CoRD: Collaborative Data Race Detection

Baris Kasikci, Cristian Zamfir, and George Candea
School of Computer & Communication Sciences
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Data Races

Data Races

- Accesses to shared memory location

Data Races

- Accesses to shared memory location
  - *By multiple threads*

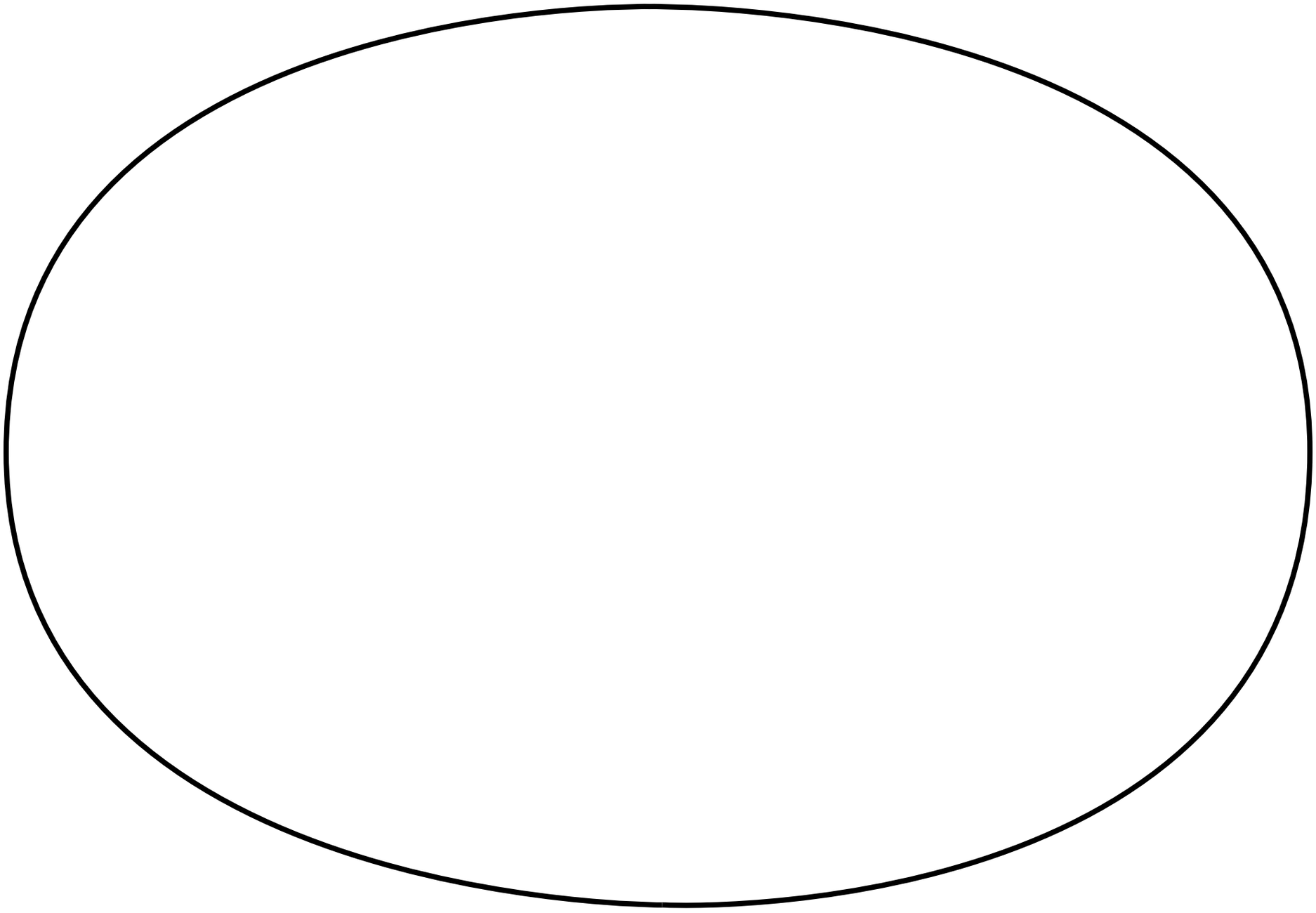
Data Races

- Accesses to shared memory location
  - *By multiple threads*
  - *At least one of the accesses is a write*

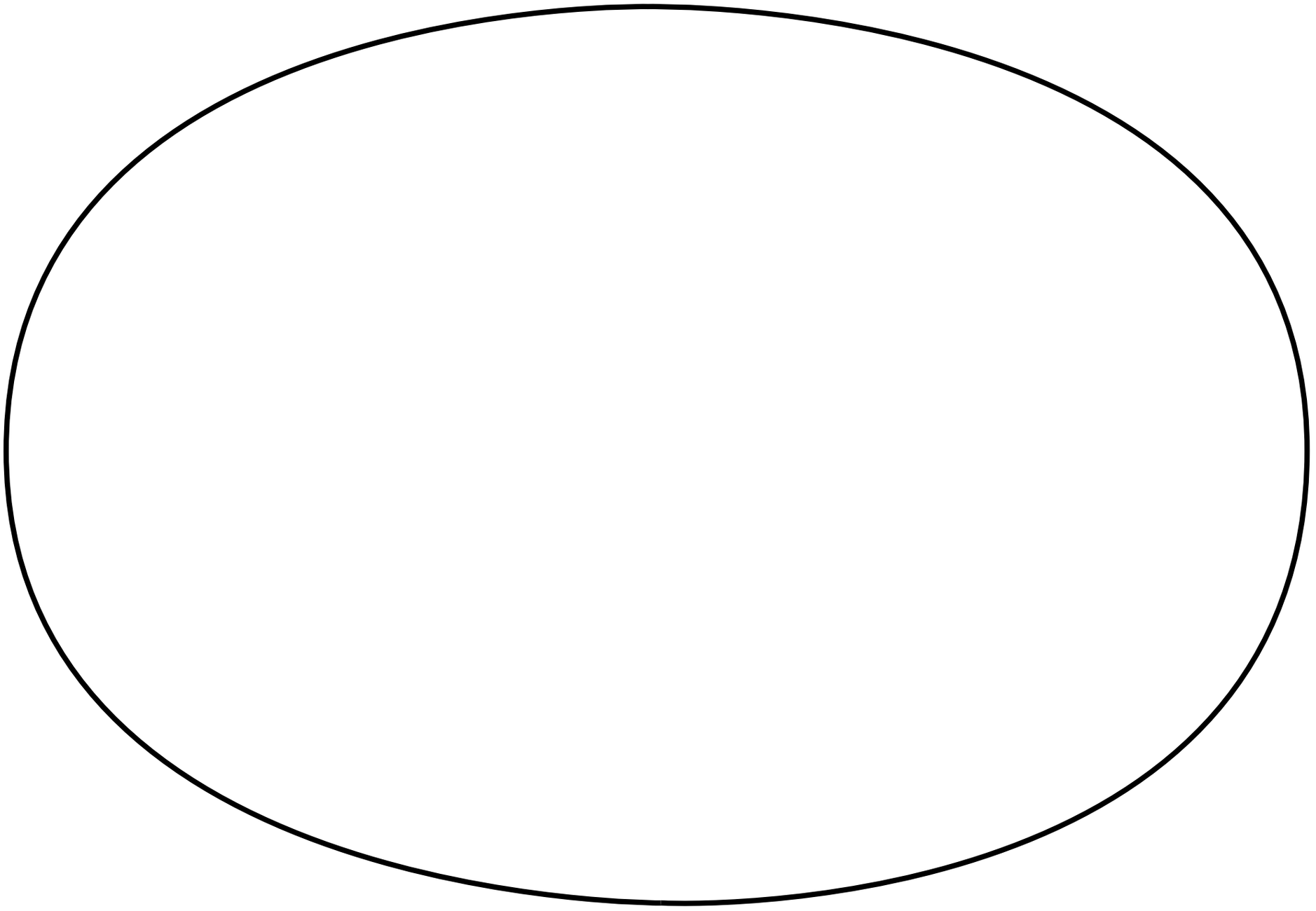
Data Races

- Accesses to shared memory location
  - *By multiple threads*
  - *At least one of the accesses is a write*
  - *The accesses can happen simultaneously*

Data Races



Data Races



Races are numerous in modern software

Data Races

→ 1000 Races

Races are numerous in modern software

Data races

C/C++

POSIX

C/C++

POSIX

Compilers can arbitrarily break racy programs

How to Ensure Software is Race-free?

How to Ensure Software is Race-free?

- Static race detectors
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Many false positives* ✘

How to Ensure Software is Race-free?

- Static race detectors
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Many false positives* ✘
- Dynamic race detectors
  - *Per-run analysis* ✘
  - *Slow* ✘
  - *Many false negatives* ✘
  - *Few false positives* ✔

How to Ensure Software is Race-free?

- Static race detectors
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Many false positives* ✘
- Dynamic race detectors
  - *Per-run analysis* ✘
  - *Slow* ✘
  - *Many false negatives* ✘
  - *Few false positives* ✔

Existing detectors have important limitations

How to Ensure Software is Race-free?

- Static race detectors
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
- Dynamic race detectors
  - *Few false positives* ✔

CoRD

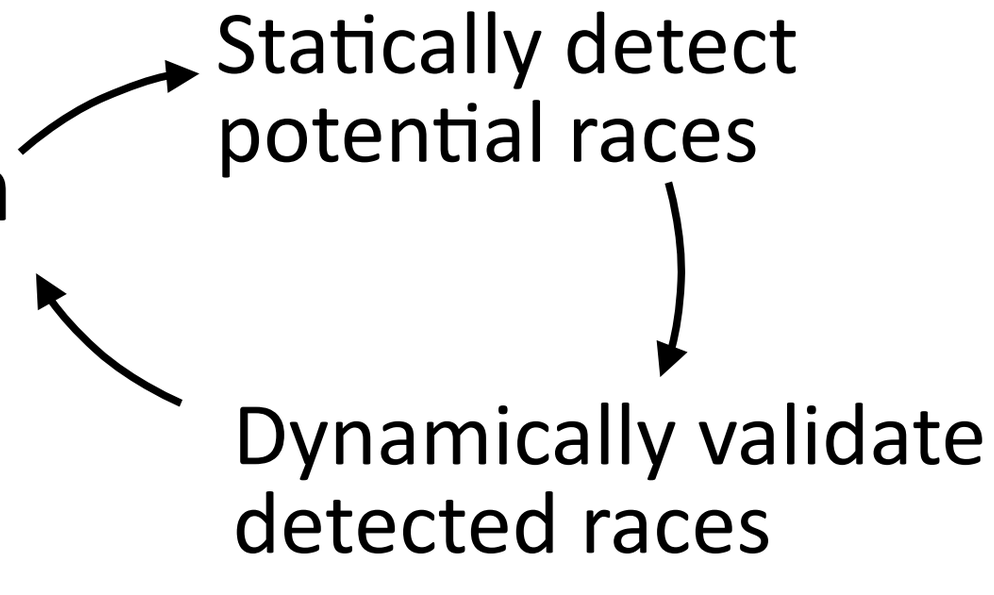
- Collaborative race detection
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Few false positives* ✔

CoRD

- Collaborative race detection → Statically detect potential races
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Few false positives* ✔

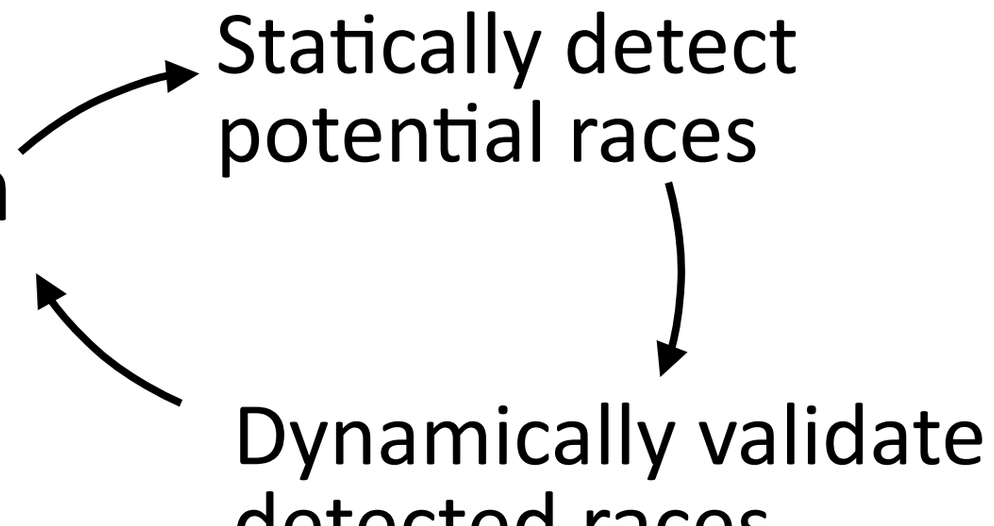
CoRD

- Collaborative race detection
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Few false positives* ✔



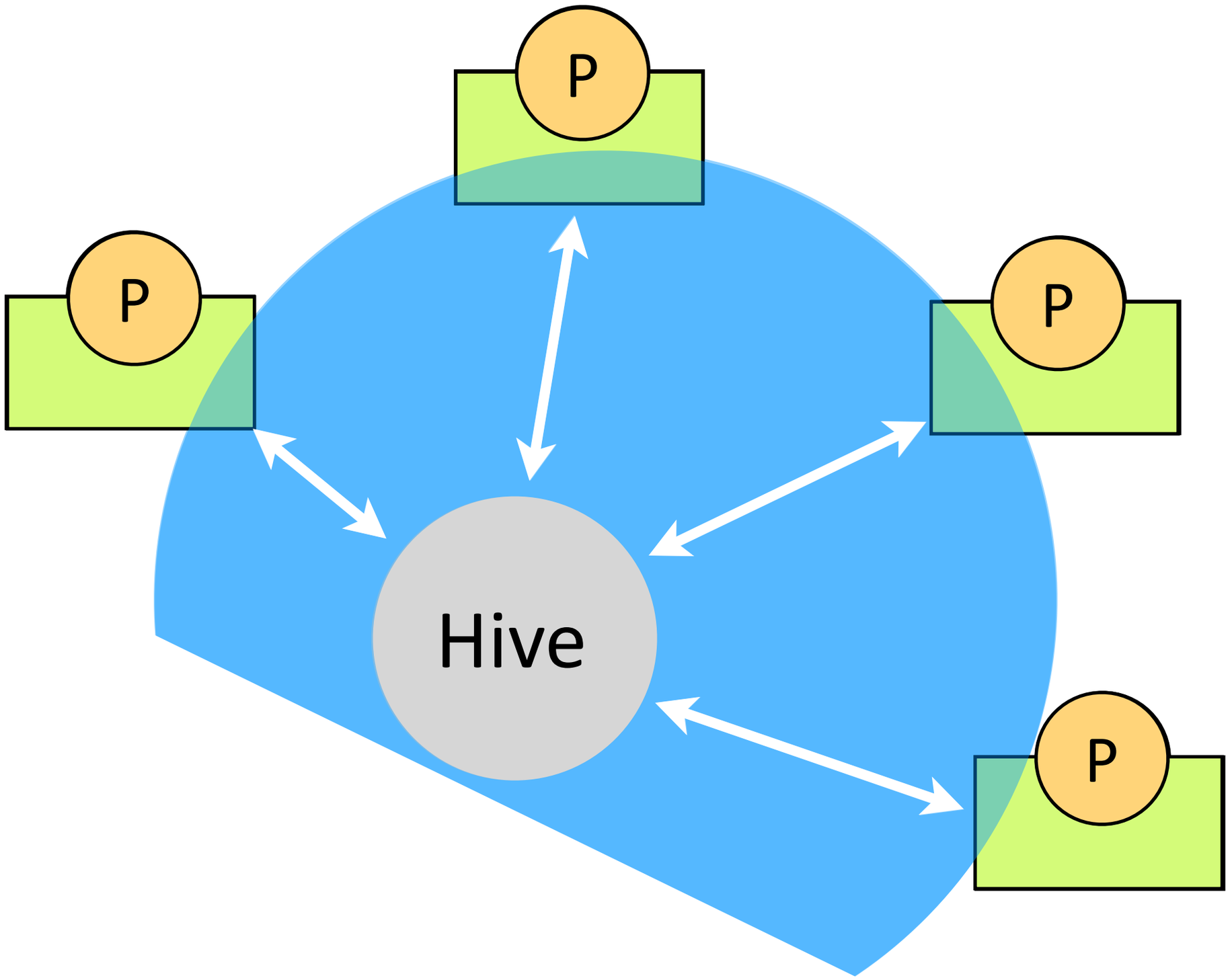
CoRD

- Collaborative race detection
  - *Full path analysis* ✔
  - *Fast* ✔
  - *Few false negatives* ✔
  - *Few false positives* ✔

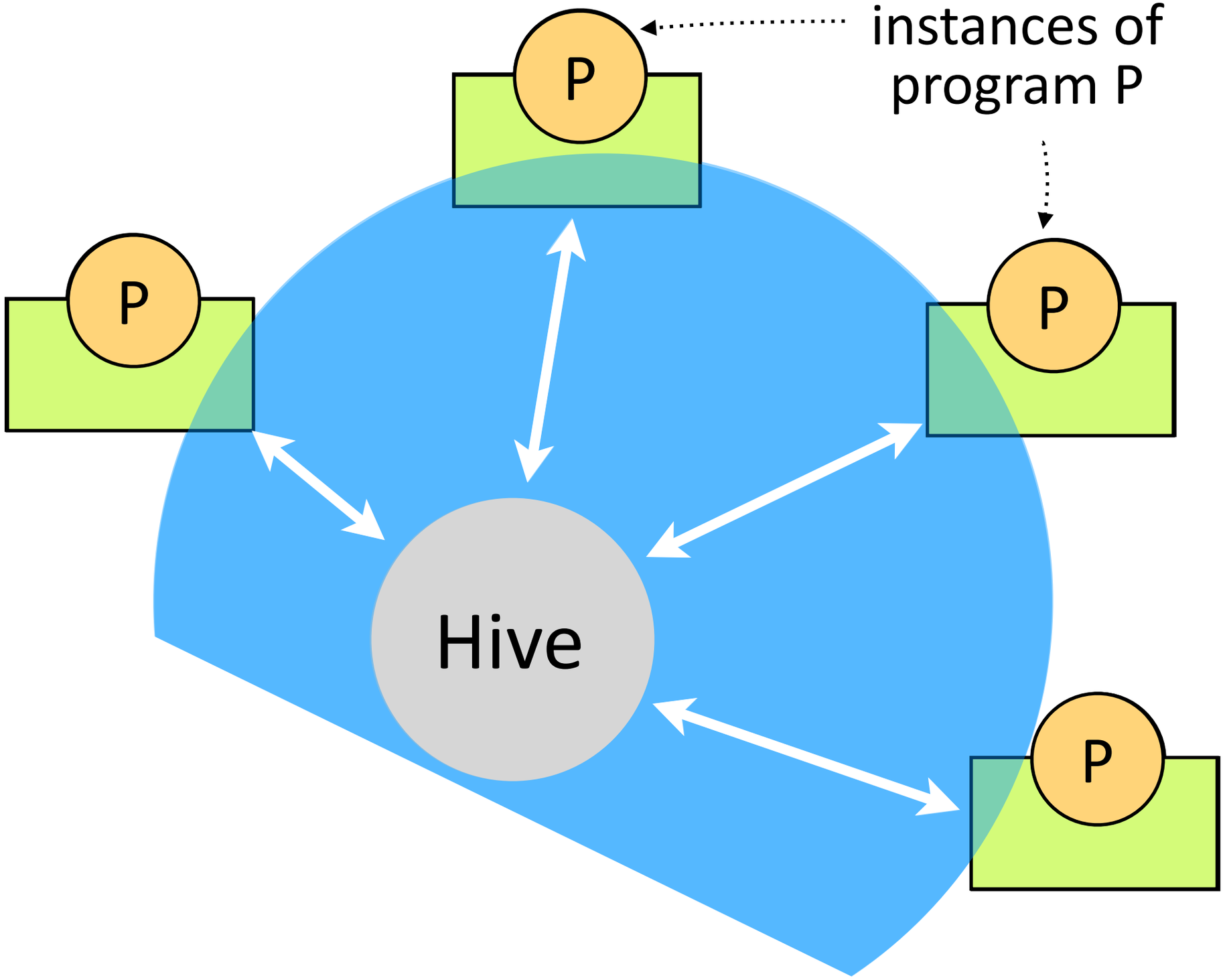


Effectively detected 8 real races in two real programs with 1% overhead

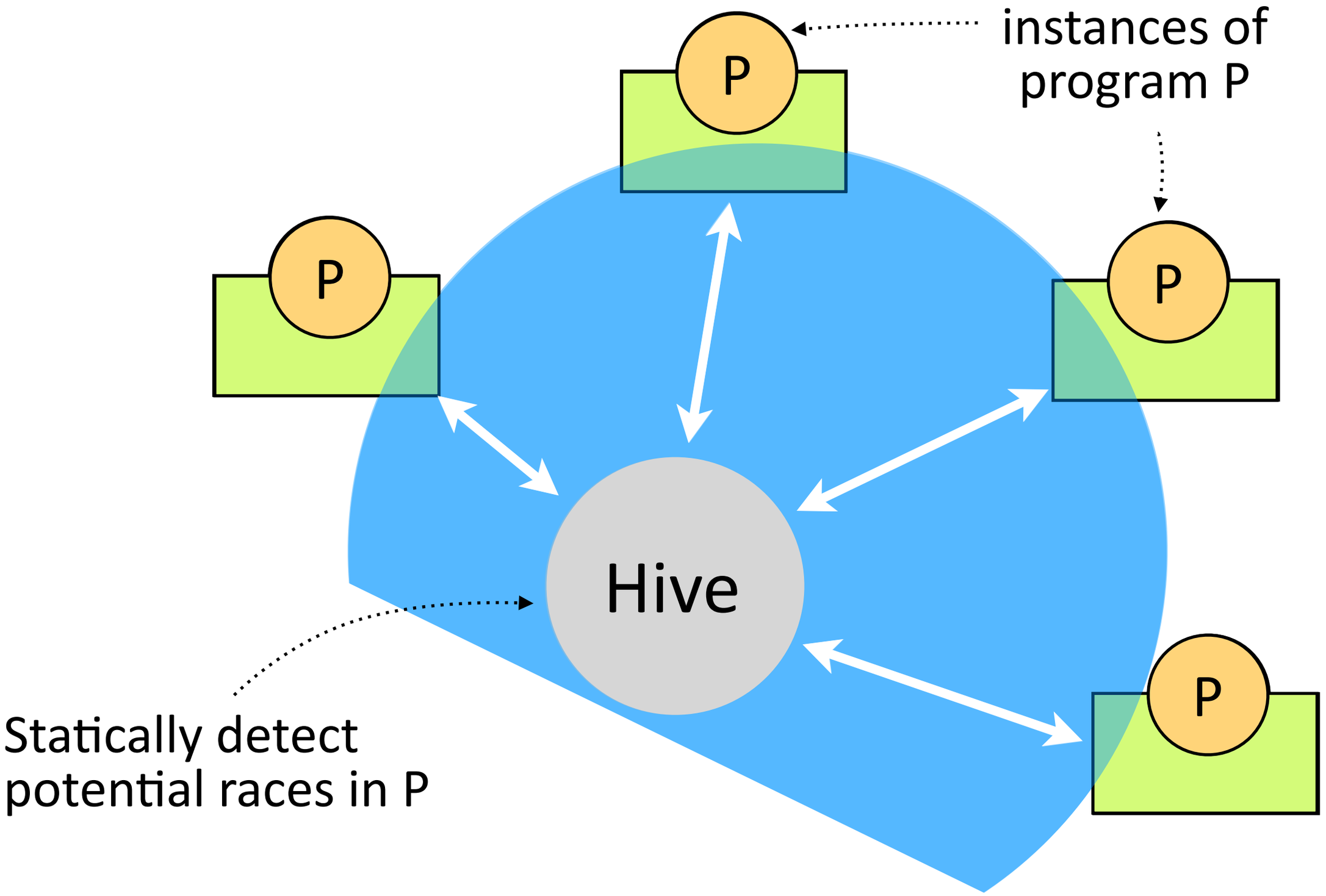
CoRD Architecture



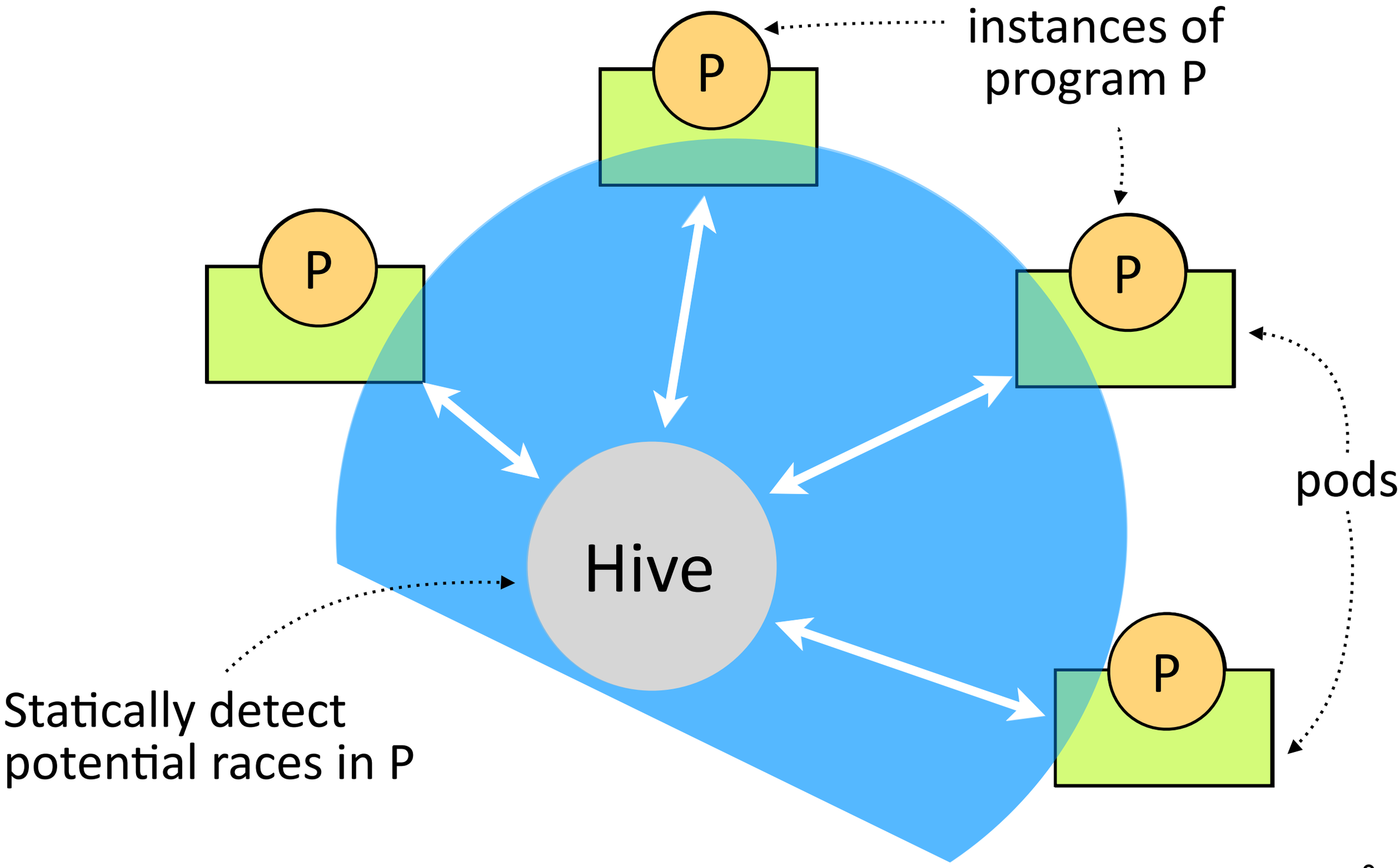
CoRD Architecture



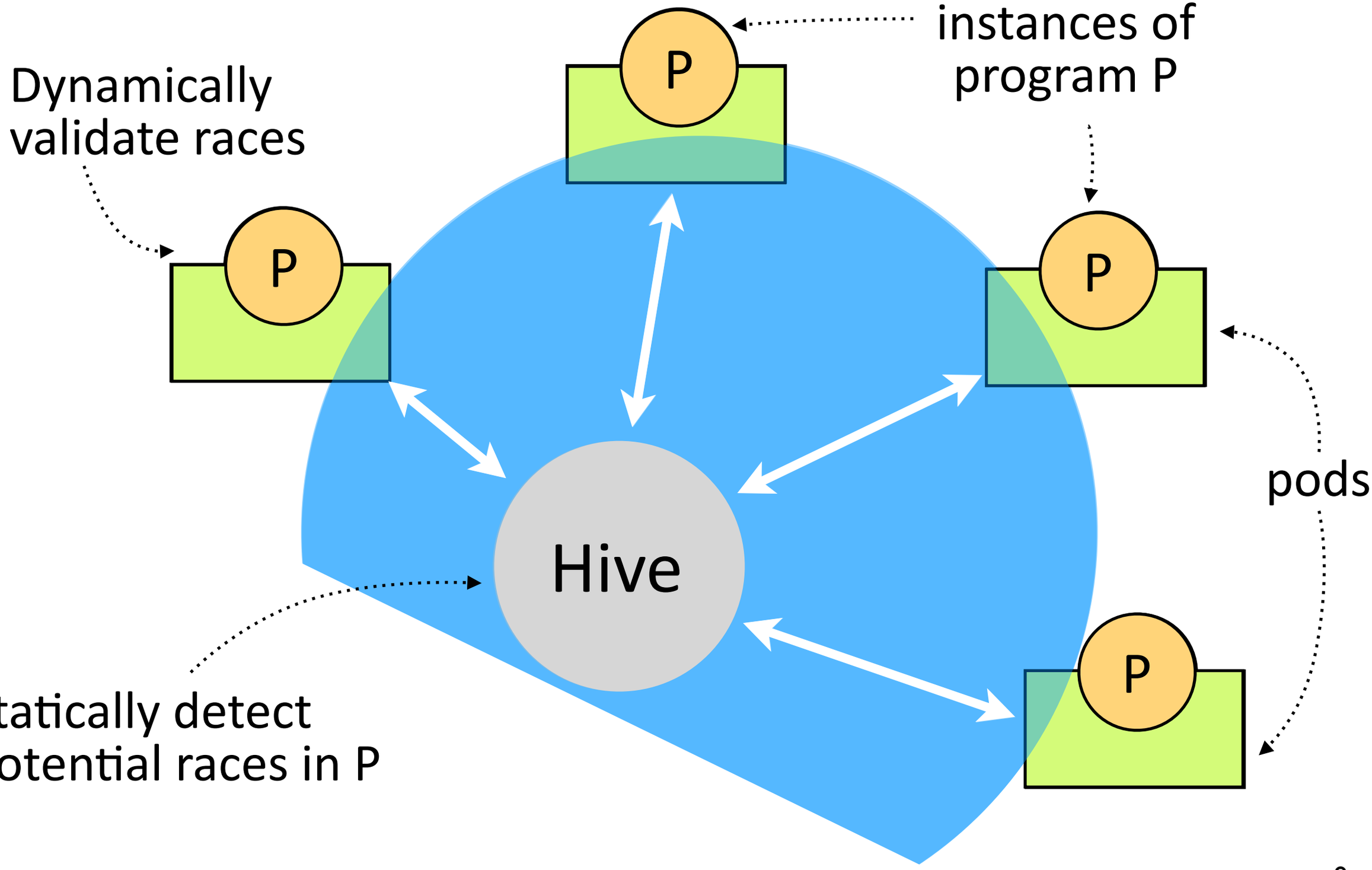
CoRD Architecture

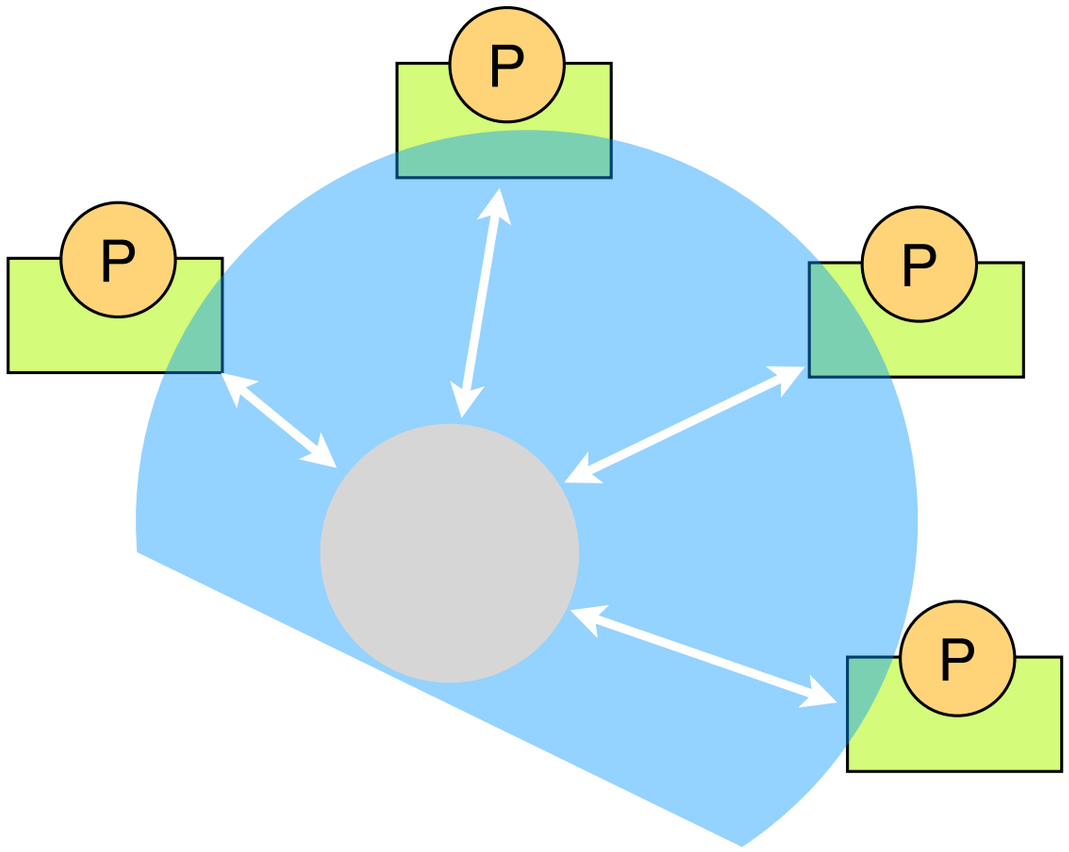


CoRD Architecture



CoRD Architecture

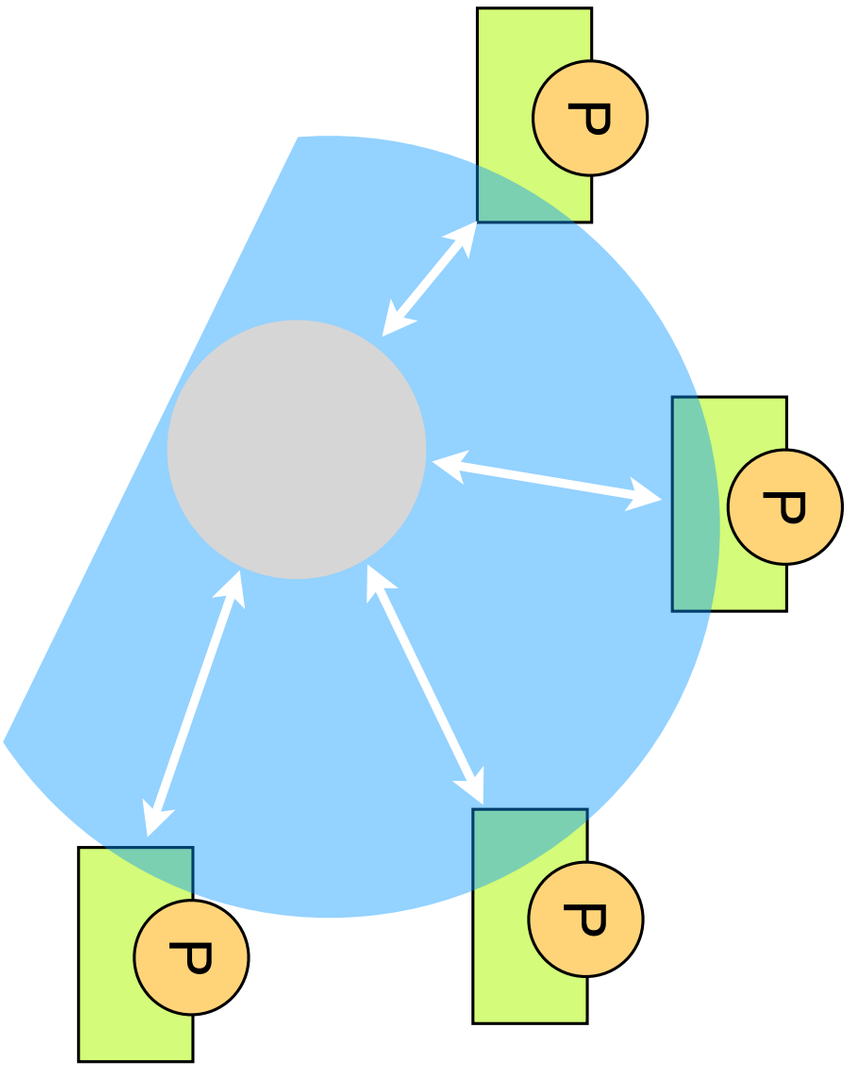




[1] Google chrome blog. http://chrome.blogspot.ch/2012_06_01_archive.html
[2] https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012
[3] R. Cozza et al, Market Share: Mobile Devices by Region and Country, Gartner, Feb 2012
[4] http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/

computers running Chrome[1]

300 Million



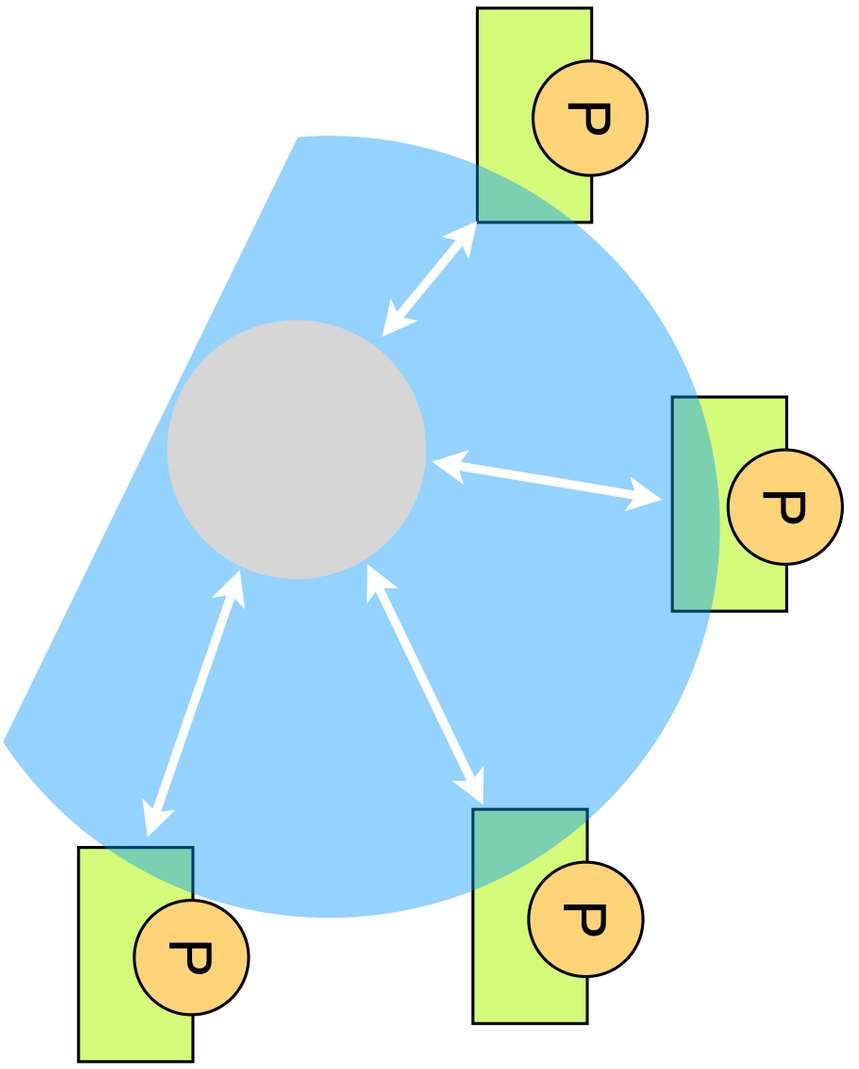
[1] *Google chrome blog. http://chrome.blogspot. ch/2012_06_01_archive.html*
[2] *https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012*
[3] *R. Cozza et al, Market Share: Mobile Devices by Region and Country, Gartner, Feb 2012*
[4] *http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/*

computers running Chrome[1]

300 Million

servers owned by Google[2]

1.8 Million



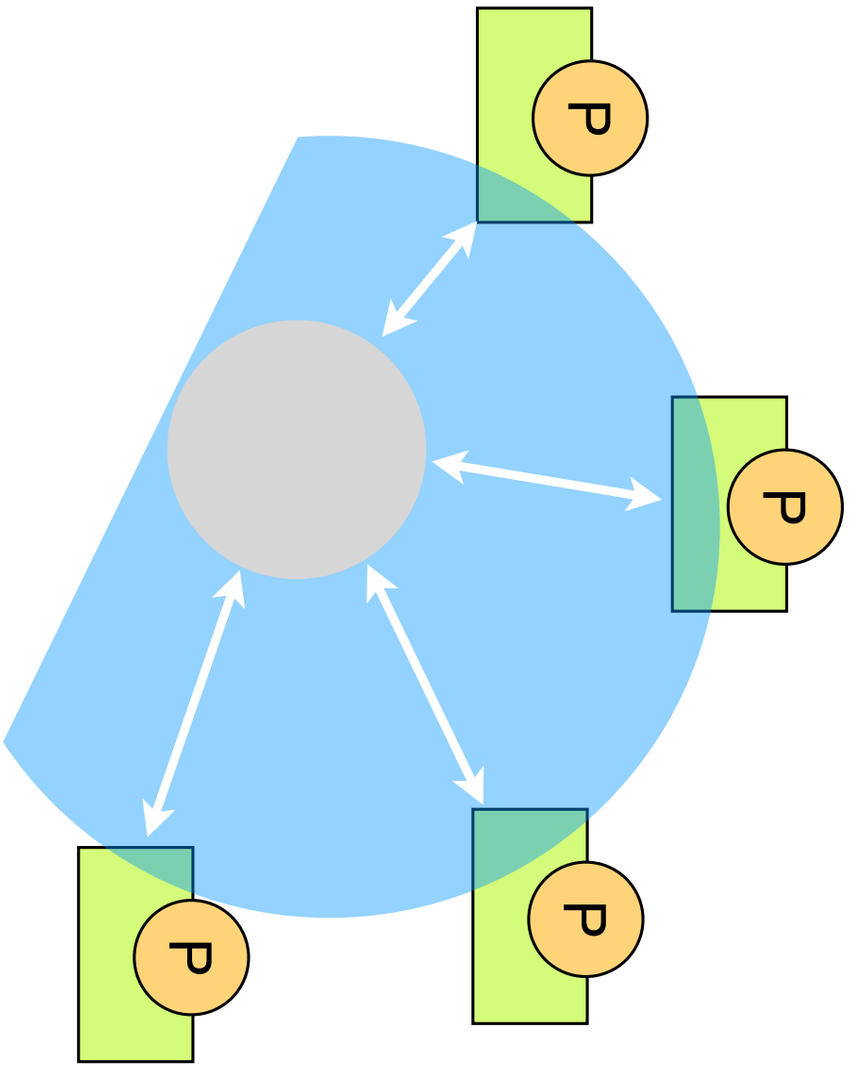
[1] *Google chrome blog. http://chrome.blogspot. ch/2012_06_01_archive.html*
[2] *https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012*
[3] *R. Cozza et al, Market Share: Mobile Devices by Region and Country, Gartner, Feb 2012*
[4] *http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/*

computers running Chrome [1]

300 Million

servers owned by Google [2]

1.8 Million



mobile devices bought in 2011 [3]

1.8 Billion

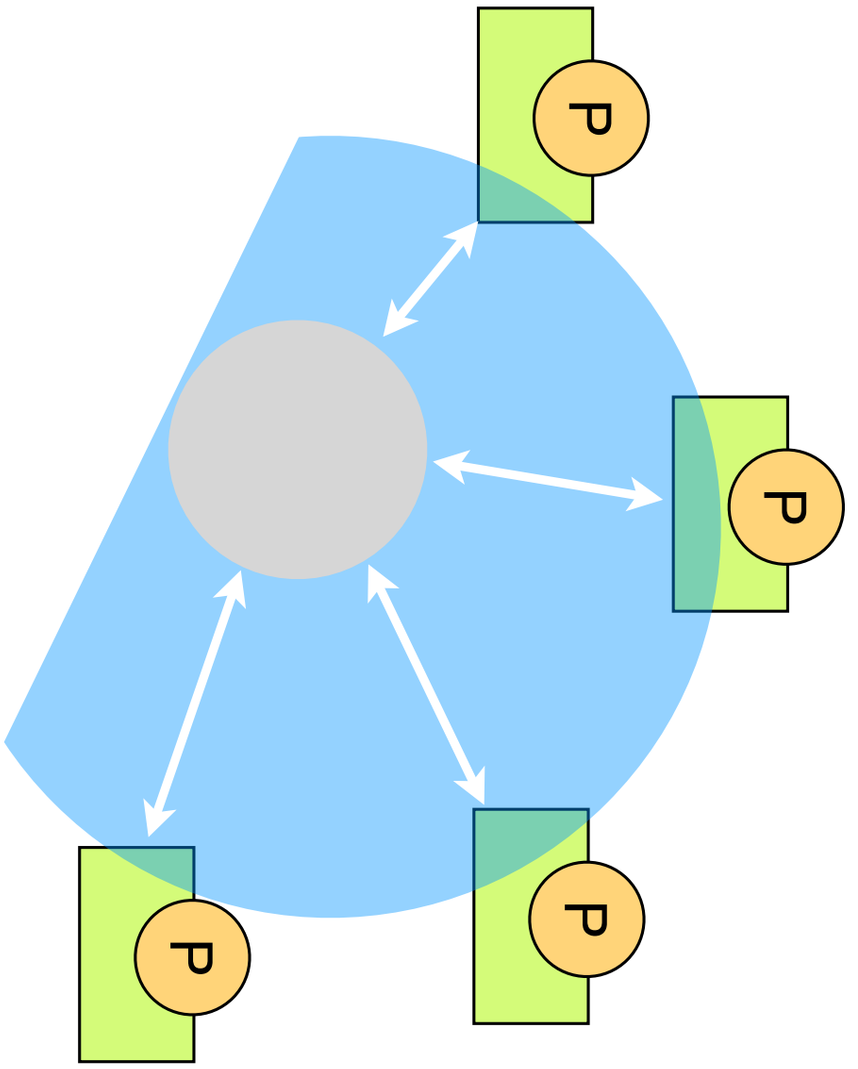
[1] Google chrome blog. http://chrome.blogspot. ch/2012_06_01_archive.html
[2] https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012
[3] R. Cozza et al, Market Share: Mobile Devices by Region and Country, Gartner, Feb 2012
[4] http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/

computers running Chrome[1]

300 Million

servers owned by Google[2]

1.8 Million



Windows 7 beta testers[4]

8 Million

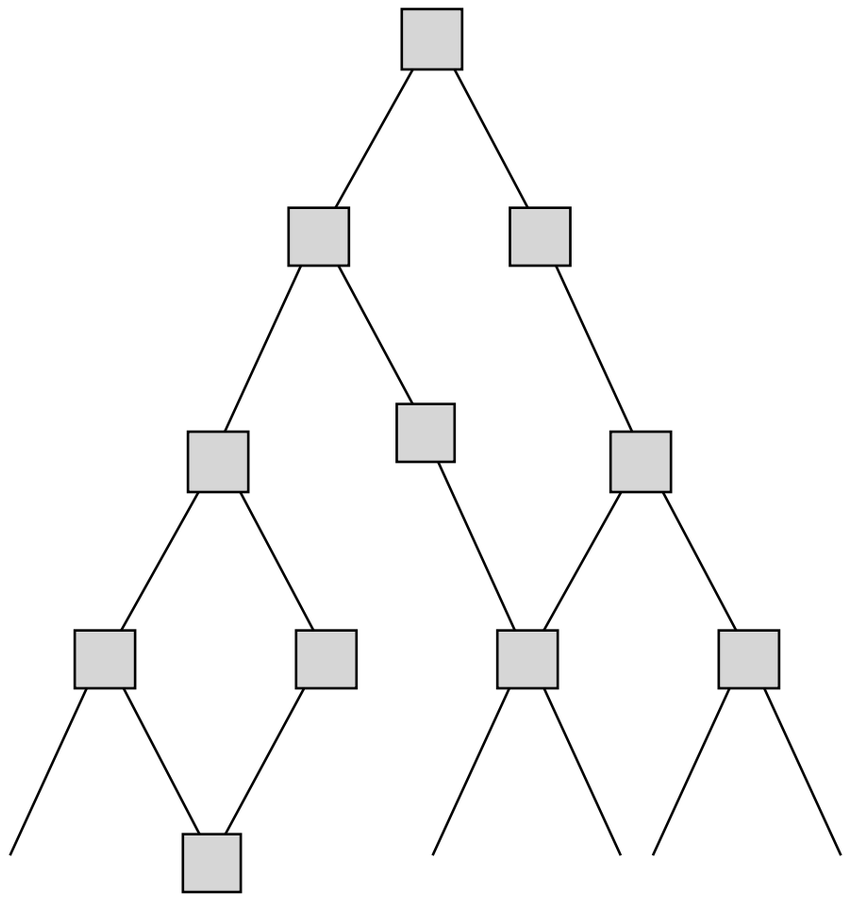
mobile devices bought in 2011[3]

1.8 Billion

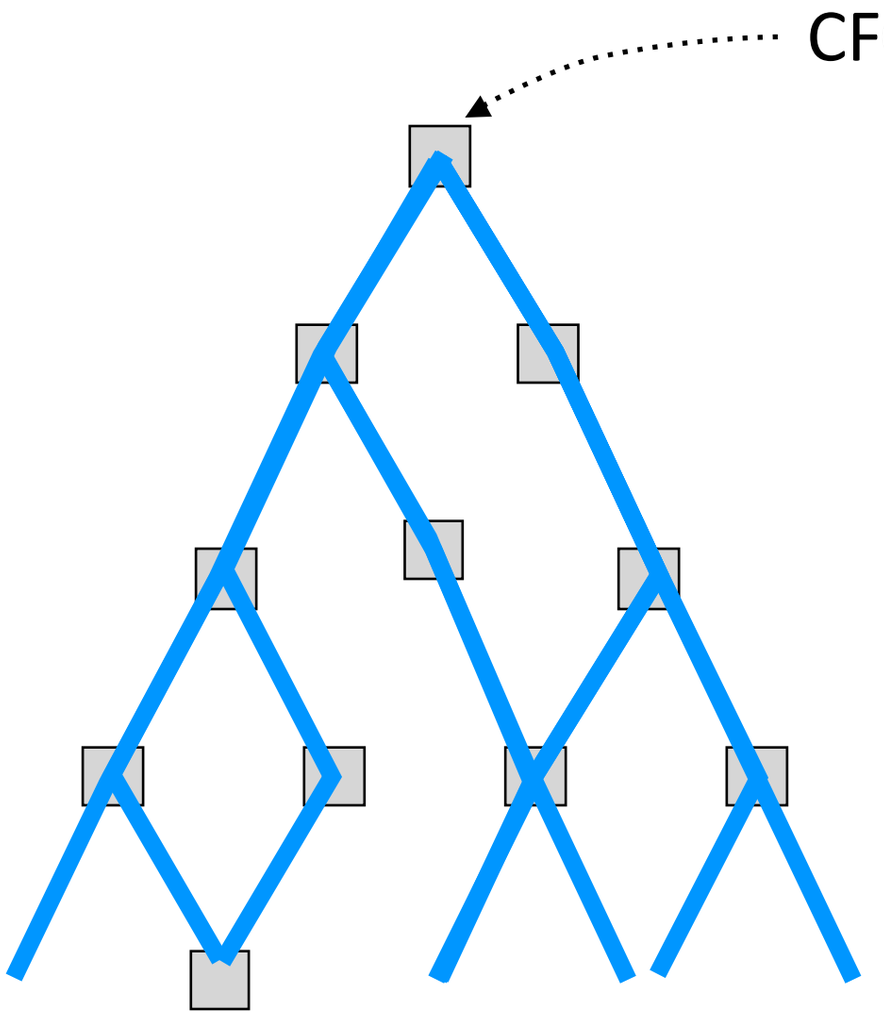
[1] *Google chrome blog. http://chrome.blogspot. ch/2012_06_01_archive.html*
[2] *https://plus.google.com/114250946512808775436/posts/VaQu9sNxJuY, 2012*
[3] *R. Cozza et al, Market Share: Mobile Devices by Region and Country, Gartner, Feb 2012*
[4] *http://arstechnica.com/information-technology/2009/10/windows-7-had-8-million-testers-biggest-beta-ever/*

Static Race Detection

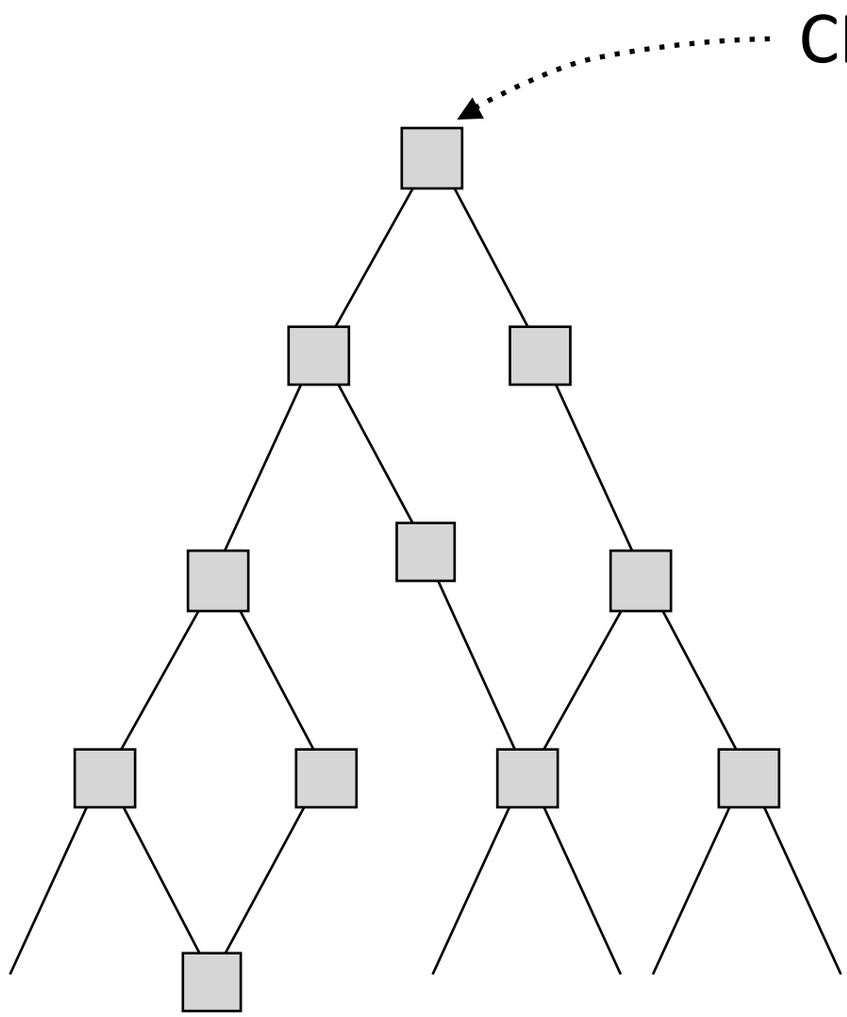
Static Race Detection



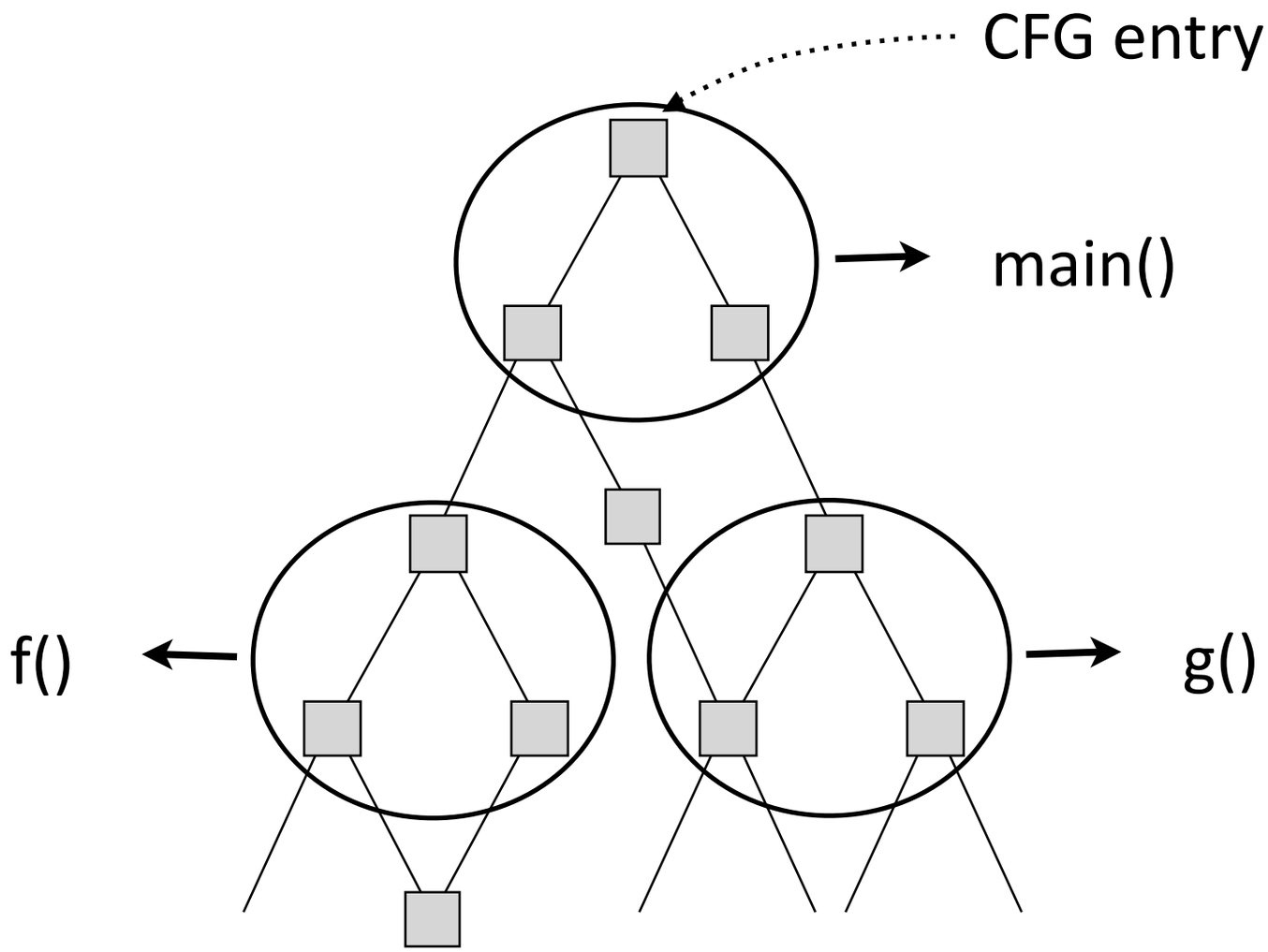
Static Race Detection



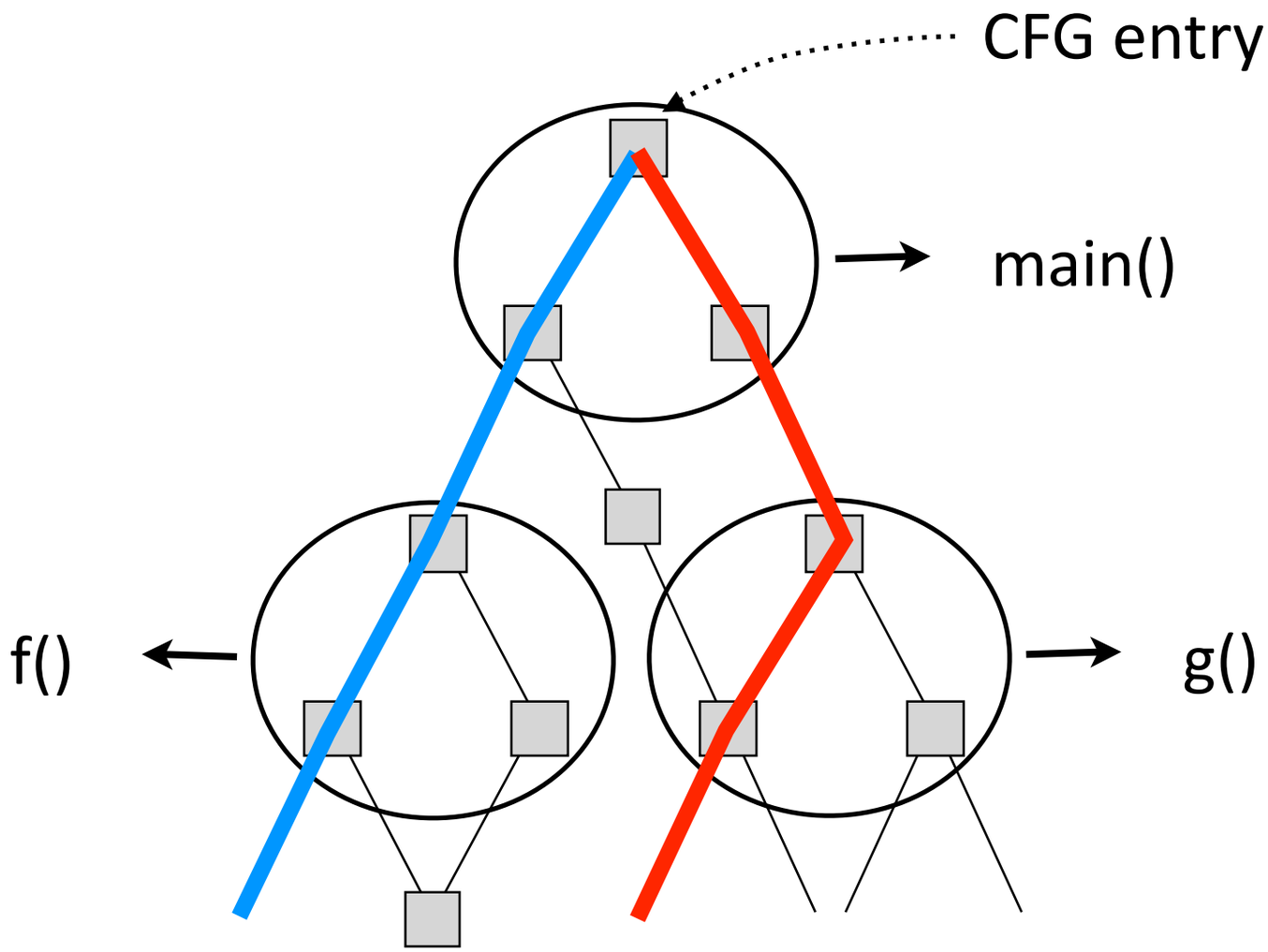
Static Race Detection



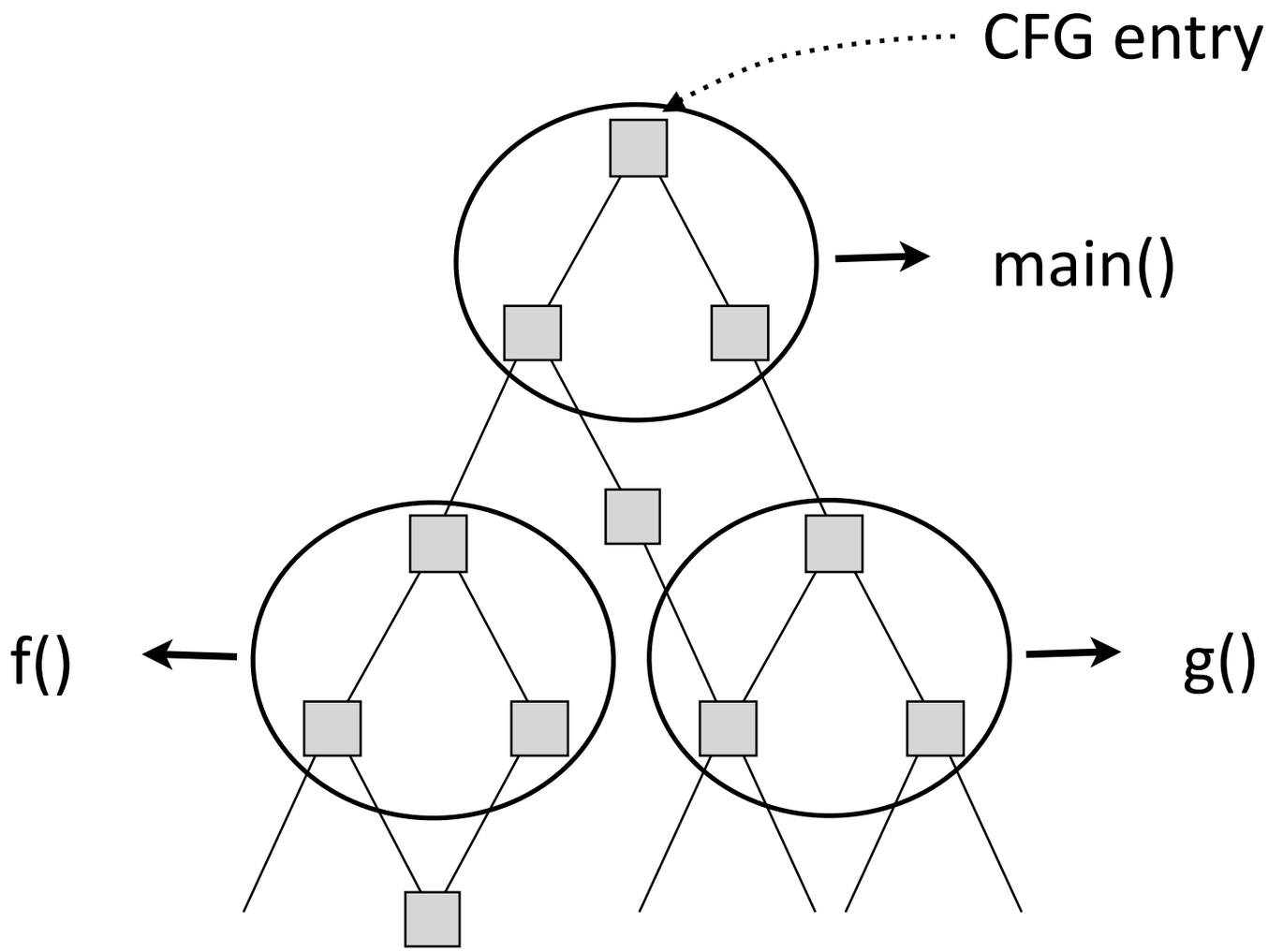
Static Race Detection

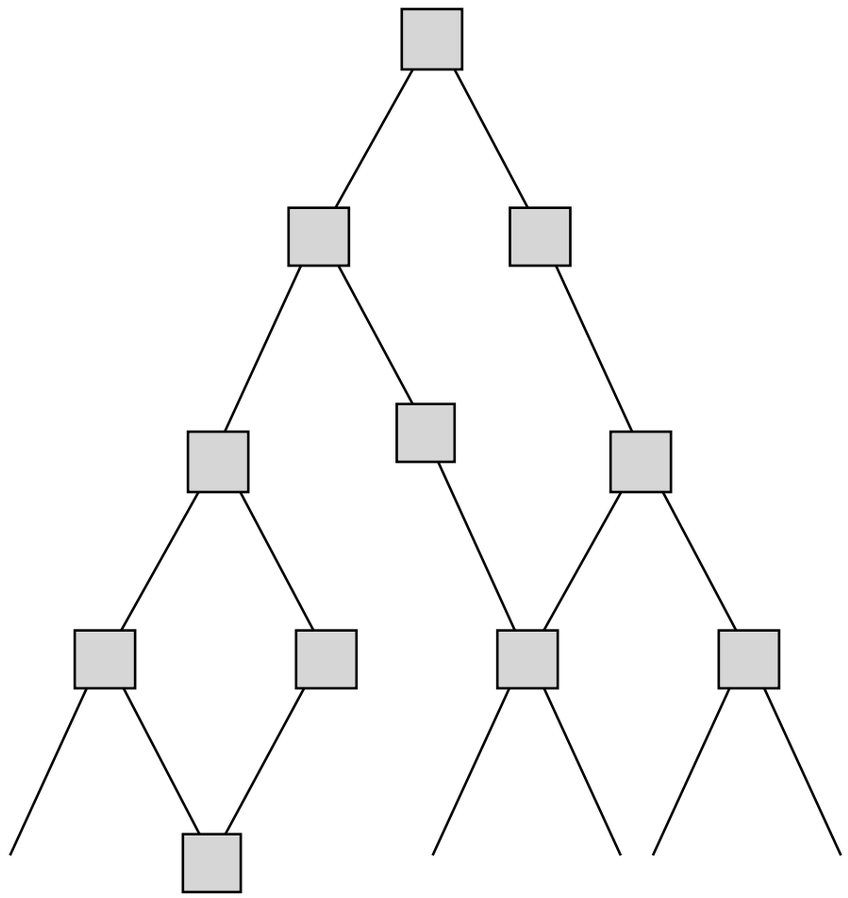


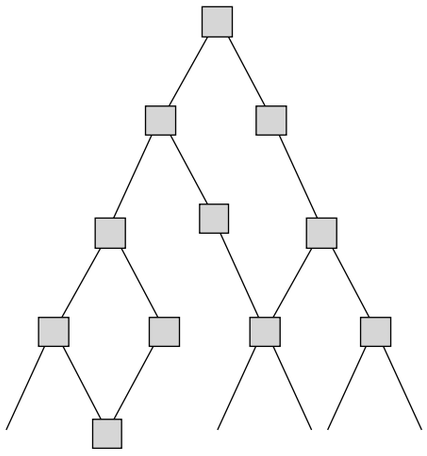
Static Race Detection



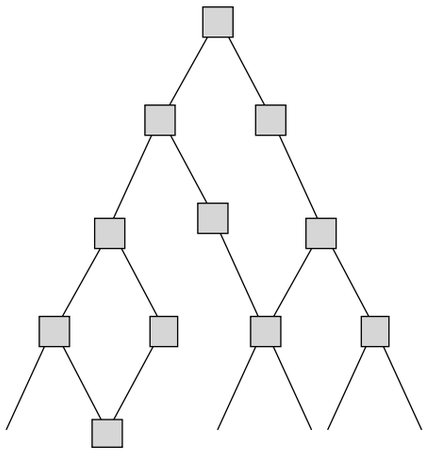
Static Race Detection





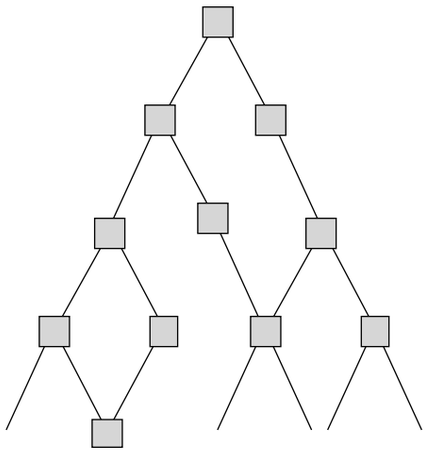


Static Race Detection



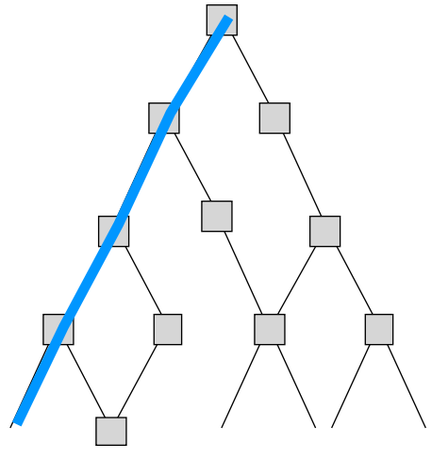
Static Race Detection

```
x = 0
```



Static Race Detection

```
x = 0
```



Path 1

```
lock(l)
x = 1
unlock(l)
```

Static Race Detection

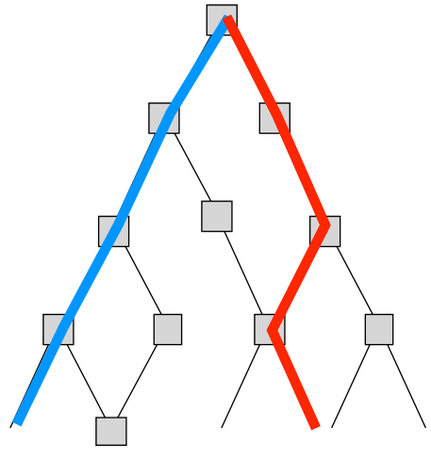
```
x = 0
```

Path 1

```
lock(l)
x = 1
unlock(l)
```

Path 2

```
lock(k)
x = 2
unlock(k)
```



Static Race Detection

```
x = 0
```

Path 1

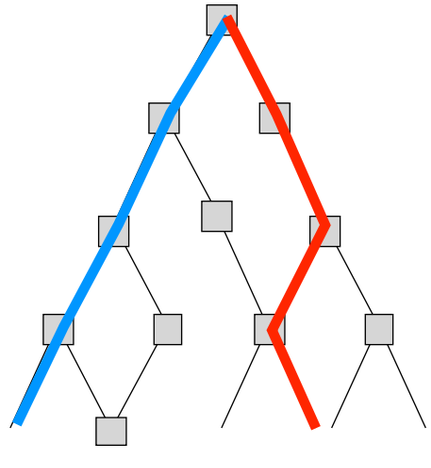
```
lock(l)
x = 1
unlock(l)
```

$LS_1 = \{l\}$

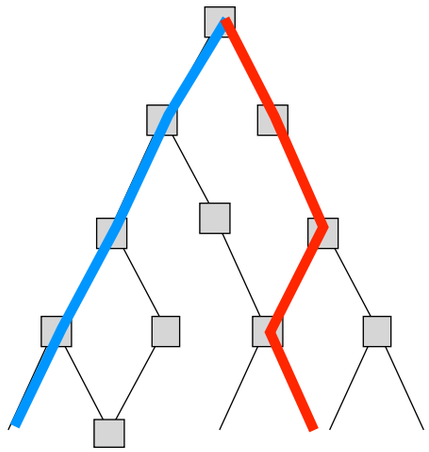
Path 2

```
lock(k)
x = 2
unlock(k)
```

$LS_1 = \{k\}$



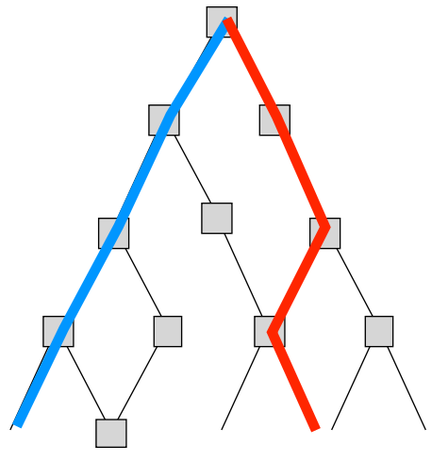
Static Race Detection



`x = 1` **LS_1 = {l}**

`x = 2` **LS_1 = {k}**

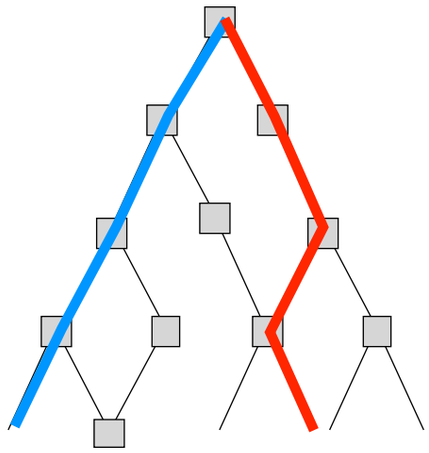
Static Race Detection



`x = 1` `x = 2`

$LS_1 = \{l\}$ $\cap$ $LS_1 = \{k\}$ $= \{\}$

Static Race Detection

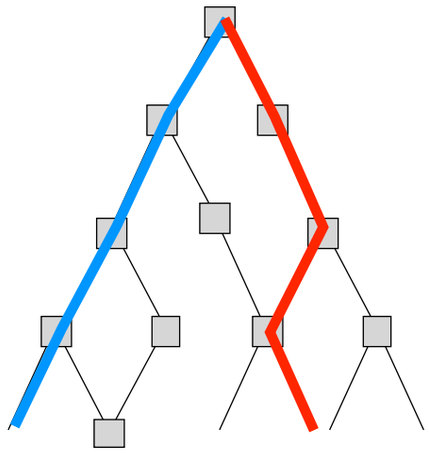


`x = 1` `x = 2`

$LS_1 = \{l\}$ $\cap$ $LS_1 = \{k\}$ $= \{\}$

=> `x = 1` and `x = 2` are RACING!

Static Race Detection



`x = 1` `x = 2`

$LS_1 = \{l\}$ $\cap$ $LS_1 = \{k\}$ $= \{\}$

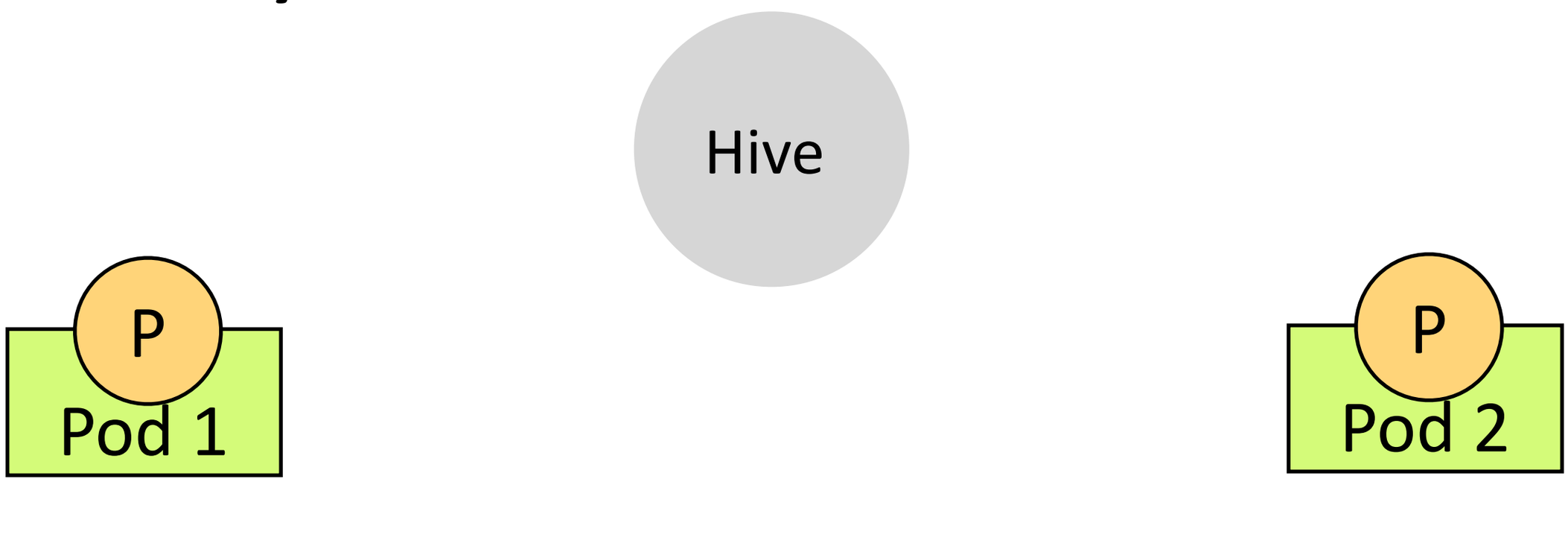
=> `x = 1` and `x = 2` are RACING!

Top-down, flow sensitive, interprocedural, lockset-based

Dynamic Race Validation

Dynamic Race Validation

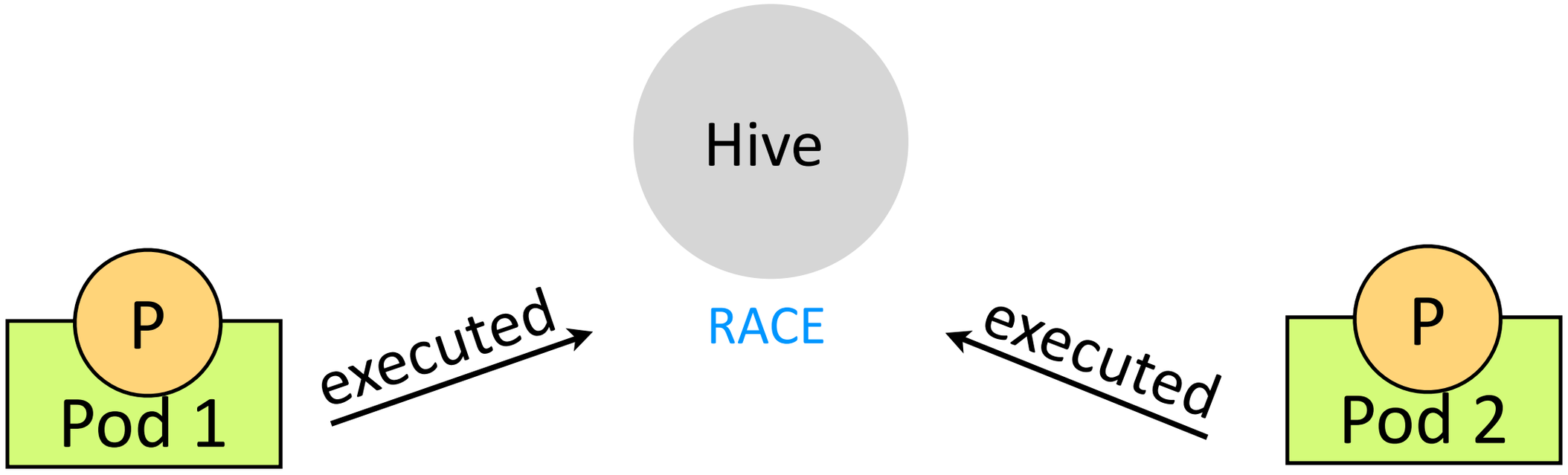
Dynamic Race Validation



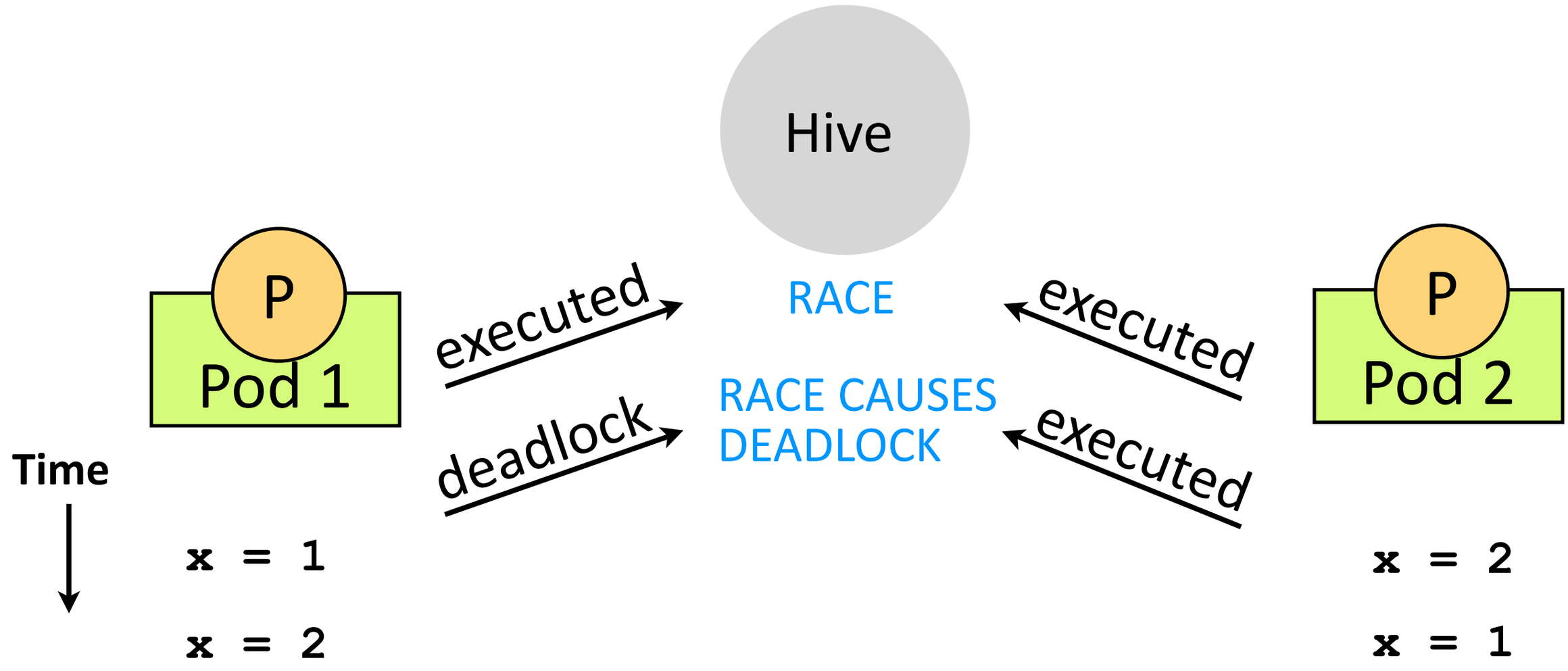

```
x = 1
x = 2
```

```
x = 2
x = 1
```

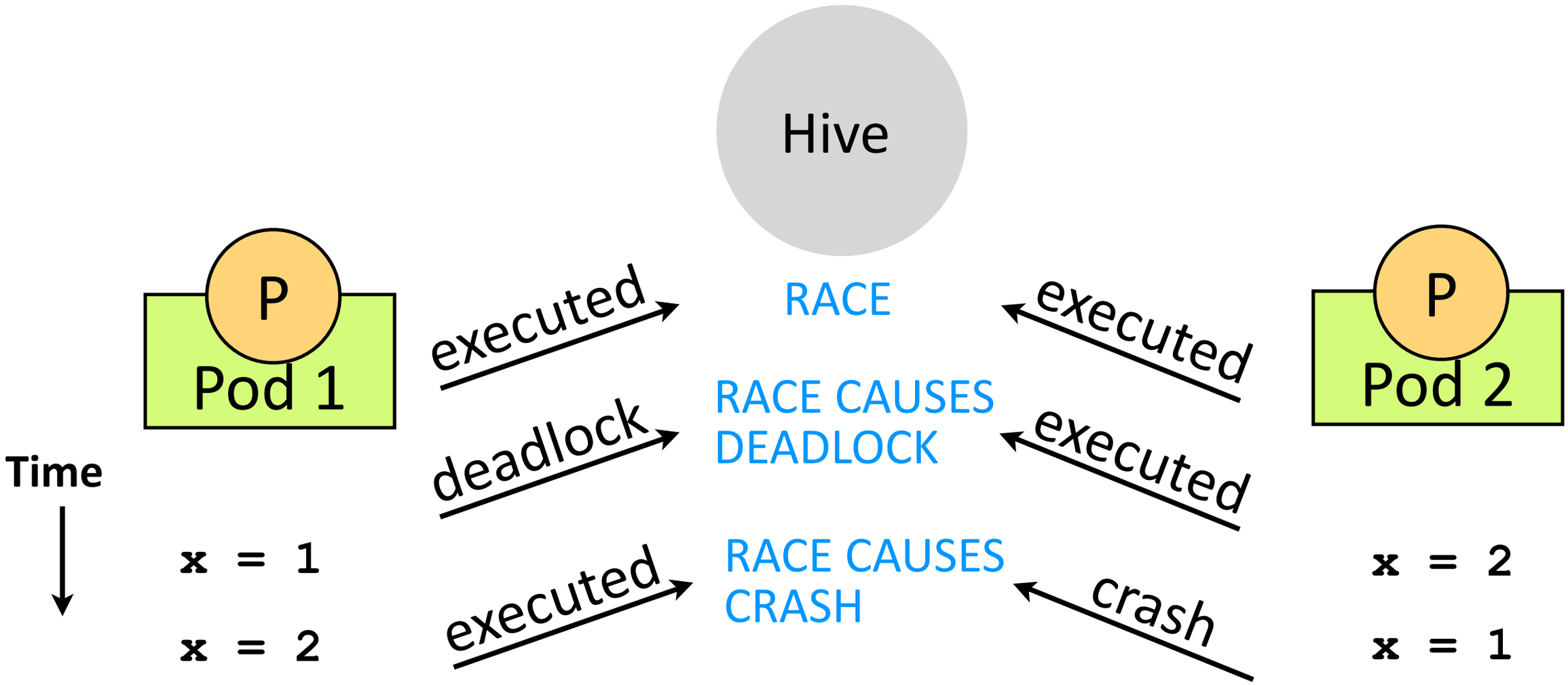
Dynamic Race Validation



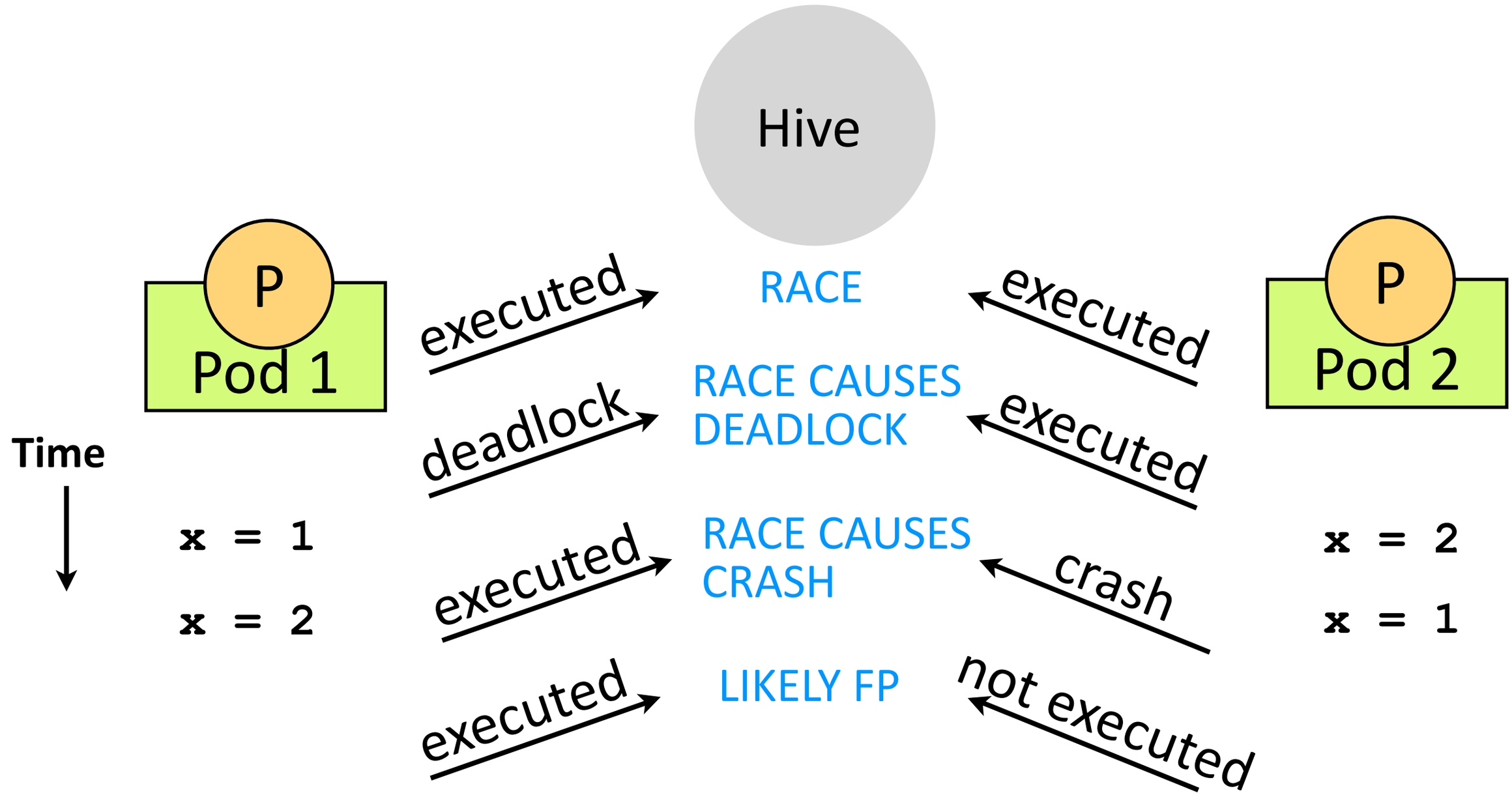
Dynamic Race Validation



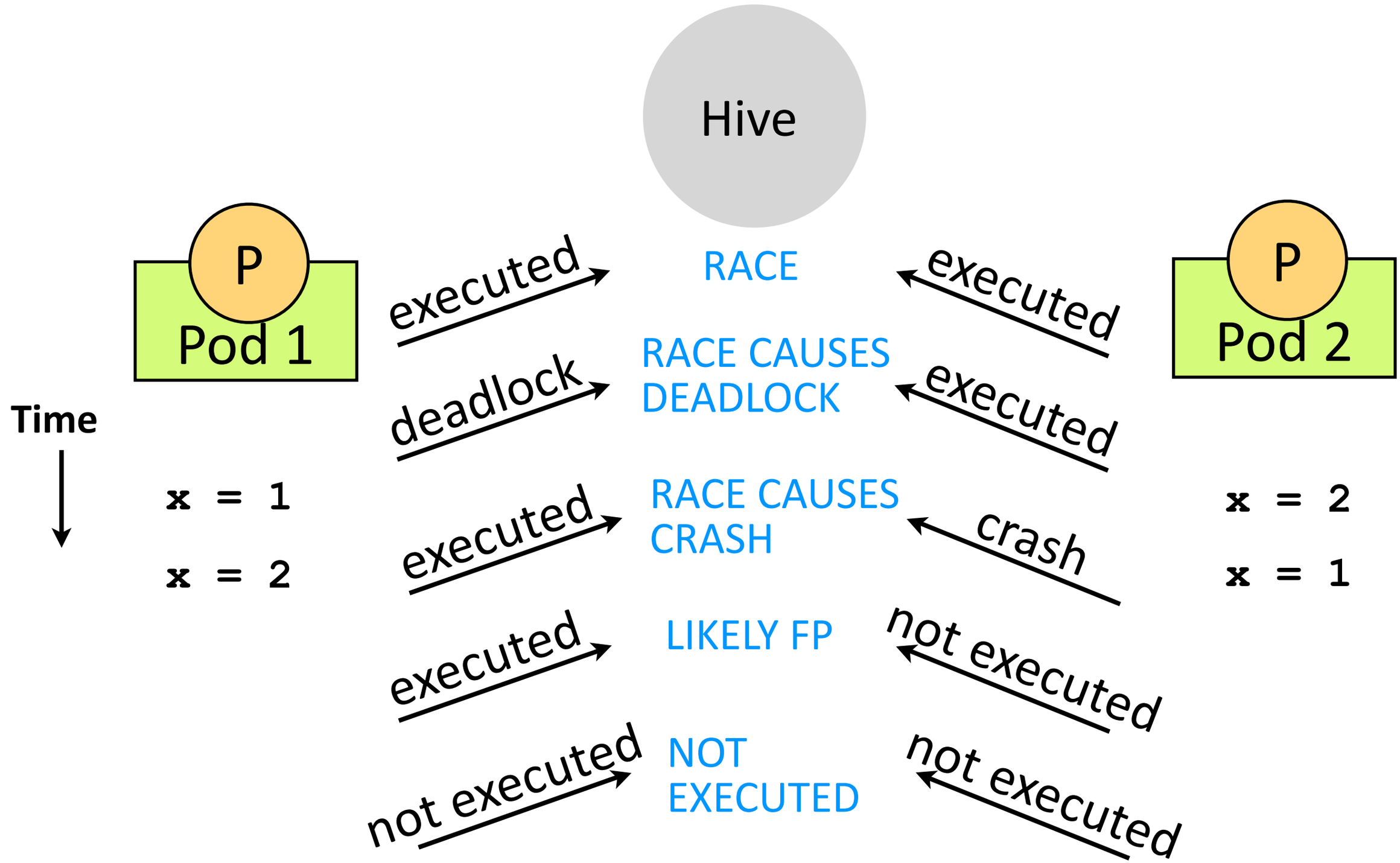
Dynamic Race Validation



Dynamic Race Validation



Dynamic Race Validation



Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL		
RACE		
RACE CAUSES DEADLOCK		
RACE CAUSES CRASH		
LIKELY FP		
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE		
RACE CAUSES DEADLOCK		
RACE CAUSES CRASH		
LIKELY FP		
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK		
RACE CAUSES CRASH		
LIKELY FP		
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH		
LIKELY FP		
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH	0	3
LIKELY FP		
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH	0	3
LIKELY FP	37	63
NOT EXECUTED		

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH	0	3
LIKELY FP	37	63
NOT EXECUTED	48	54

RUNTIME OVERHEAD		

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH	0	3
LIKELY FP	37	63
NOT EXECUTED	48	54

RUNTIME OVERHEAD	0.99%	0.91%

Detection Results and Efficiency

	SQLite	Pbzip2
TOTAL	88	122
RACE	0	1
RACE CAUSES DEADLOCK	4	0
RACE CAUSES CRASH	0	3
LIKELY FP	37	63
NOT EXECUTED	48	54

RUNTIME OVERHEAD	0.99%	0.91%

Effective and low overhead

Comparison to Other Detectors

- Dynamic detectors have high overhead

	SQLite	Pbzip2
CoRD	0.99%	0.91%
ThreadSanitizer	972%	3,001%

Comparison to Other Detectors

- Dynamic detectors have high overhead

	SQLite	Pbzip2
CoRD	0.99%	0.91%
ThreadSanitizer	972%	3,001%

- Static detectors have false positives and don’t provide any classification

Summary

- Collaborative race detection
  - *Statically detect races*
  - *Dynamically validate them*
- Effective
  - *Detected 8 real races in 2 real programs*
- Efficient
  - *Has < 1% overhead*

Roadmap

- Synthesizing fixes
- Privacy implications
- Extension to other types of bugs