter_1} < S_{filter_2}$

$0 \leq S_{filter_1} \leq 31$

$0 \leq S_{filter_2} \leq 31$
Target-specific resource encoding: RMT

```plaintext
filter table 1

match src_ip

action 1 (input: server_id)
    add_header(sequence);
    tmp1 = server_id << 28;
    tmp2 = ig_ts & 0x0FFFFFFF;

action 2
    drop();

filter table 2

action 1
    sequence.seq = tmp1 & tmp2;

S_{\text{filter}_1} < S_{\text{filter}_2}
0 \leq S_{\text{filter}_1} \leq 31
0 \leq S_{\text{filter}_2} \leq 31

S_{\text{filter}_1} = 0
S_{\text{filter}_2} = 1
```

SMT solver
Target-specific resource encoding: RMT

match src_ip

filter table 1

action 1 (input: server_id)

add_header(sequence);
tmp1 = server_id << 28;
tmp2 = ig_ts & 0x0FFFFFF;

action 2
drop();

filter table 2

sequence.seq = tmp1 & tmp2;

action 1

SRAM memory
TCAM memory
Packet Header Vector
Table num per stage

$S_{filter_1} < S_{filter_2}$

0 $\leq S_{filter_1} \leq 31$

SMT solver

$S_{filter_1} = 0$

$S_{filter_2} = 1$

0 $\leq S_{filter_2} \leq 31$
Lyra: A high-level data plane language & compiler

- **Portability**: Compile Lyra into structured low level languages
- **Composition**: Checks whether the program can fit into the switch
- **Extensibility**: Correctly deploy the program in the scope
Lyra: A high-level data plane language & compiler

Lyra program

One-big-pipeline model

Lyra compiler

Parser
Preprocessor
Analyzer
Frontend

Language synthesizer
Chip-specific constraint
Extensibility
Backend

NPL (Trident-4)
P4 (Tofino 32Q)
P4 (Tofino 64Q)
P4 (Silicon One)
Evaluation
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P414</th>
<th>Lyra</th>
<th>Synthesized P414</th>
<th>Synthesized NPL</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Loc / Logic LoC</td>
<td>Tables</td>
<td>Actions</td>
<td>Registers</td>
<td>Lyra Loc / Logic LoC</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
<td>207/62</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>193/46</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>197/47</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P4_14</th>
<th>Lyra</th>
<th>Synthesized P4_14</th>
<th>Synthesized NPL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LoC/Logic LoC</td>
<td>Tables</td>
<td>Actions</td>
<td>Registers</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

<table>
<thead>
<tr>
<th>Program</th>
<th>P4₁₄</th>
<th>Lyra</th>
<th>Synthesized P4₁₄</th>
<th>Synthesized NPL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LoC / Logic LoC</td>
<td>Tables</td>
<td>Actions</td>
<td>Registers</td>
</tr>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
</tr>
</tbody>
</table>
Lyra can reduce resource usage

### Lyra

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
<th>P414</th>
<th>Logic LoC / Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td></td>
<td></td>
<td>154/77</td>
<td>33/90</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td></td>
<td></td>
<td>142/73</td>
<td>32/90</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td></td>
<td></td>
<td>187/71</td>
<td>36/98</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
<td>6</td>
<td>194/97</td>
<td>1.352s</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
<td>40</td>
<td>372/153</td>
<td>1.909s</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>177/73</td>
<td>1.530s</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
<td>5</td>
<td>150/69</td>
<td>1.158s</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>113/43</td>
<td>0.91s</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>72/31</td>
<td>0.852s</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
<td>0</td>
<td>4151/2563</td>
<td>33.6s</td>
</tr>
</tbody>
</table>

### Synthesized NPL

<table>
<thead>
<tr>
<th>Registers</th>
<th>Lyra Compile Time</th>
<th>Tables</th>
<th>Registers</th>
<th>Longest Code Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.78s</td>
<td>4</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>0</td>
<td>0.72s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>0.73s</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>0.95s</td>
<td>9</td>
<td>6</td>
<td>18</td>
</tr>
<tr>
<td>40</td>
<td>1.17s</td>
<td>3</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>0.85s</td>
<td>6</td>
<td>2</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>0.84s</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0.70s</td>
<td>4</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>0</td>
<td>0.67s</td>
<td>3</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>19.4s</td>
<td>125</td>
<td>0</td>
<td>53</td>
</tr>
</tbody>
</table>
## NetCache

### Program

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
</tr>
</tbody>
</table>

### P414

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
</tr>
</tbody>
</table>

### Longest Code Path

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
</tr>
</tbody>
</table>

### Compile Time

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
</tr>
</tbody>
</table>

### Registers

<table>
<thead>
<tr>
<th>Program</th>
<th>LoC / Logic LoC</th>
<th>Tables</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ingress INT</td>
<td>308/99</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>Transit INT</td>
<td>275/66</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Egress INT</td>
<td>282/73</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Speedlight</td>
<td>453/351</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>NetCache</td>
<td>1137/937</td>
<td>96</td>
<td>96</td>
</tr>
<tr>
<td>NetChain</td>
<td>319/211</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>NetPaxos</td>
<td>241/140</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>flowlet_switching</td>
<td>195/130</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>simple_router</td>
<td>101/66</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>switch</td>
<td>4924/3876</td>
<td>131</td>
<td>363</td>
</tr>
</tbody>
</table>

### Lyra can reduce resource usage

Lyra can reduce resource usage in various network applications. For example, in NetCache, using Lyra, the code path is significantly reduced, from 372/153 to 372/153, leading to a 1.352s compile time. In other programs like simple_router and switch, the resource usage is also reduced, optimizing performance and efficiency.
Conclusion

- Lyra is the first high-level data plane language and compiler that achieves portability, extensibility and composition.

- Lyra offers a one-big-pipeline programming model and can generate runnable chip-specific code across multiple switches.

- The programs generated by Lyra use fewer hardware resources than human-written programs.
Thanks !