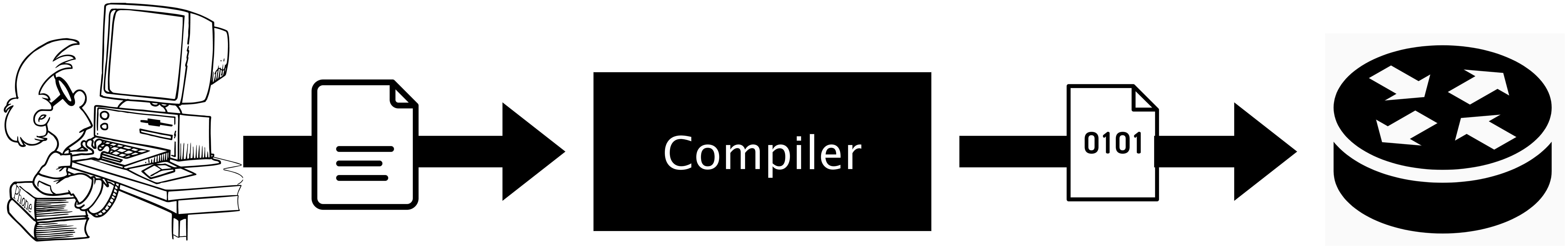
Have an awesome idea
Write it in P4
Cannot fit in hardware
Cannot fit in hardware
imgflip.com

Why is this so hard?



Compiler

0101

Programmer

high-level P4 code

Switch

Limited resources

Who should fix this?

Who should fix this?

Compiler

Who should fix this?

Knows intended functionality

Ignores hardware specifics

Compiler

Who should fix this?

Compiler

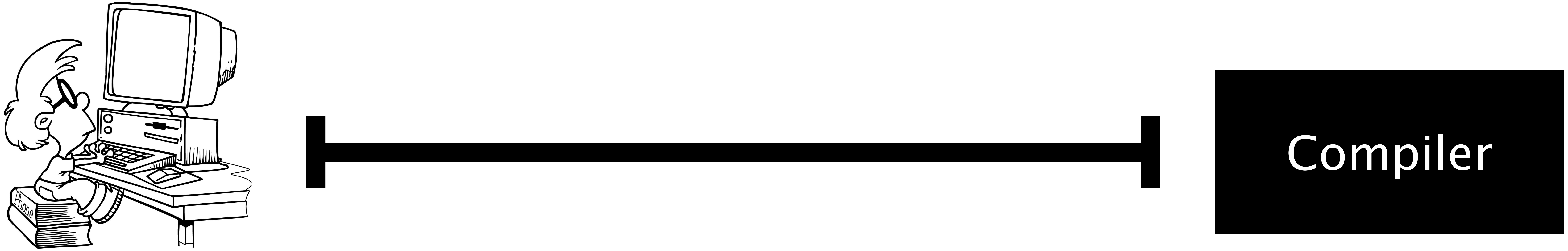
Knows intended functionality

Ignores hardware specifics

Knows hardware constraints

Assumes all possible inputs

There is a gap between the programmer and the compiler



Compiler

Knows intended functionality

Ignores hardware specifics

Knows hardware constraints

Assumes all possible inputs

P^{2}GO bridges the gap between the programmer and the compiler

P^{2}GO

Compiler

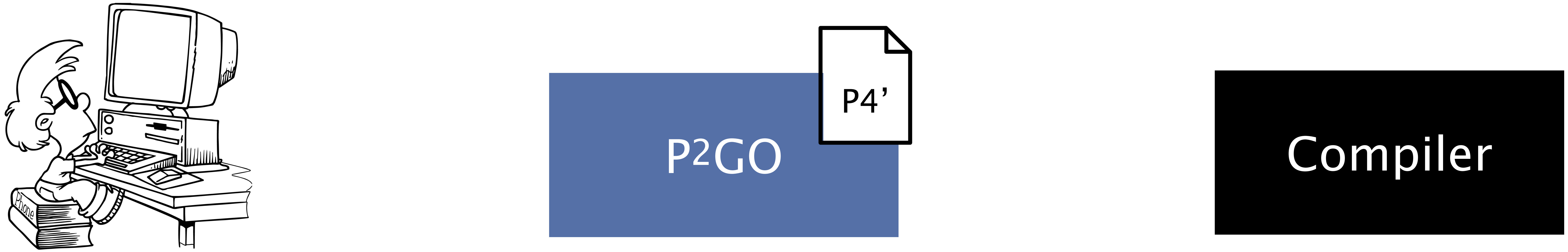
Knows intended functionality

Ignores hardware specifics

Knows hardware constraints

Assumes all possible inputs

P^2GO optimizes the P4 programs for the expected case



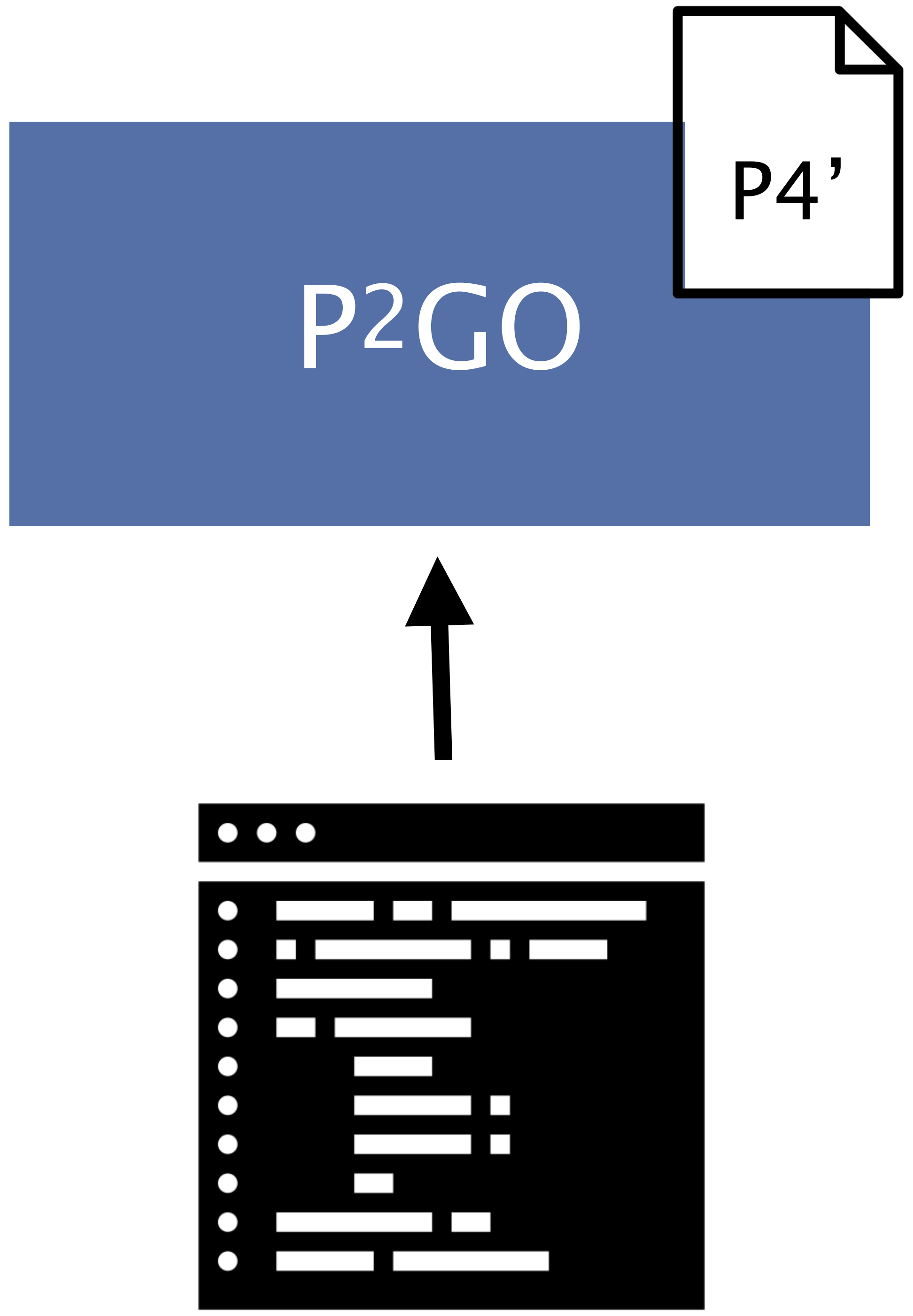
Knows intended functionality
Ignores hardware specifics

Knows hardware constraints
Assumes all possible inputs

P^{2}GO uses the program’s profile to approximate the program’s intended functionality

Knows intended functionality

Ignores hardware specifics



P^{2}GO

P4’

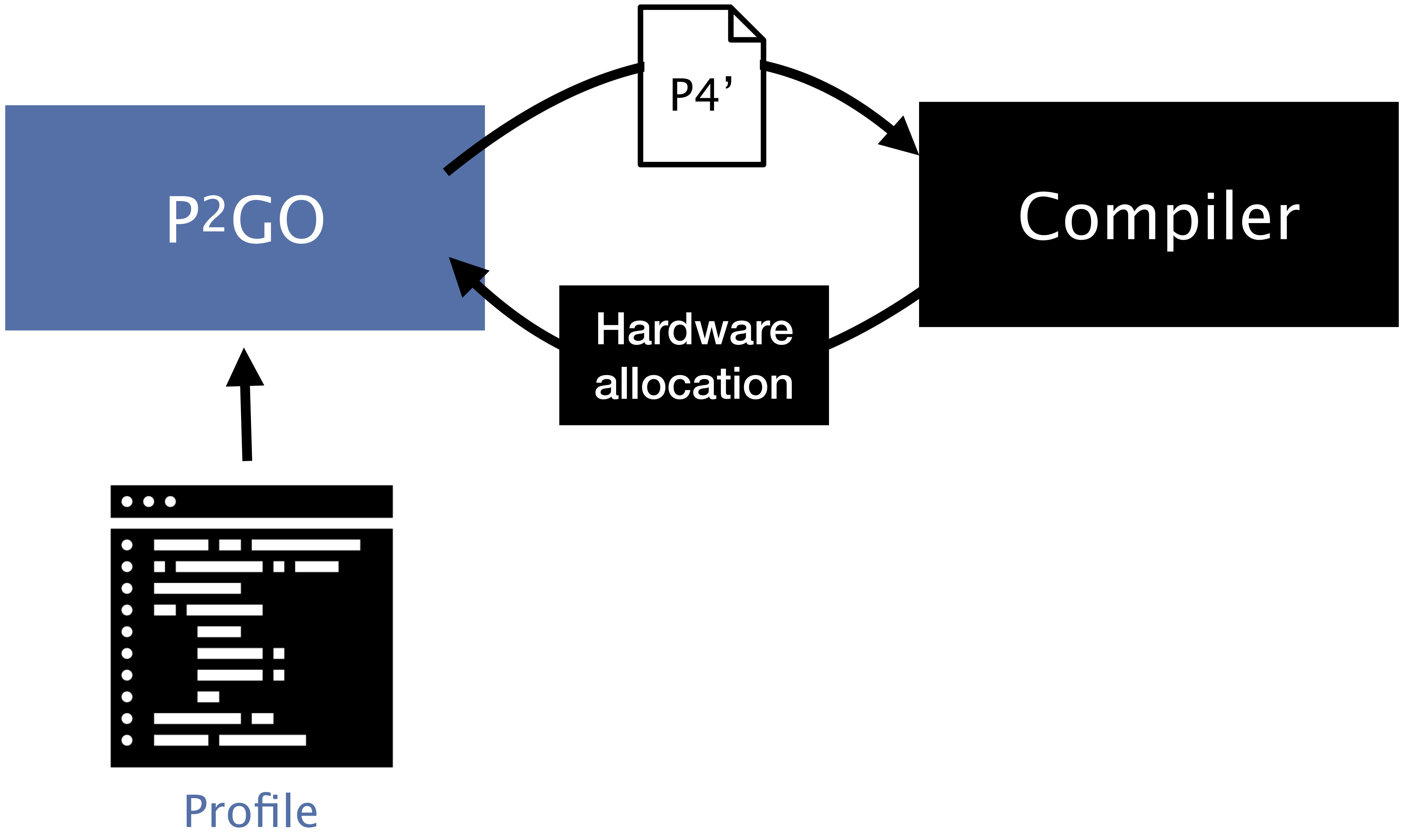
Profile

Compiler

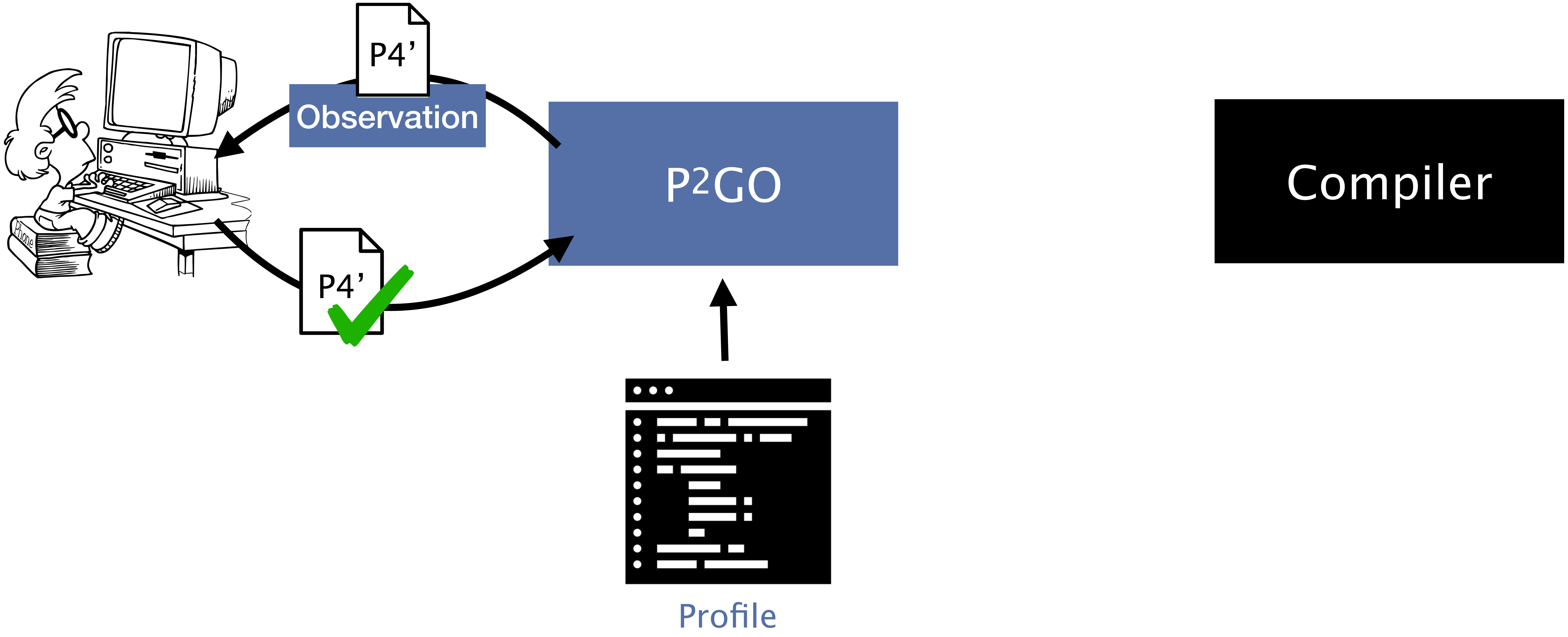
Knows hardware constraints

Assumes all possible inputs

P^{2}GO probes the compiler to check that P4’ uses less hardware resources



P2GO asks the programmer to verify that P4’ satisfies the intended functionality



P^{2}GO: P4 Profile-Guided Optimizations

ETH Zürich

Patrick Wintermeyer

Maria Apostolaki

Alexander Dietmüller

Laurent Vanbever

P^{2}GO: **P**4 **P**rofile-**G**uided **O**ptimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary evaluation

Open research questions

P^2**GO**: **P**4 **P**rofile-**G**uided **O**ptimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary evaluation

Open research questions

What is the program’s profile?

What is the program’s profile?

…is a description of the program’s behavior during runtime

What is the program’s profile?

…is a description of the program’s behavior during runtime

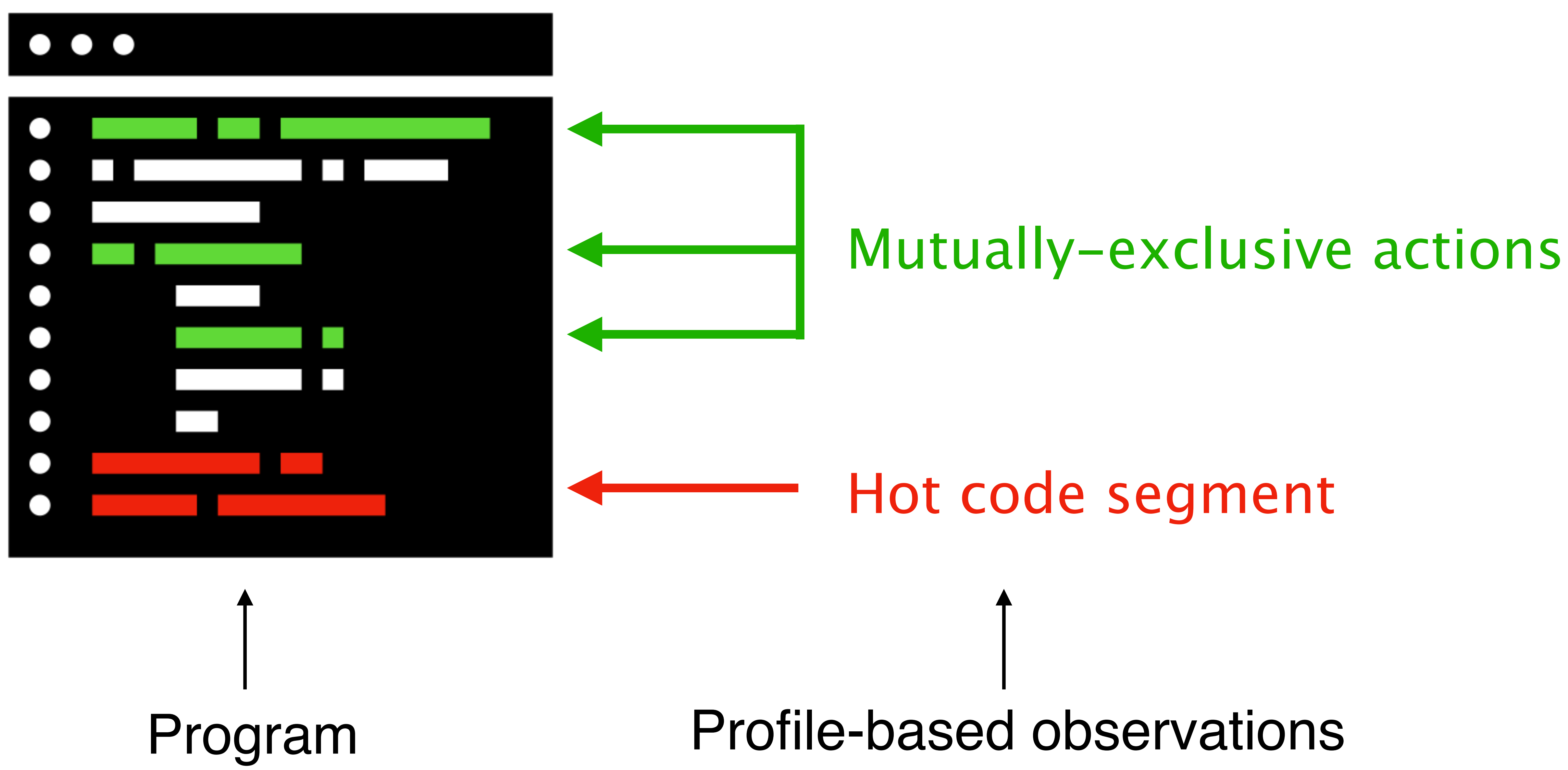
…contains the control paths that packets
of a realistic traffic trace take in the program’s control logic

What is the program’s profile?

…is a description of the program’s behavior during runtime

…contains the control paths that packets
of a realistic traffic trace take in the program’s control logic

What is the program’s profile?



Mutually-exclusive actions

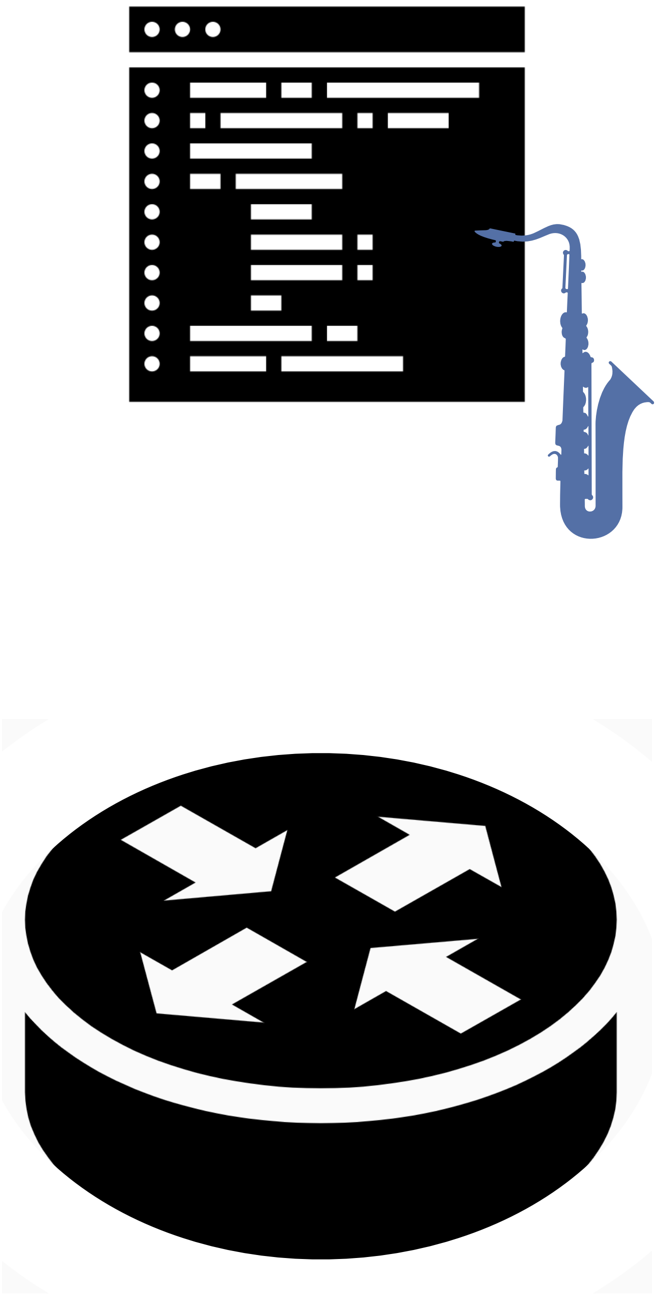
Hot code segment

Program

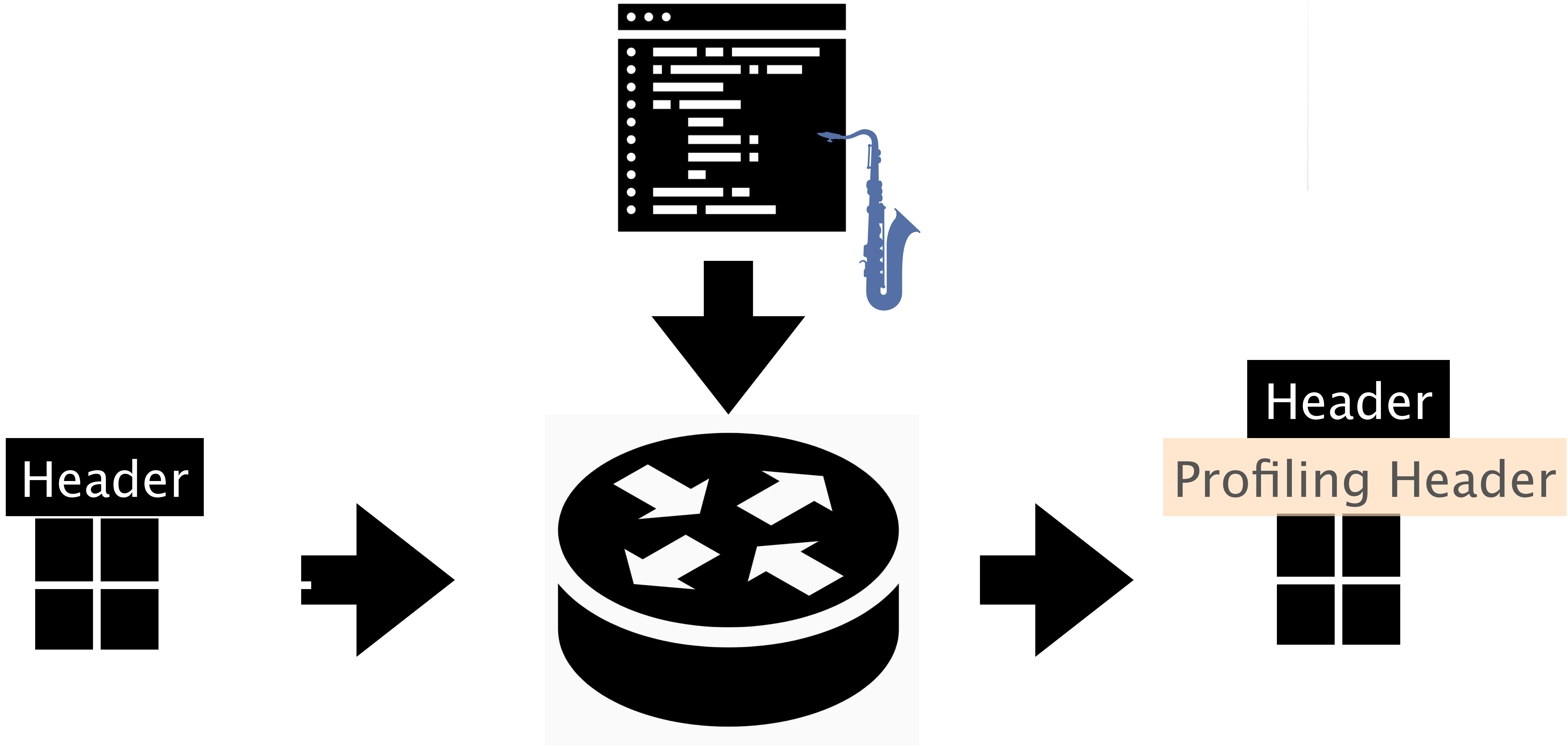
Profile-based observations

P^{2}GO obtains the profile offline

To obtain the profile P^{2}GO first instruments the P4 program



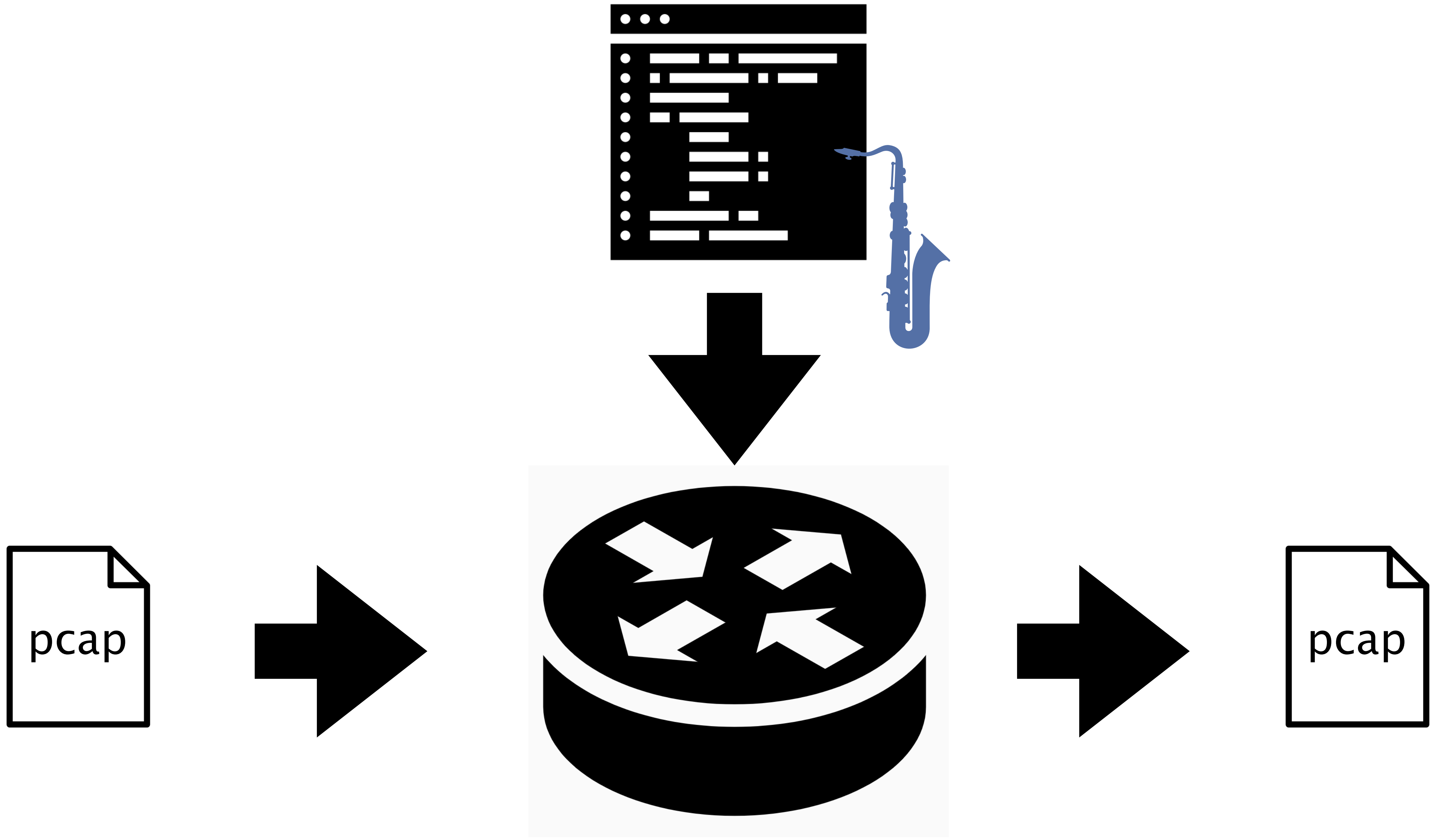
To obtain the profile P^{2}GO first instruments the P4 program



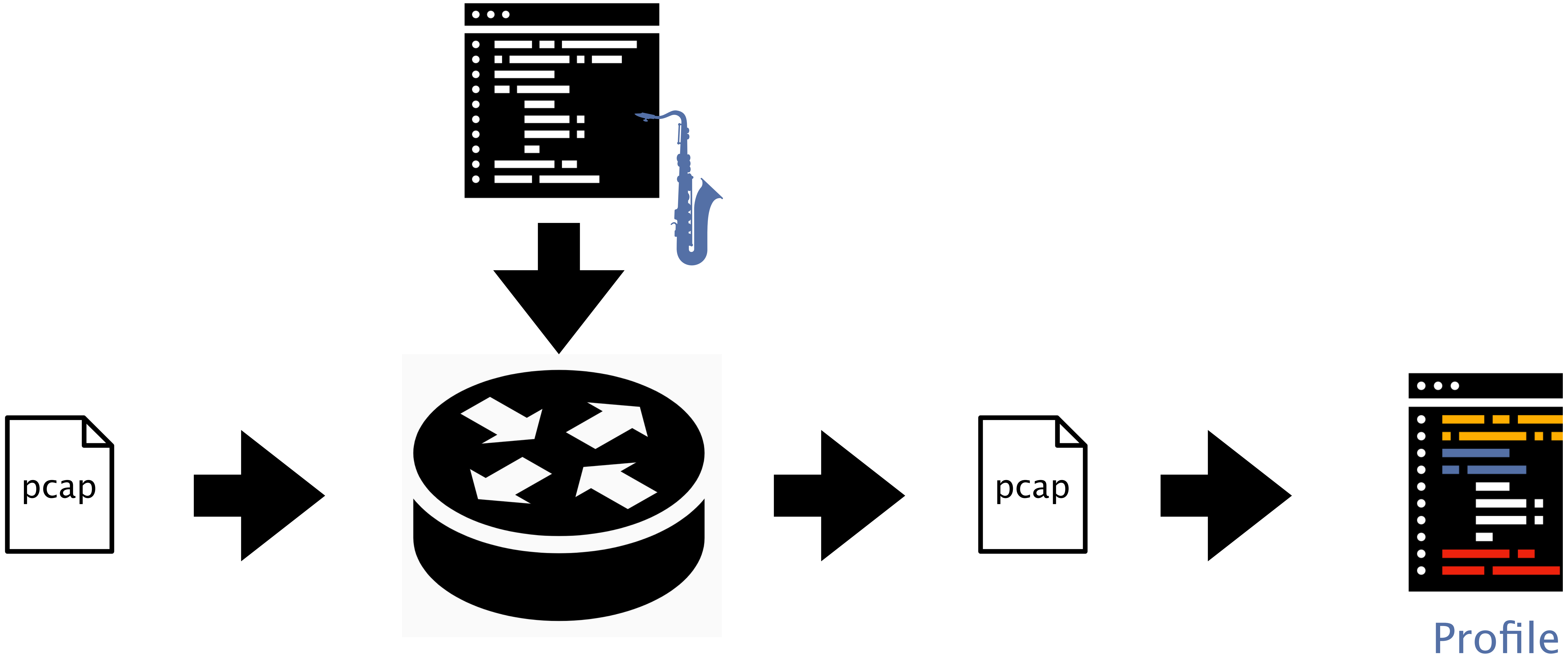
This packet passed though actions
X, Y, Z. . .

To obtain the profile P^{2}GO first instruments the P4 program

Next P^{2}GO runs it on a traffic trace and collects the output



The profile contains the hit rate of each action and the non-mutually exclusive actions



OPTIMIZATIONS
OPTIMIZATIONS EVERYWHERE
imgflip.com

P^2GO uses three profile-guided optimizations

to reduce the number of stages occupied by a P4 program

P^{2}GO uses three profile-guided optimizations to reduce the number of stages occupied by a P4 program

Increase pipeline concurrency

Reduce resource waste

Improve hardware-software split

The programmer implements a program in P4

```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
 MAGIC_1
 MAGIC_2
 MAGIC_3
```

The programmer implements a program in P4

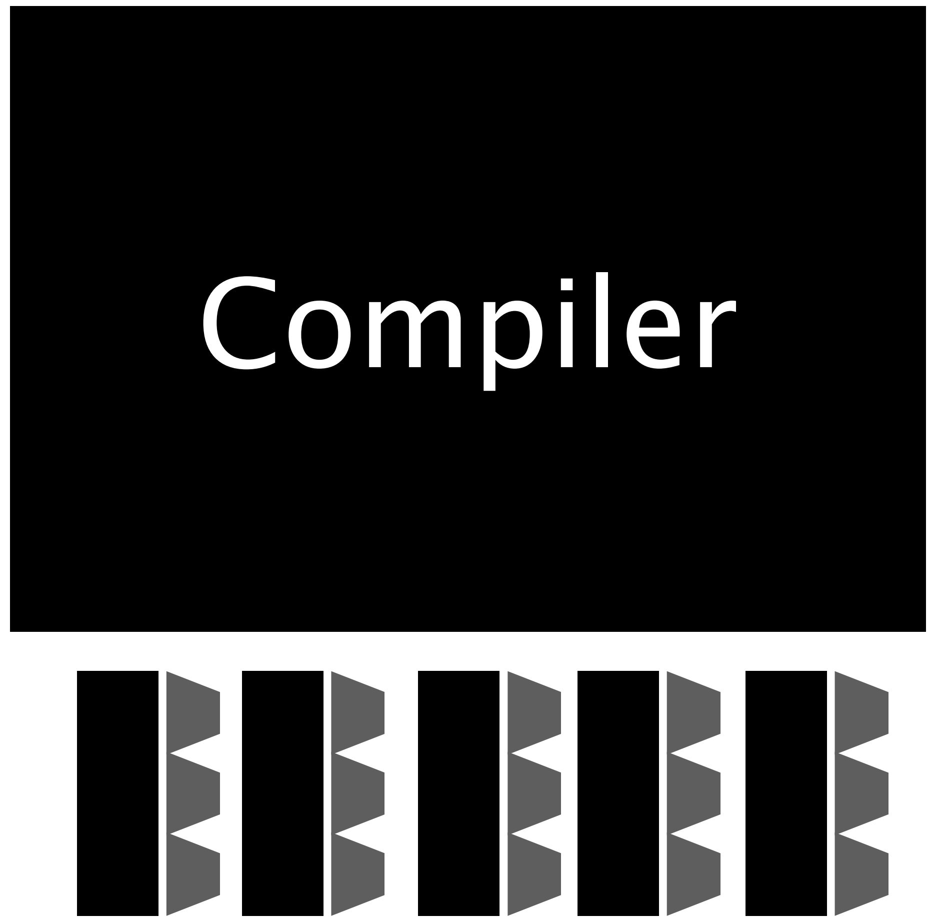
```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
 MAGIC_1
 MAGIC_2
 MAGIC_3
```

The programmer implements a program in P4

```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
 MAGIC_1
 MAGIC_2
 MAGIC_3
```

The compiler maps the program to hardware using five stages

```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
 MAGIC_1
 MAGIC_2
 MAGIC_3
```

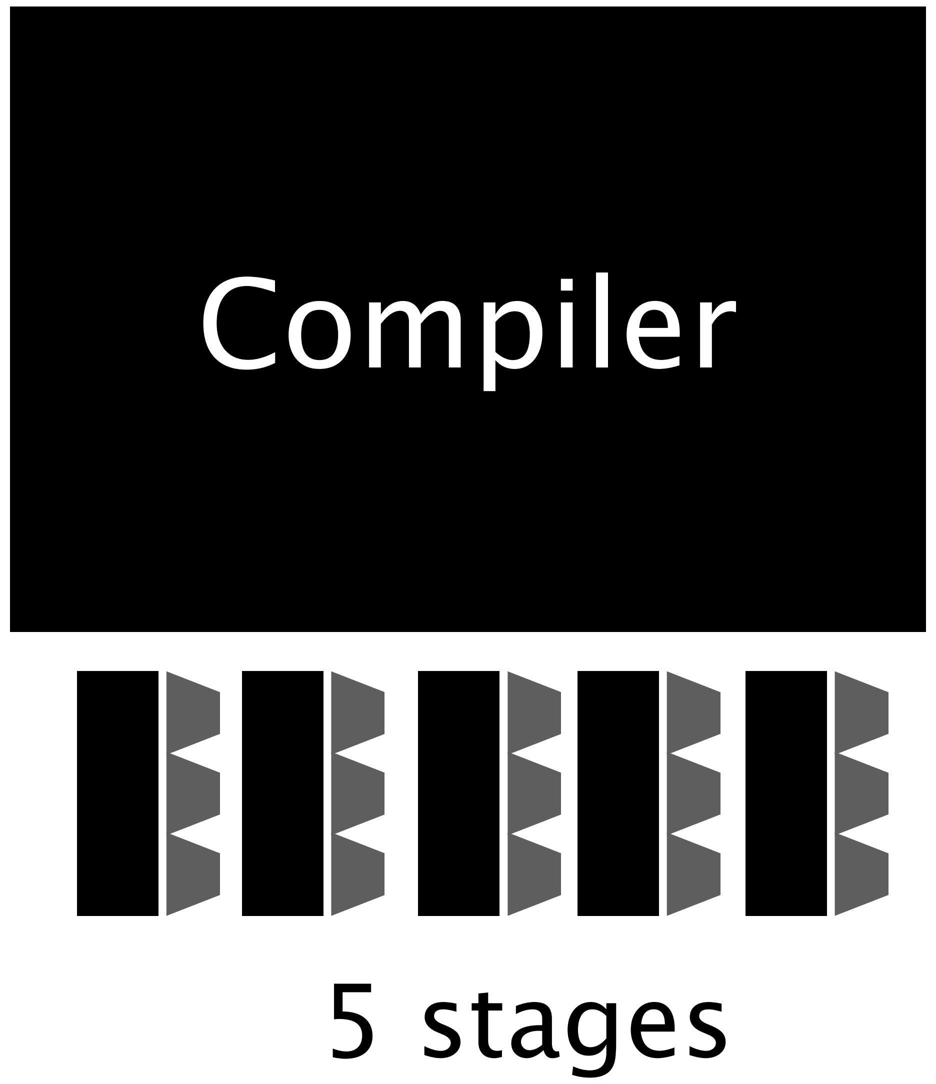


5 stages

SYNFWD needs to precede the execution of MAGIC_1

```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
  MAGIC_1
  MAGIC_2
  MAGIC_3
```

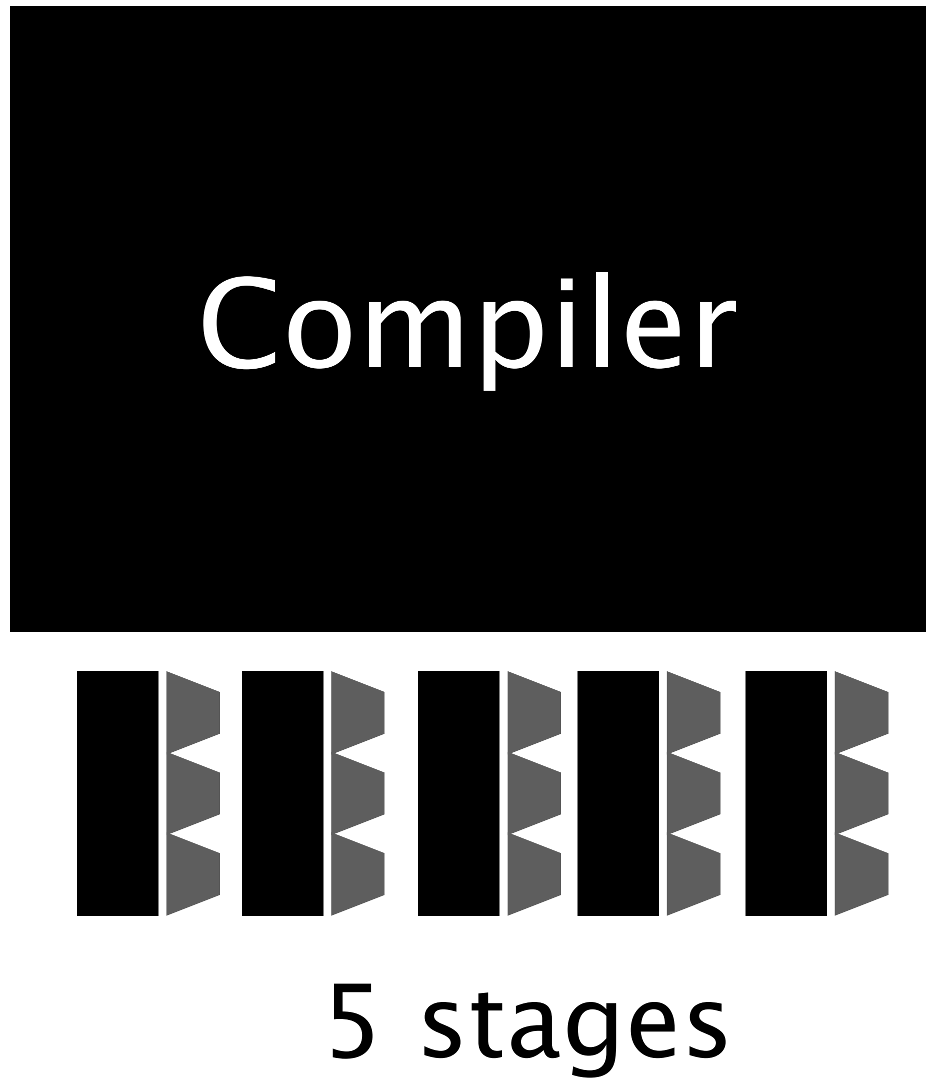
To execute MAGIC_* we need to have results of SYNFWD.



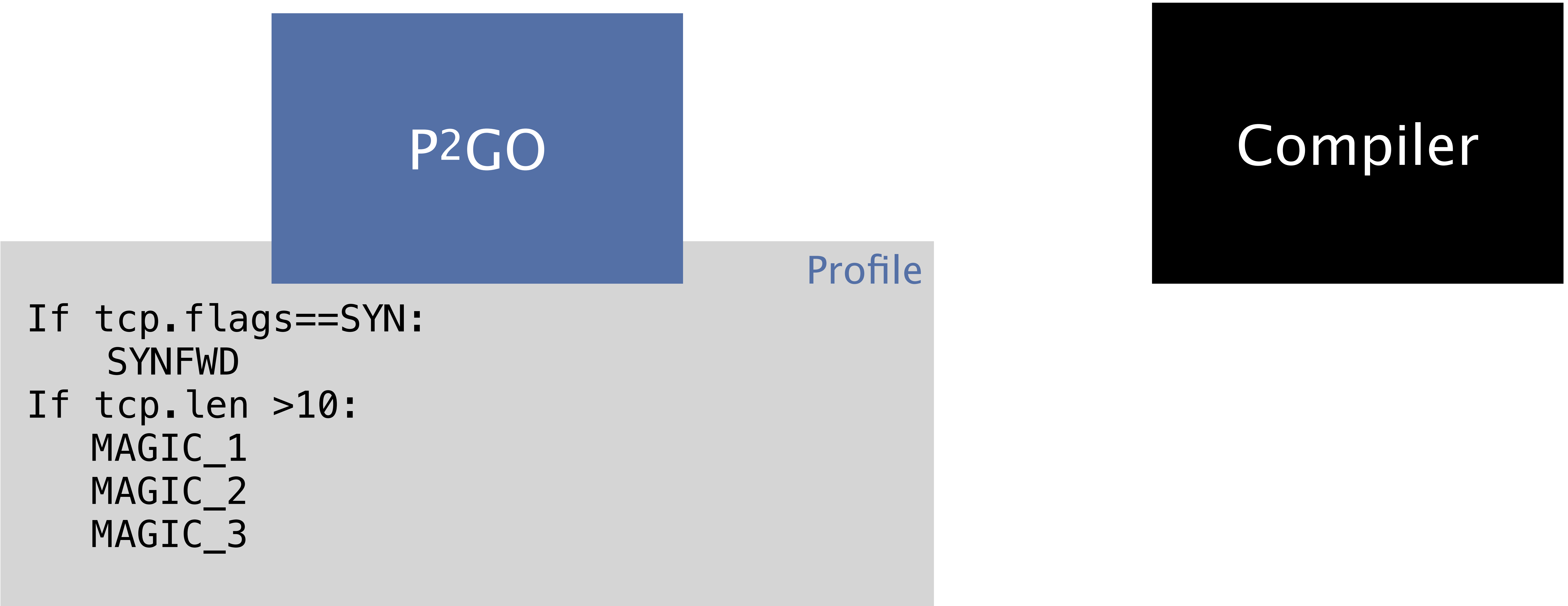
MAGIC_2 uses more memory than is available on a single stage

```
If tcp.flags==SYN:
  SYNFWD
If tcp.len >10:
 MAGIC_1
 MAGIC_2
 MAGIC_3
```

Too many M&A rules.



P^{2}GO uses the program’s profile and the compiler’s output to reduce the number of stages used by the example program



```
If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

P^{2}GO uses the profile to reduce the number of stages, while not changing the program's semantic

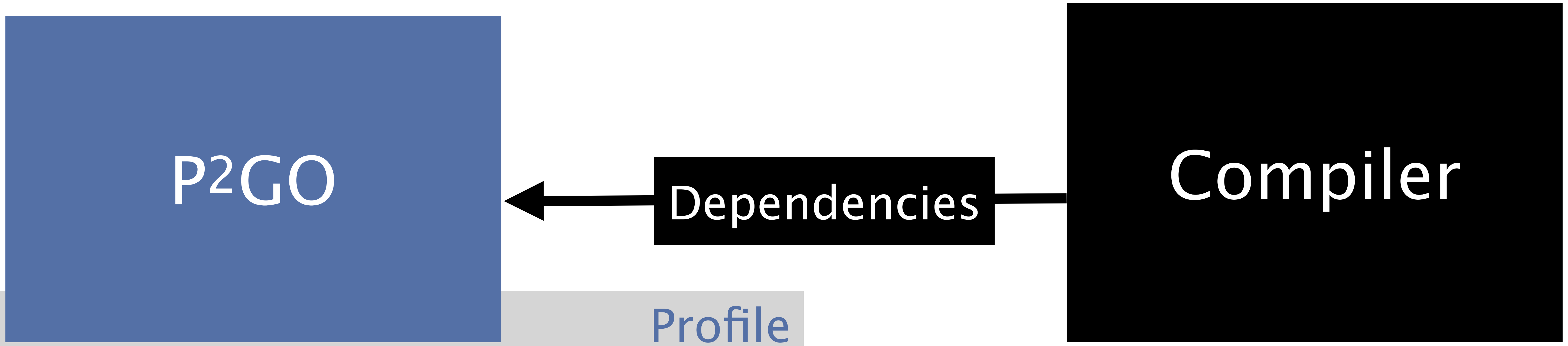
Increase pipeline concurrency

Reduce resource waste

Improve hardware–software split

remove fake dependencies

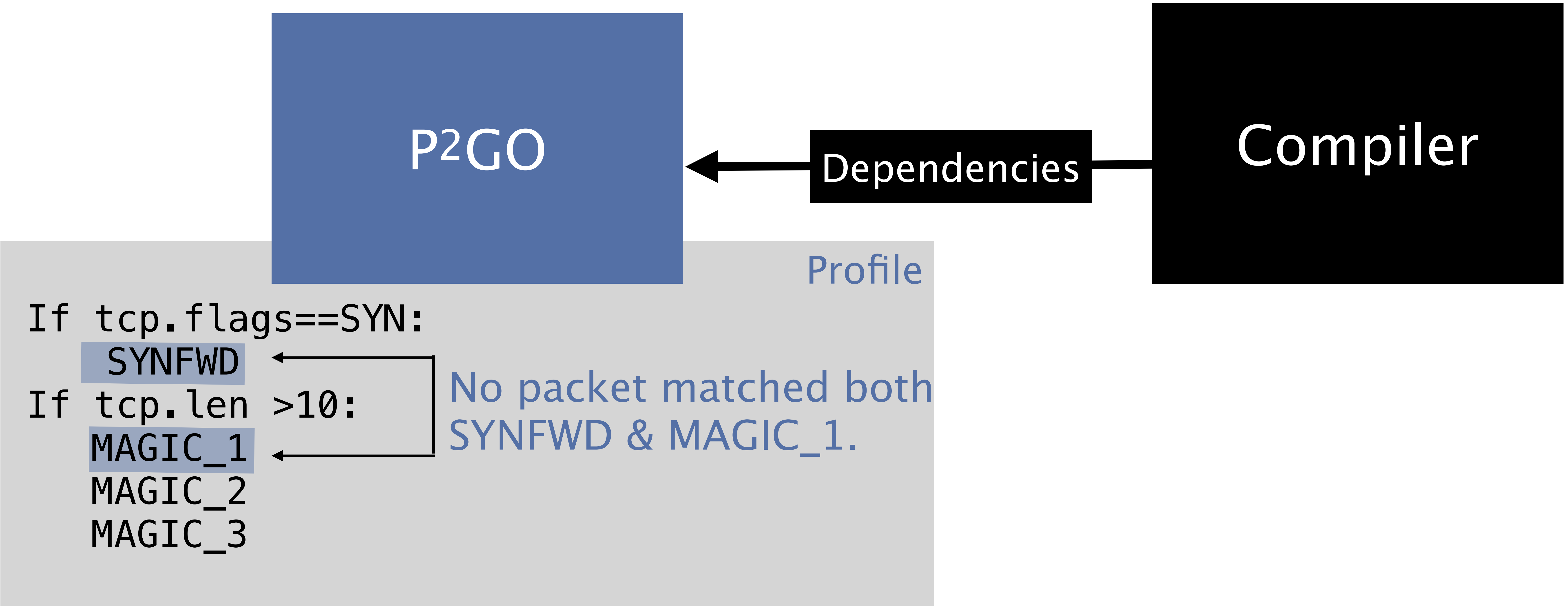
P^{2}GO extracts the program’s dependencies from the compiler



Profile

```
If tcp.flags==SYN:
    SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

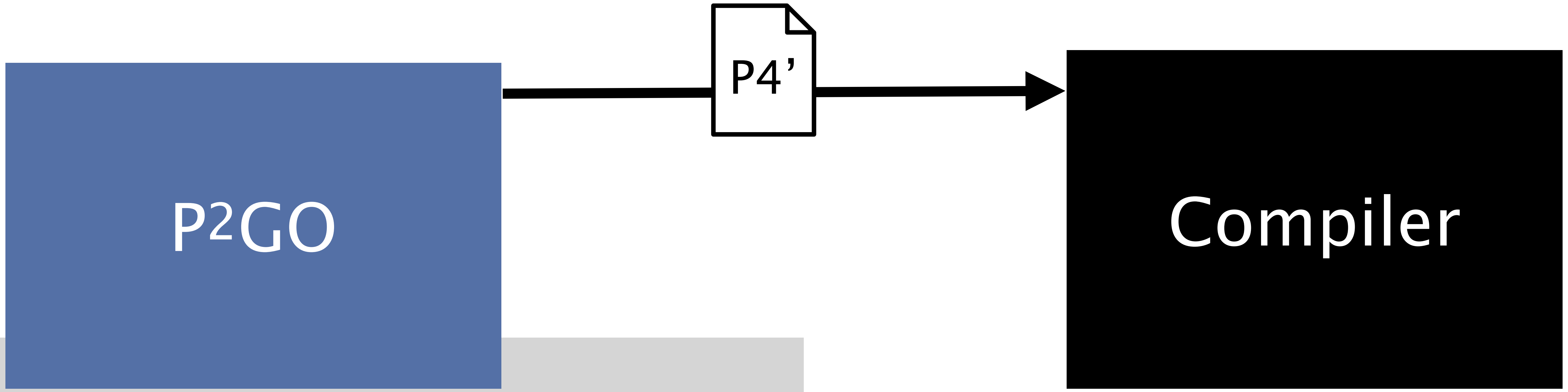
P^{2}GO compares the profile with the dependencies of the static analysis



```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

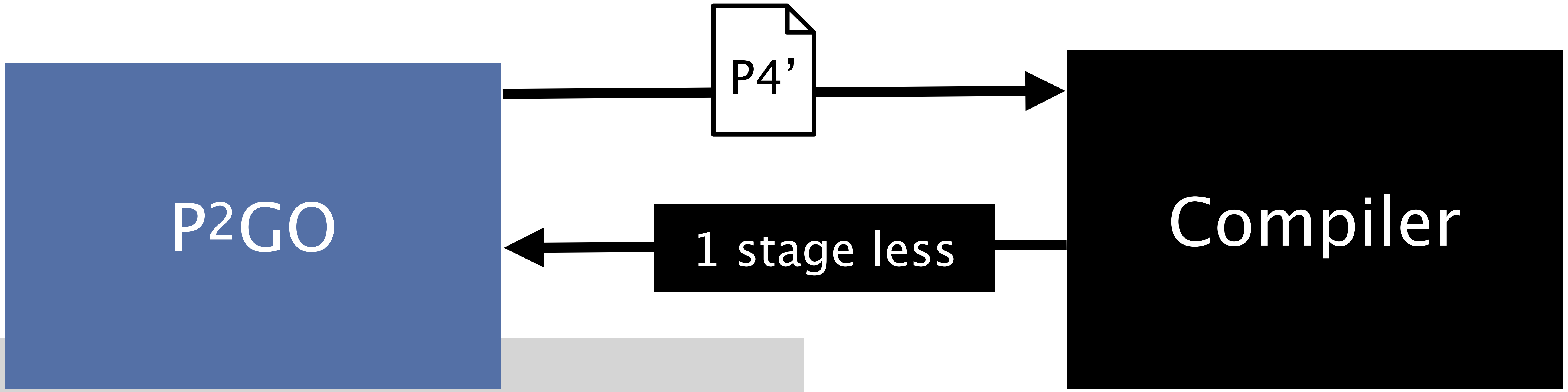
No packet matched both SYNFWD & MAGIC_1.

P^{2}GO automatically generates a new program with the dependency resolved



```
If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

P^{2}GO verifies that the change will reduce the hardware allocation



```
If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

P^{2}GO asks the programmer to accept the modification, explaining the profile-based observation that triggered it

If you are sure that SYNFWD & MAGIC_1 are never applied to the same packet you can gain a stage.

P^{2}GO

Compiler

```
If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

The programmer examines and accepts the modification

If you are sure that SYNFWD & MAGIC_1 are never applied to the same packet you can gain a stage.

Ah yes, SYN packets have zero payload

P^{2}GO

Compiler

```
If tcp.flags==SYN:
    SYNFWD
Else If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

P^{2}GO uses the profile to reduce the number of stages, while not changing the program’s semantic

Increase pipeline concurrency

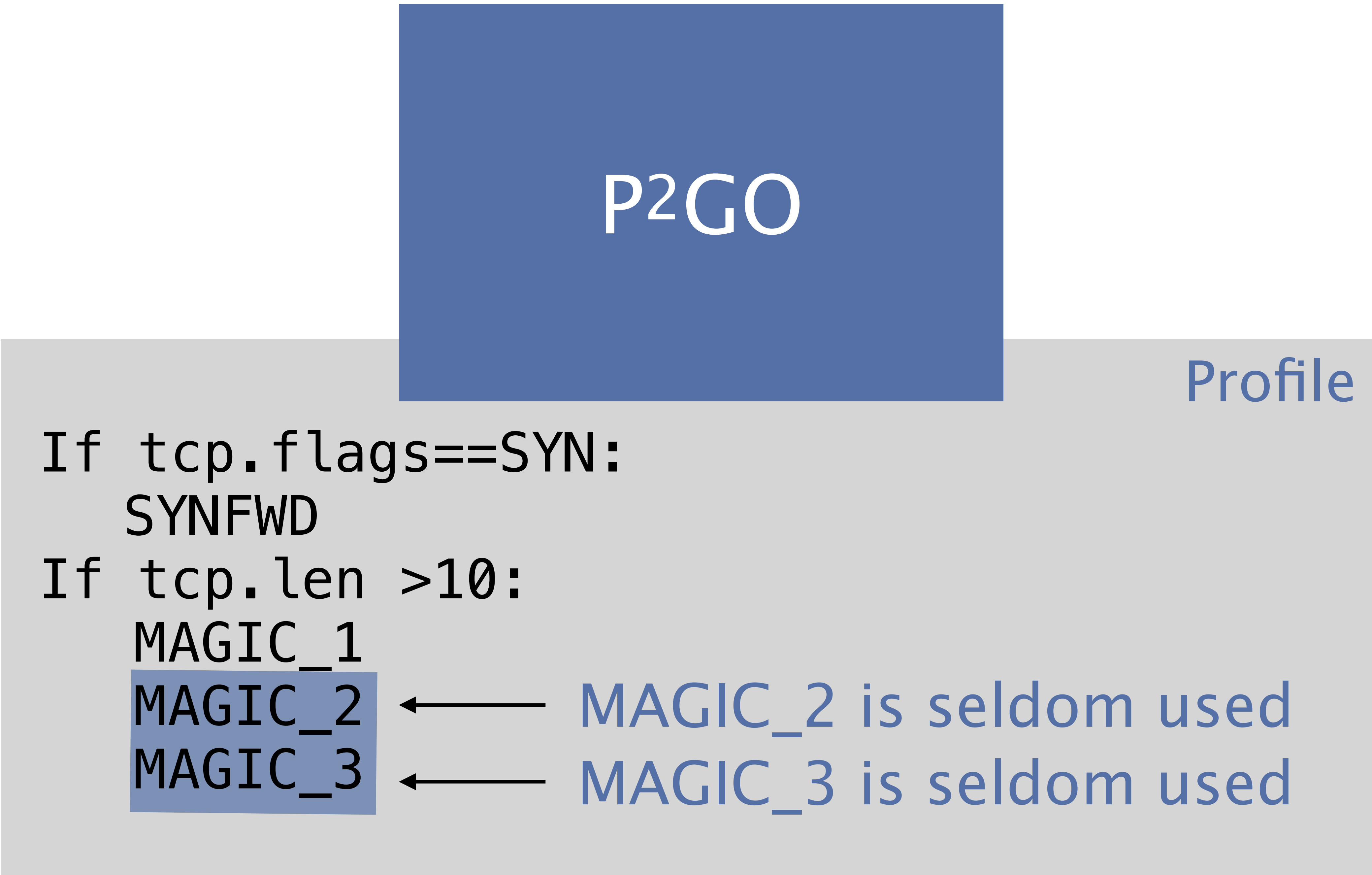
remove fake dependencies

Reduce resource waste

reduce memory usage

Improve hardware–software split

P^{2}GO fetched the most seldom used tables from the profile

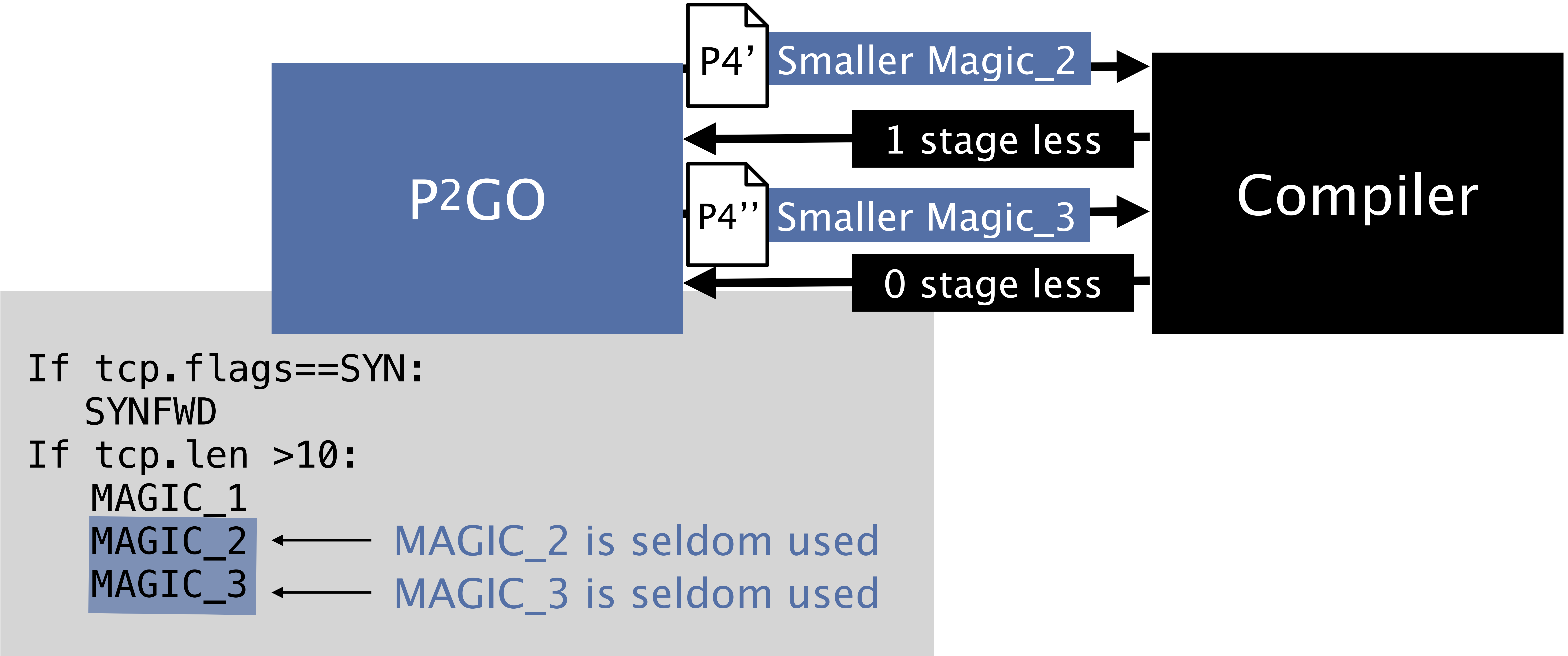


```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_2 is seldom used

MAGIC_3 is seldom used

P^{2}GO generates programs with reduced memory and resubmits them

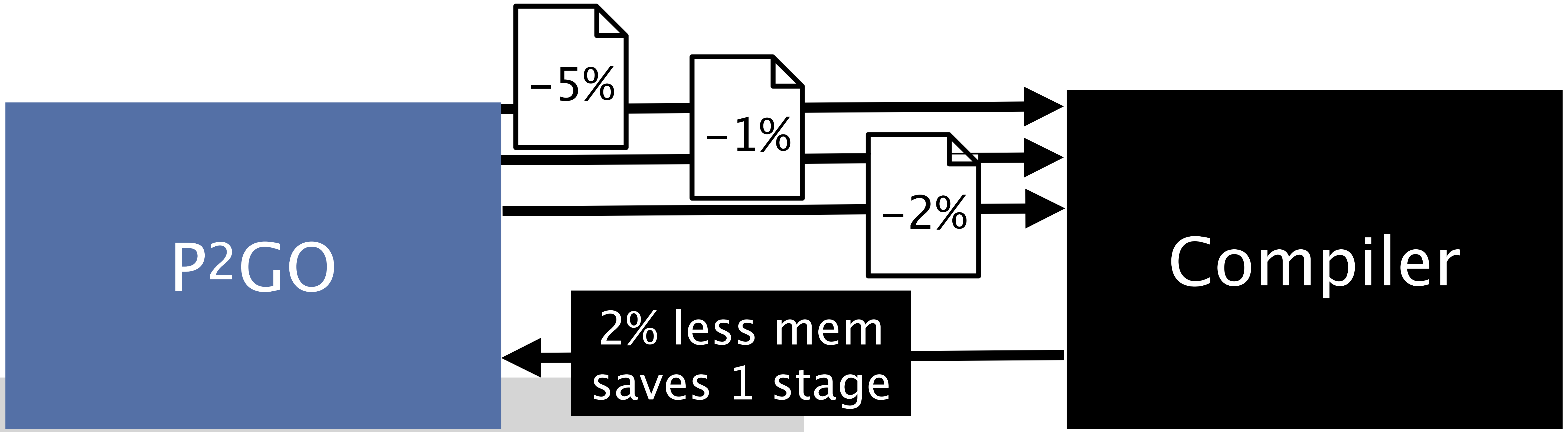


```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_2 is seldom used

MAGIC_3 is seldom used

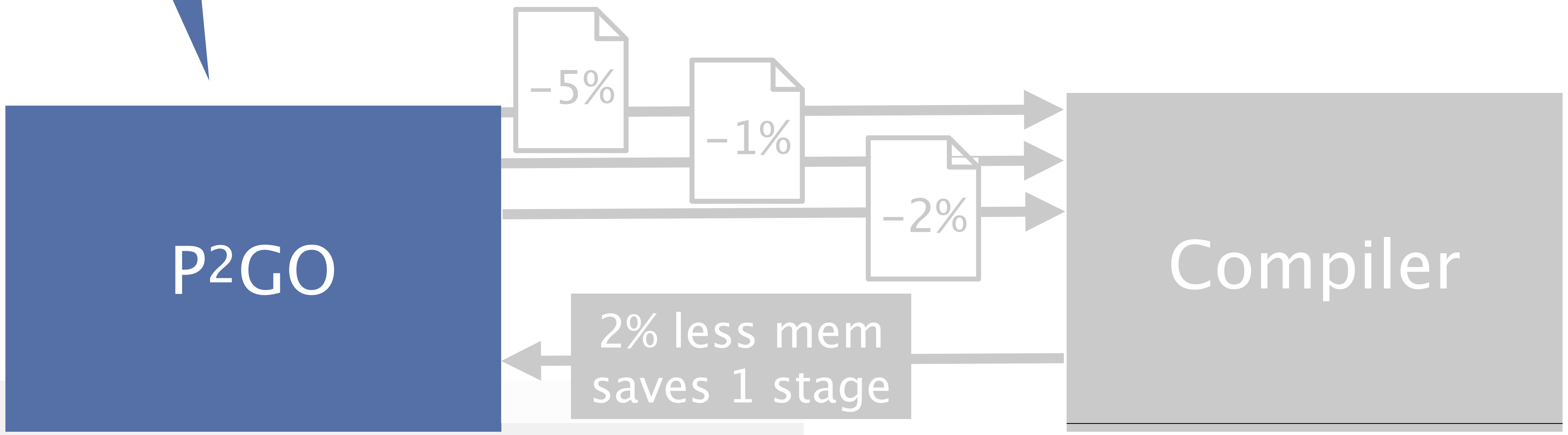
P^{2}GO repeats the process to find the minimum change to save a stage



```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2   ←——— MAGIC_2 is seldom used
   MAGIC_3   ←——— MAGIC_3 is seldom used
```

P^{2}GO asks the programmer whether he would accept the reduction

With 2% less memory for MAGIC_2
you can save a stage.

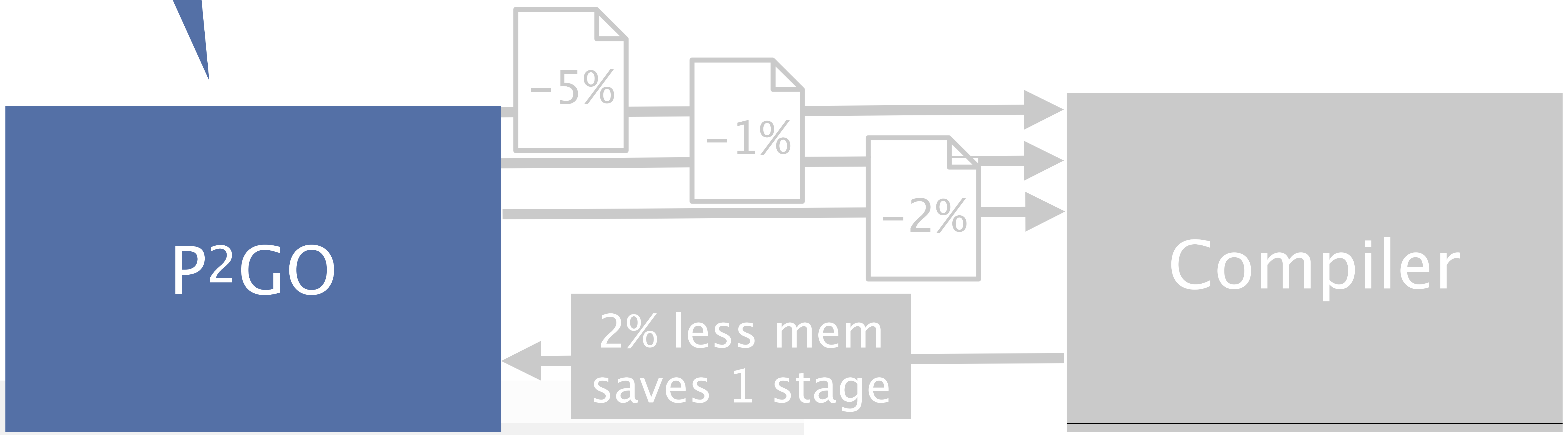


```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2  ←——  MAGIC_2 is seldom used
   MAGIC_3  ←——  MAGIC_3 is seldom used
```

The programmers considers and accepts the change

With 2% less memory for MAGIC_2
you can save a stage.

Ah sure that was a rough
estimate anyway.



```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2  ←——  MAGIC_2 is seldom used
   MAGIC_3  ←——  MAGIC_3 is seldom used
```

P^{2}GO uses the profile to reduce the number of stages, while not changing the program's semantic

Increase pipeline concurrency

remove fake dependencies

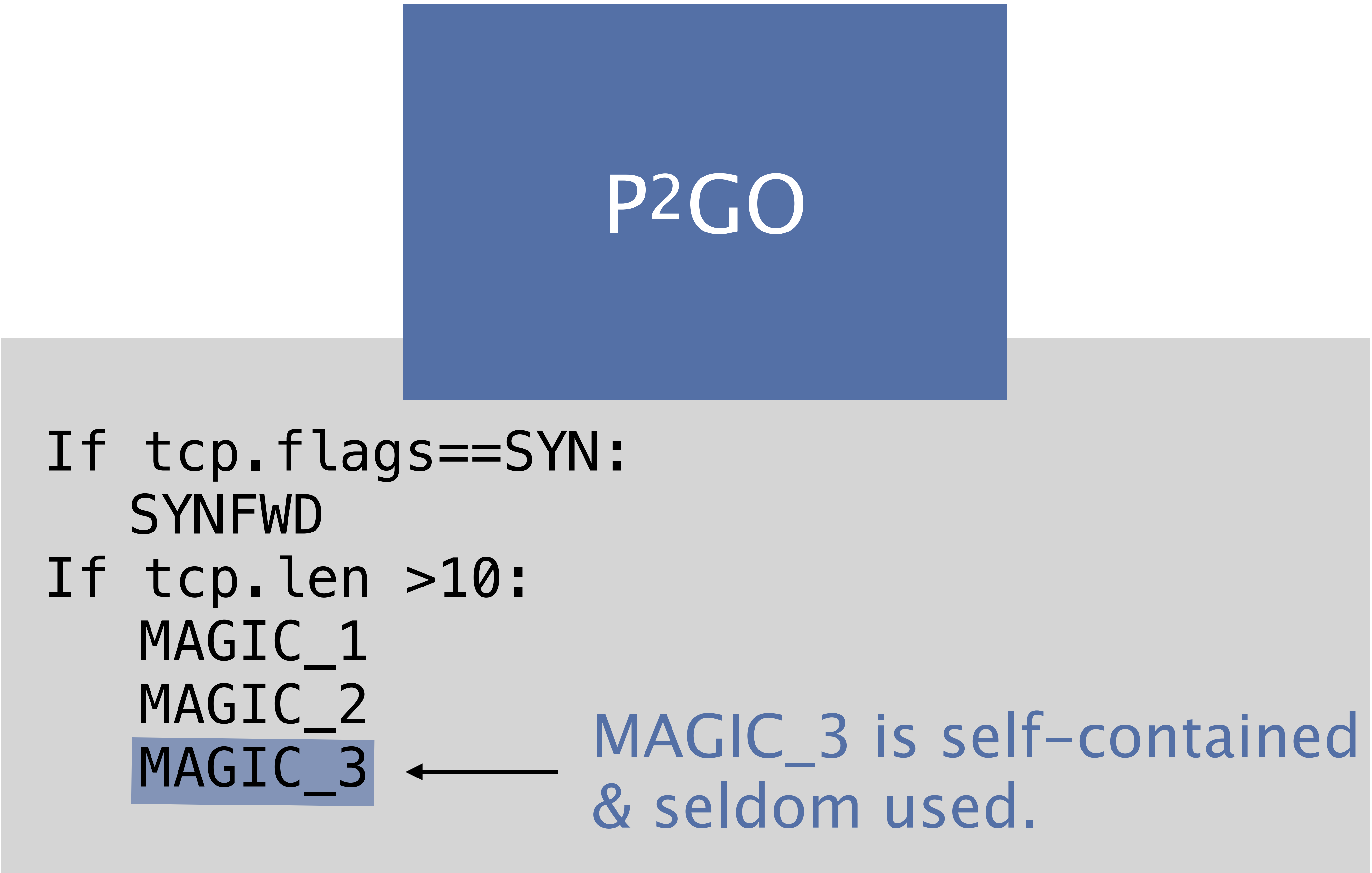
Reduce resource waste

reduce memory usage

Improve hardware–software split

migrate code segments to software

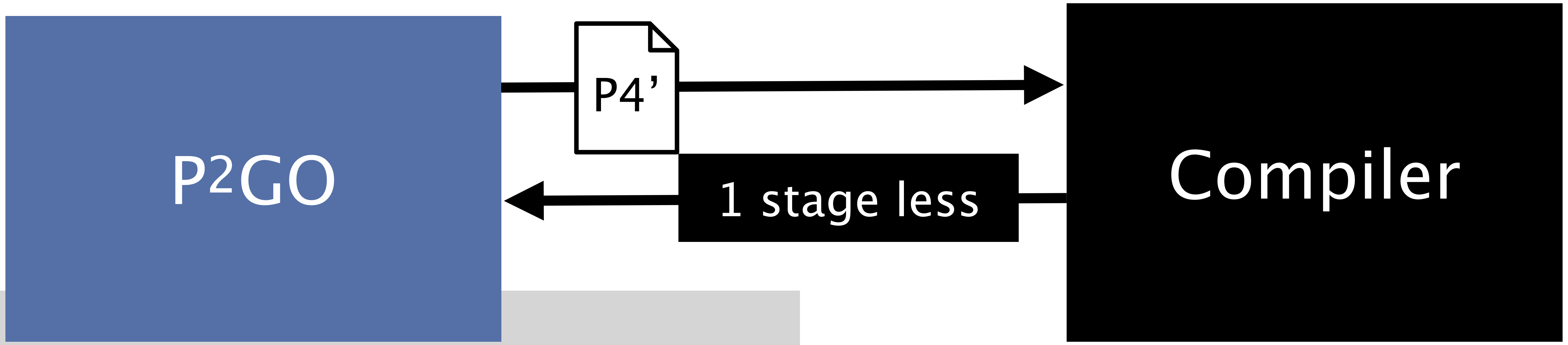
P^{2}GO fetches the least-used self-contained segment



```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_3 is self-contained
& seldom used.

P^{2}GO generates a program without MAGIC_3
that sends the corresponding packets to the controller

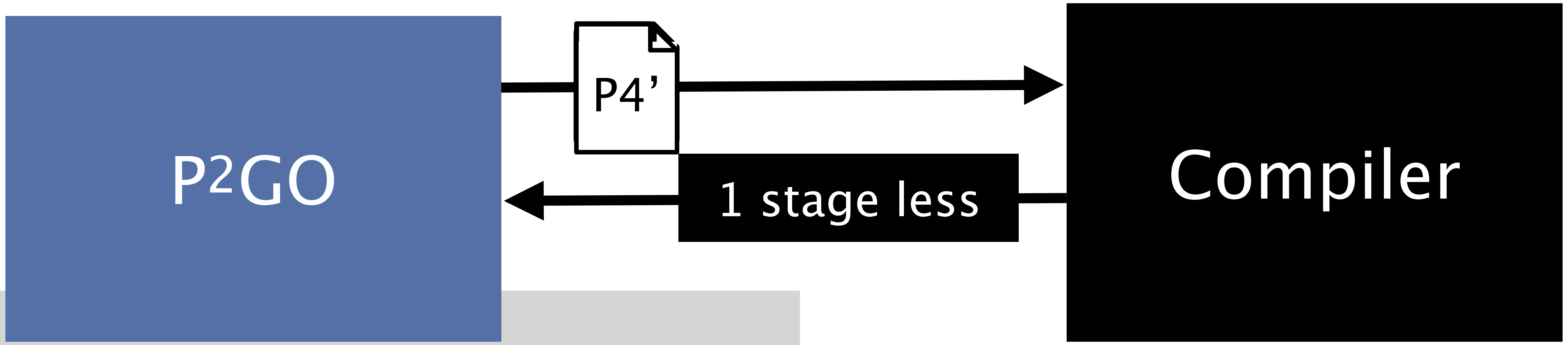


```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_2 is self-contained
& seldom used.

P^{2}GO generates a program without MAGIC_3
that sends the corresponding packets to the controller

MAGIC_3 is seldom used and self-contained,
by moving it to software you can save a stage.



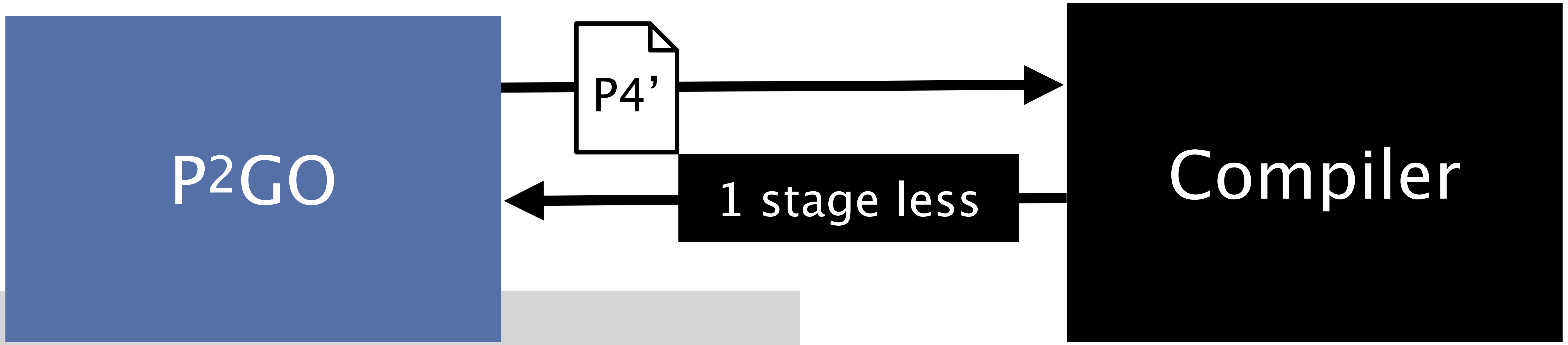
```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_3 is self-contained
& seldom used.

P^{2}GO generates a program without MAGIC_3 that sends the corresponding packets to the controller

MAGIC_3 is seldom used and self-contained,
By moving it to software you can save a stage.

Ah no! MAGIC_3 will stay in the data plane. The traffic trace used for profiling happened to not contain such traffic.



```
If tcp.flags==SYN:
   SYNFWD
If tcp.len >10:
   MAGIC_1
   MAGIC_2
   MAGIC_3
```

MAGIC_3 is self-contained
& seldom used.

P^{2}GO uses the profile to reduce the number of stages, while not changing the program's semantic

Increase pipeline concurrency

remove fake dependencies

Reduce resource waste

reduce memory usage

Improve hardware–software split

migrate code segments to software

P^{2}GO: **P**4 **P**rofile-**G**uided **O**ptimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary evaluation

Open research questions

P^{2}GO working alongside the Tofino compiler reduces the pipeline length of realistic examples

Example	Used Optimization	# Stages Before	# Stages After
NAT & GRE	Removing Dependencies	4	3
Sourceguard	Reducing Memory	5	4
Failure Detection	Offloading Code	4	2

P^2**GO**: **P**4 **P**rofile-**G**uided **O**ptimizations

Profiling a P4 program

Optimization 1: remove fake dependencies

Optimization 2: reduce resource waste

Optimization 3: improve hardware-software split

Preliminary Evaluation

Open research questions

Open research questions

Representative traffic trace

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

Open research questions

Representative traffic trace

How can we find a representative traffic trace? Would the problem be solved with online profiling?

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

Open research questions

Representative traffic trace

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

Can we ensure that the optimized program is semantically equivalent without involving the programmer?

Open research questions

Representative traffic trace

Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

How to detect and mitigate inaccuracies of the profile or of the programmer?

Open research questions

Representative traffic trace

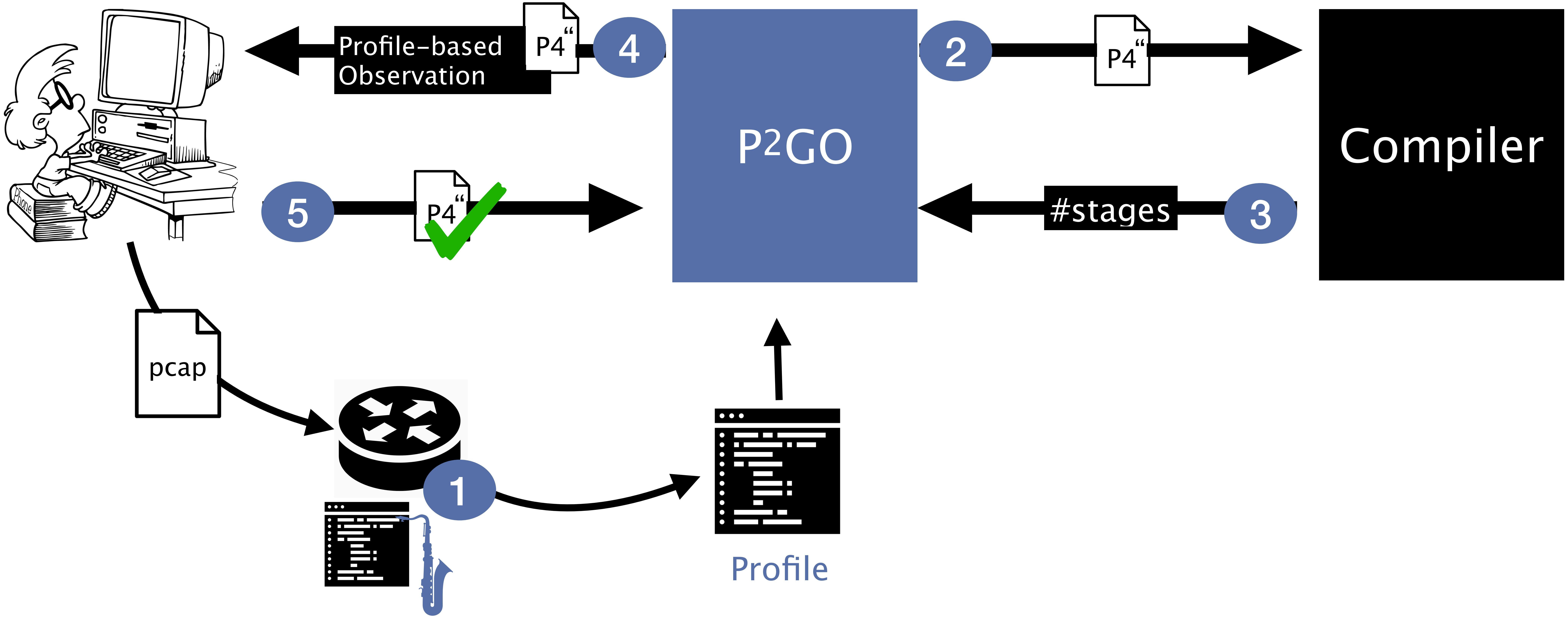
Semantic equivalence

Mis-speculation

Optimize across multiple dimensions

What if a program does not compile due to other resources ? How to optimize the program across multiple dimensions?

P²GO P4 Profile-Guided Optimizations


Profile-based
Observation
P4“
4
2
P4“
P²GO
Compiler
5
P4“
#stages
3
pcap
1
Profile