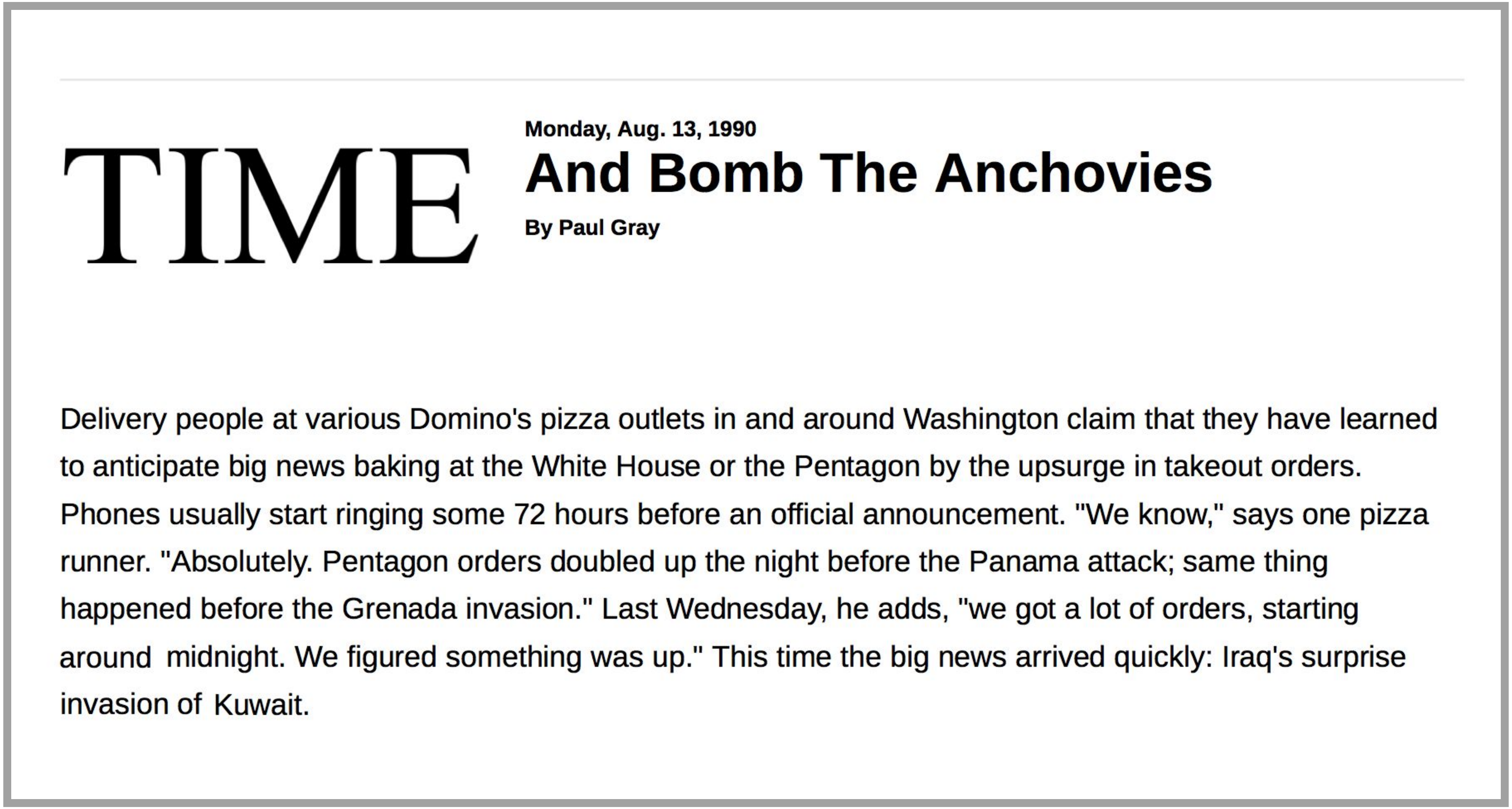
Information Leakage in Arbiter Protocols

Nestan Tsiskaridze, *Lucas Bang*, *Joseph McMahan*, *Tevfik Bultan* and *Timothy Sherwood*

University of California Santa Barbara

October 9, 2018

ATVA 2018: INTERNATIONAL SYMPOSIUM ON AUTOMATED TECHNOLOGY FOR VERIFICATION AND ANALYSIS



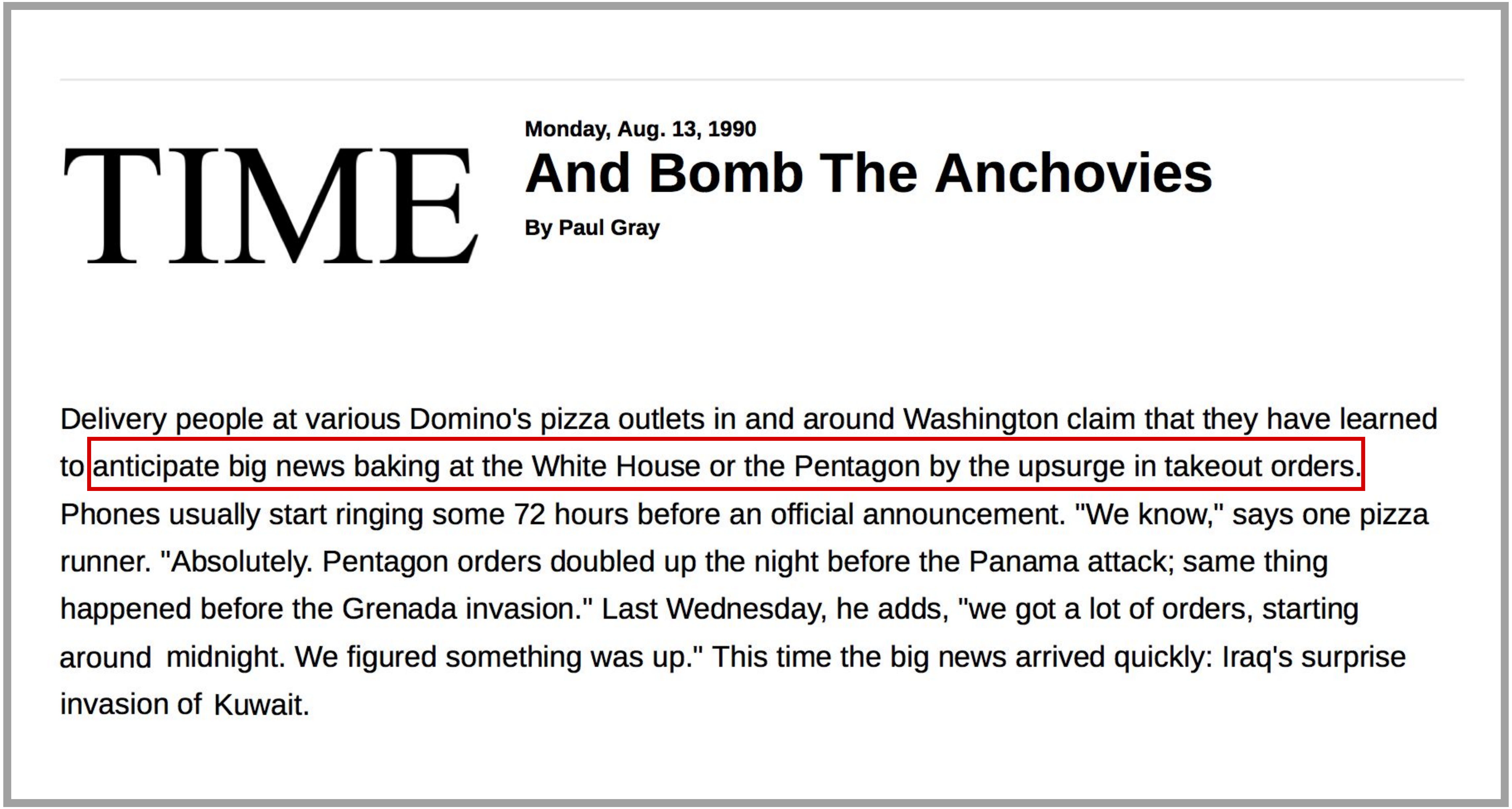
TIME

Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.



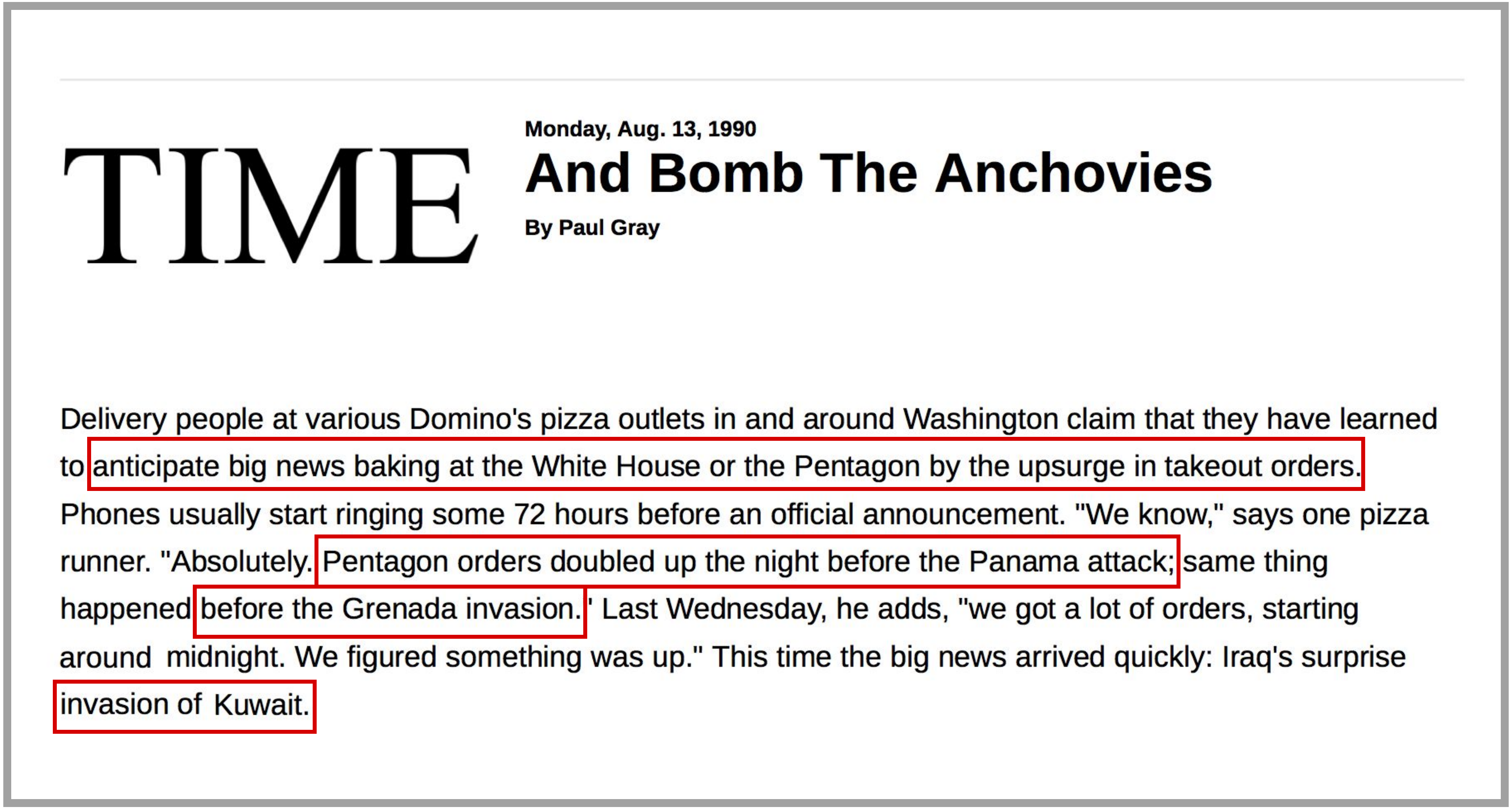
TIME

Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion." Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.



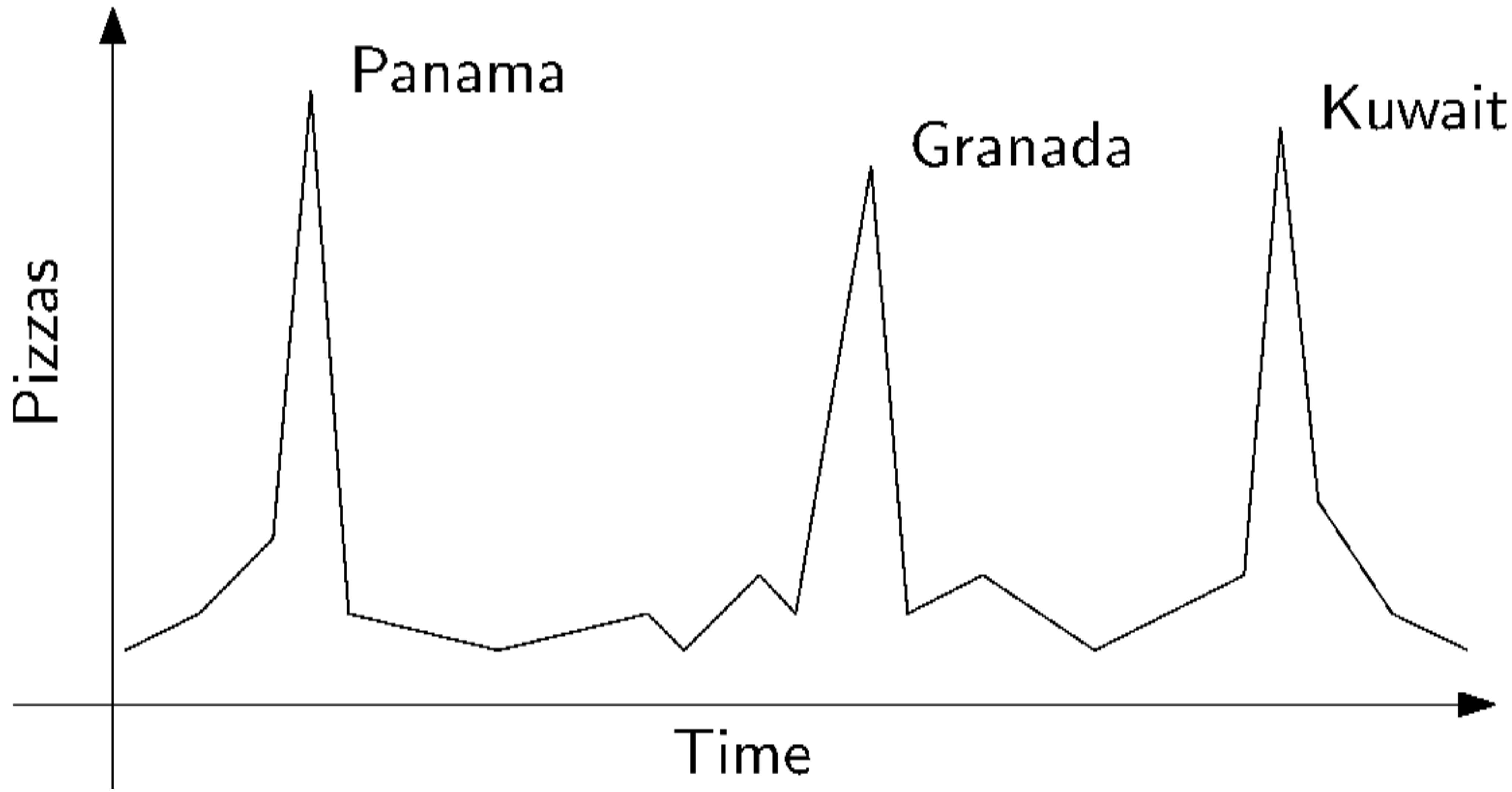
TIME

Monday, Aug. 13, 1990

And Bomb The Anchovies

By Paul Gray

Delivery people at various Domino's pizza outlets in and around Washington claim that they have learned to anticipate big news baking at the White House or the Pentagon by the upsurge in takeout orders. Phones usually start ringing some 72 hours before an official announcement. "We know," says one pizza runner. "Absolutely. Pentagon orders doubled up the night before the Panama attack; same thing happened before the Grenada invasion.' Last Wednesday, he adds, "we got a lot of orders, starting around midnight. We figured something was up." This time the big news arrived quickly: Iraq's surprise invasion of Kuwait.


Panama
Granada
Kuwait
Pizzas
Time

- **Does** the **information leak?**

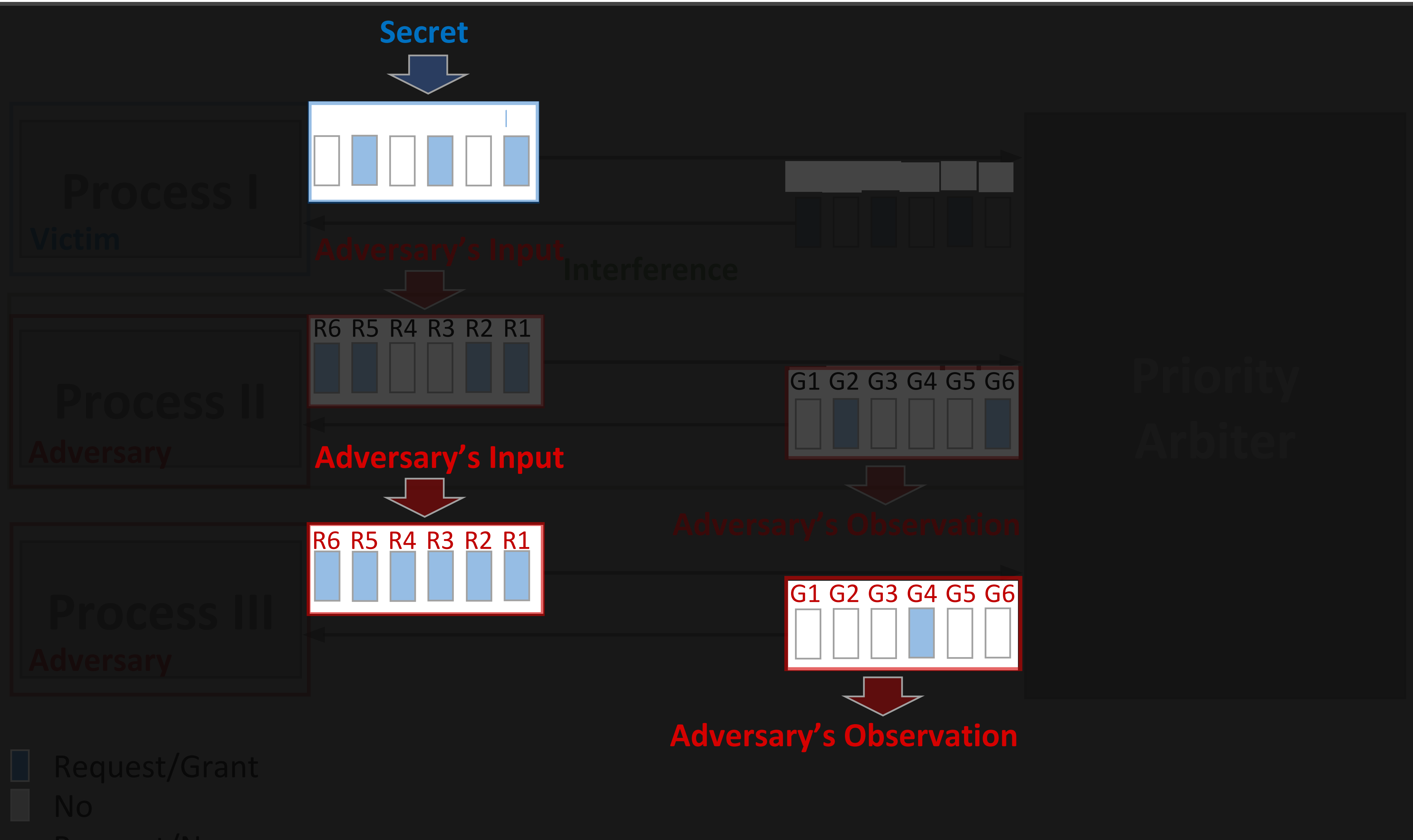
- **How much** information leaks**?**

- **Can** we **always prevent information** from **crossing** from **one domain** to **another?**

- **Can** we **identify** and **quantify** the **information leakage?**

- **Can** we **automate?**

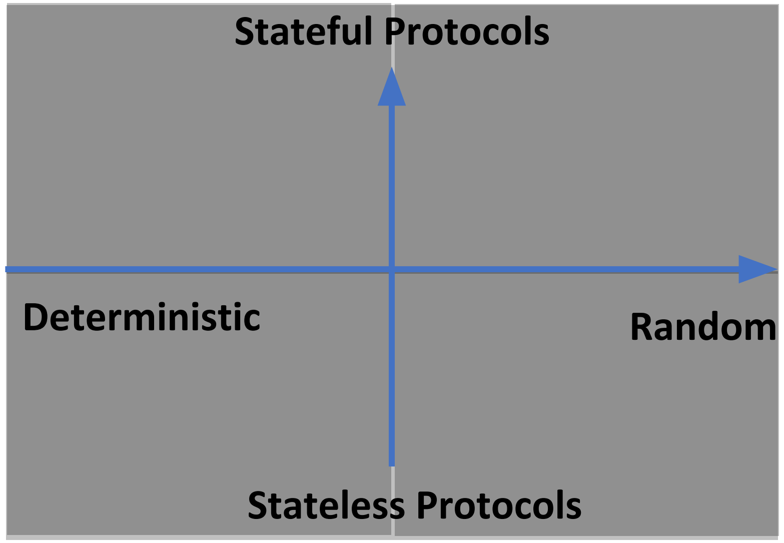
Priority Arbiter: process with the lowest ID gets access.



Arbiter Protocols

Categories:

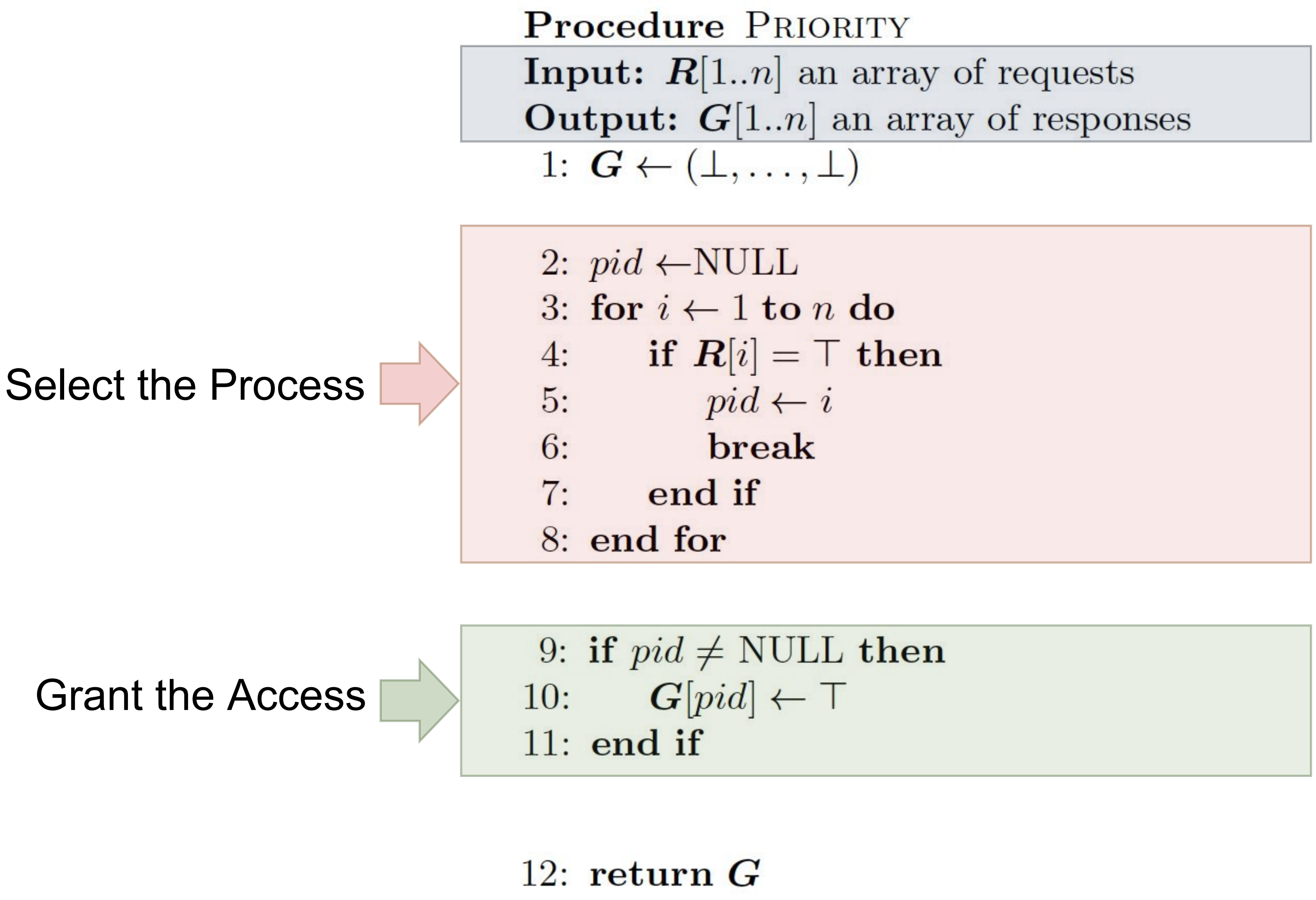
How to **resolve** **concurrent requests**?



Are the **processes** **stateful**/**stateless**?

A **process** is **stateless** if *requests* at each round are **independent** from those of the previous rounds; Otherwise, a **process** is **stateful**.

Stateless Protocols



Select the Process

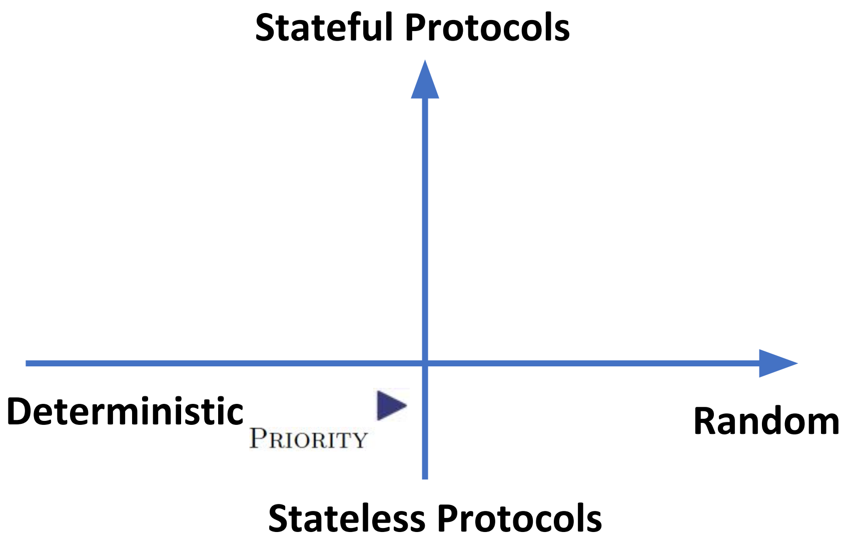
Grant the Access

```
Procedure PRIORITY
Input: $\boldsymbol{R}[1..n]$ an array of requests
Output: $\boldsymbol{G}[1..n]$ an array of responses
$\boldsymbol{G} \leftarrow (\bot, \ldots, \bot)$

pid $\leftarrow$ NULL
for $i \leftarrow 1$ to $n$ do
    if $\boldsymbol{R}[i] = \top$ then
        $pid \leftarrow i$
        break
    end if
end for

if $pid \neq$ NULL then
    $\boldsymbol{G}[pid] \leftarrow \top$
end if

return $\boldsymbol{G}$
```



Stateful Protocols

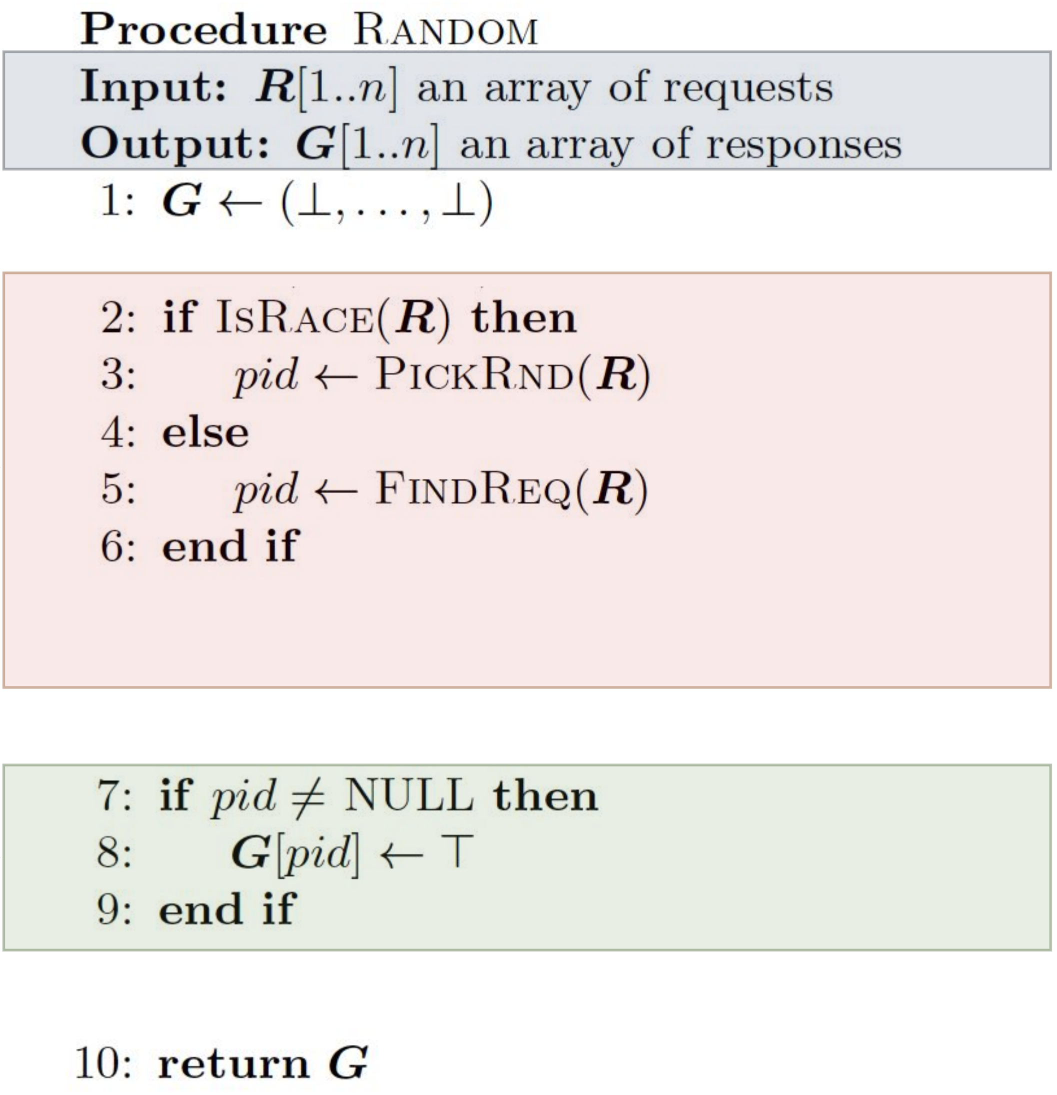
Deterministic

PRIORITY

Random

Stateless Protocols

Stateless Protocols



Procedure RANDOM

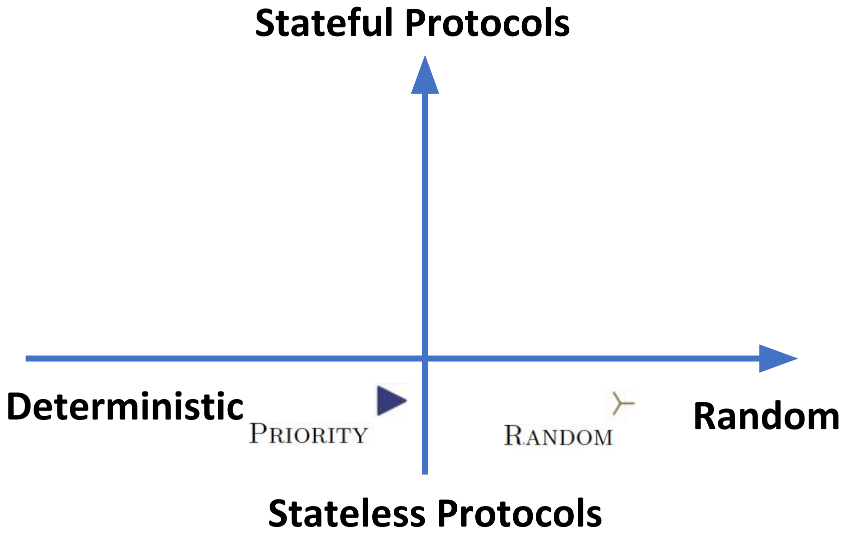
Input: $\boldsymbol{R}[1..n]$ an array of requests

Output: $\boldsymbol{G}[1..n]$ an array of responses

Select the Process

Grant the Access

```
$\boldsymbol{G} \leftarrow (\perp, \ldots, \perp)$
if IsRace($\boldsymbol{R}$) then
    $pid \leftarrow$ PickRnd($\boldsymbol{R}$)
else
    $pid \leftarrow$ FindReq($\boldsymbol{R}$)
end if
if $pid \neq$ NULL then
    $\boldsymbol{G}[pid] \leftarrow \top$
end if
return $\boldsymbol{G}$
```



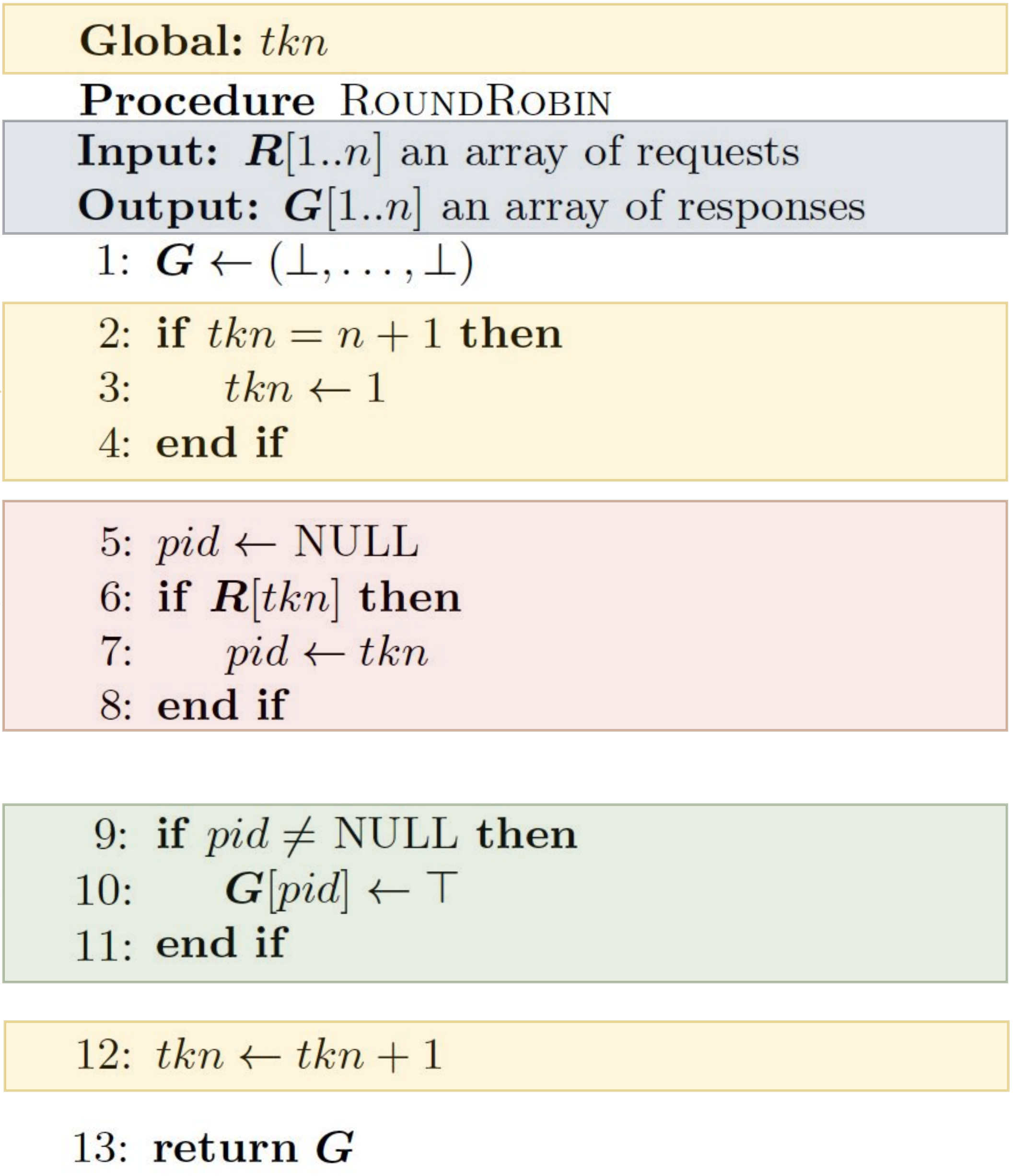
Stateful Protocols

Update the Global Data

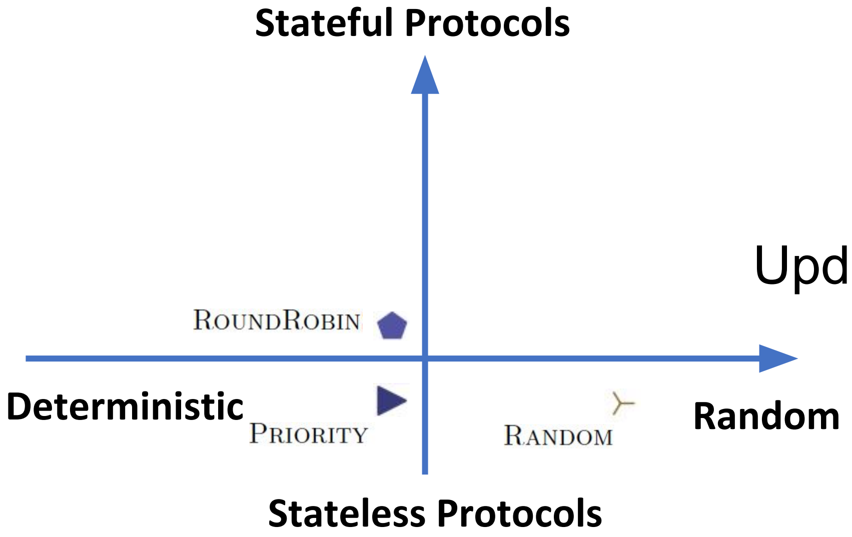
Select the Process

Grant the Access

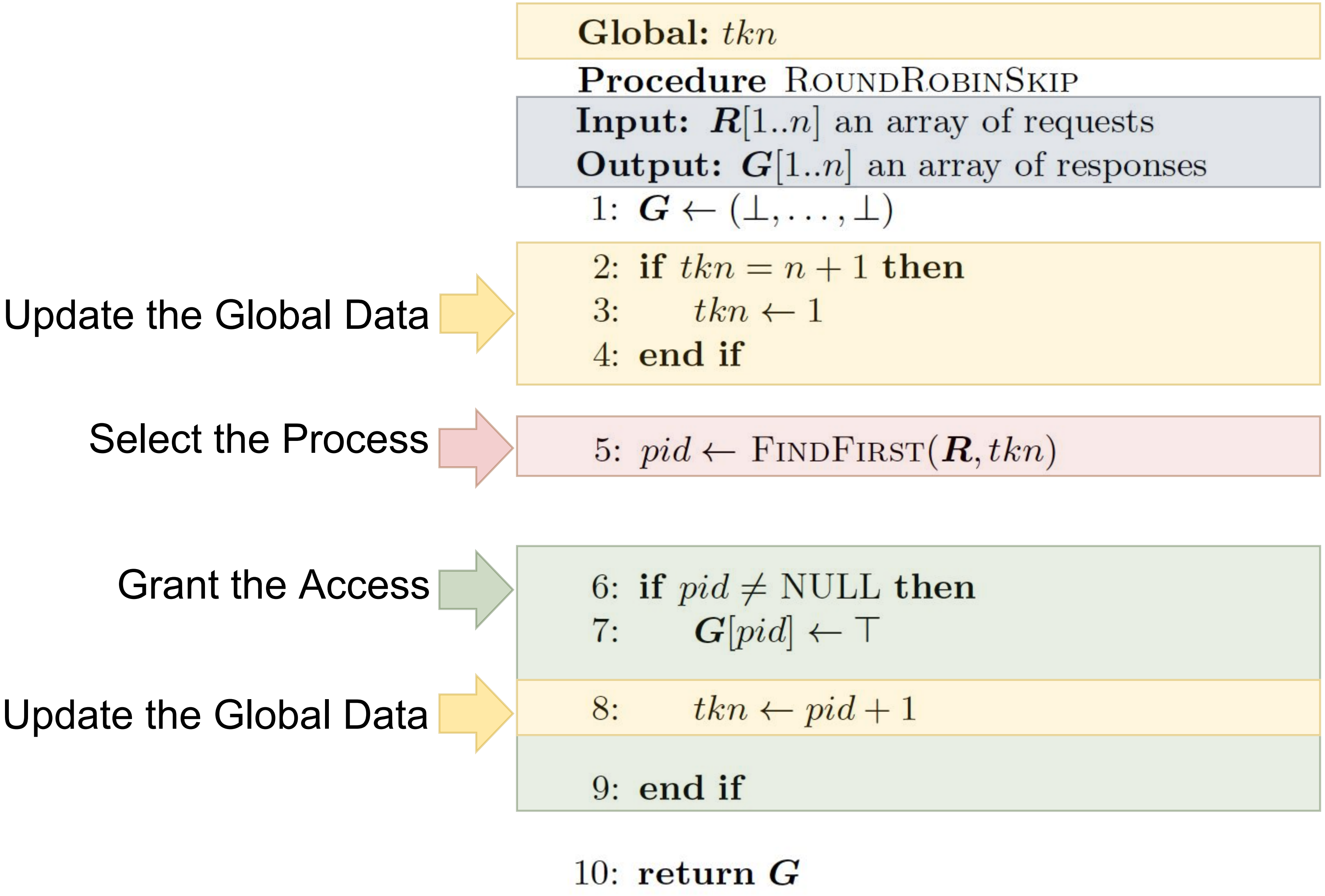
Update the Global Data



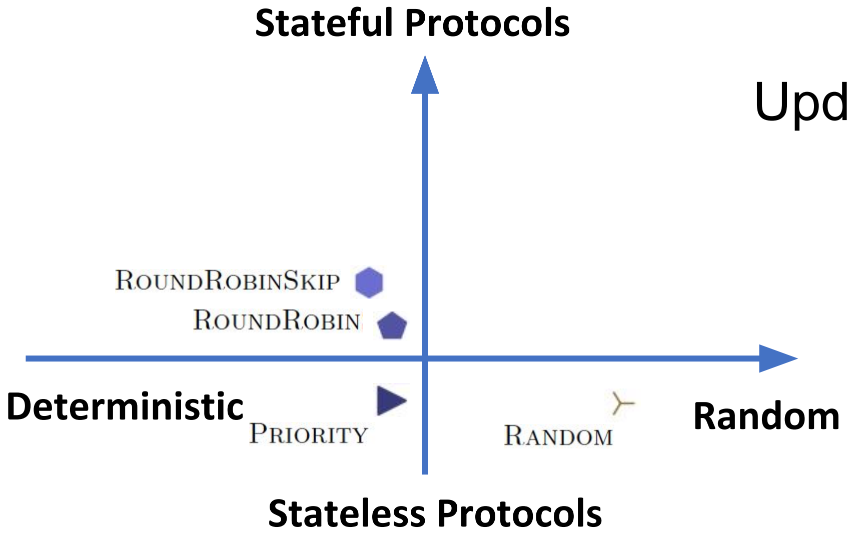
```
Global: $tkn$
Procedure RoundRobin
Input: $R[1..n]$ an array of requests
Output: $G[1..n]$ an array of responses
$G \leftarrow (\perp, \ldots, \perp)$
if $tkn = n + 1$ then
    $tkn \leftarrow 1$
end if
$pid \leftarrow$ NULL
if $R[tkn]$ then
    $pid \leftarrow tkn$
end if
if $pid \neq$ NULL then
    $G[pid] \leftarrow \top$
end if
$tkn \leftarrow tkn + 1$
return $G$
```



Stateful Protocols



```
Global: $tkn$
Procedure RoundRobinSkip
Input: $\boldsymbol{R}[1..n]$ an array of requests
Output: $\boldsymbol{G}[1..n]$ an array of responses
$\boldsymbol{G} \leftarrow (\bot, \ldots, \bot)$
if $tkn = n + 1$ then
    $tkn \leftarrow 1$
end if
$pid \leftarrow \textsc{FindFirst}(\boldsymbol{R}, tkn)$
if $pid \neq \text{NULL}$ then
    $\boldsymbol{G}[pid] \leftarrow \top$
    $tkn \leftarrow pid + 1$
end if
return $\boldsymbol{G}$
```



Stateful Protocols

```
Global: $\boldsymbol{W}[1..n]$ an array of wait-times
Procedure LOTTERY
Input: $\boldsymbol{R}[1..n]$ an array of requests
Output: $\boldsymbol{G}[1..n]$ an array of responses
$\boldsymbol{G} \leftarrow (\bot, \ldots, \bot)$
```

Update the Global Data

```
for $i \leftarrow 1$ to $n$ do
    if $\boldsymbol{R}[i] = \top$ then
        $\boldsymbol{W}[i] \leftarrow \boldsymbol{W}[i] + 1$
    else
        $\boldsymbol{W}[i] \leftarrow 0$
    end if
end for
```

Select the Process

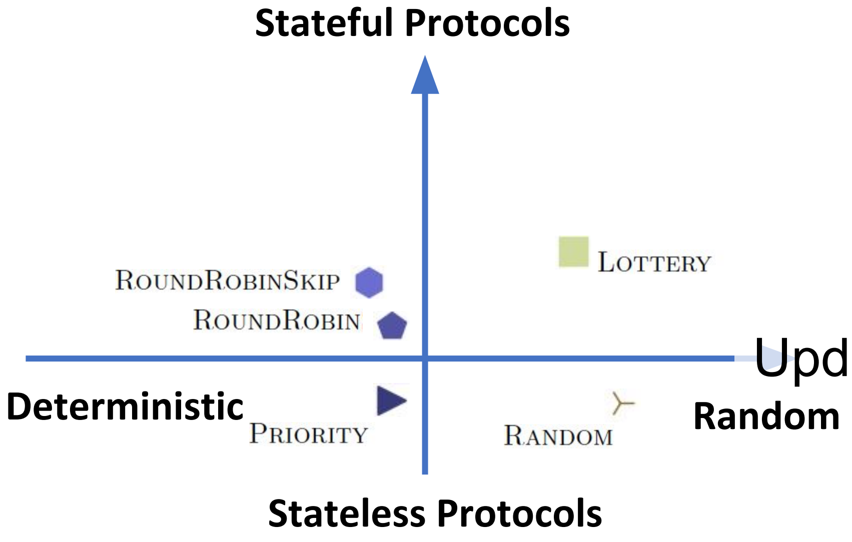
```
if IsRace($\boldsymbol{R}$) then
    $pid \leftarrow$ PickRnd($\boldsymbol{W}$)
else
    $pid \leftarrow$ FindReq($\boldsymbol{R}$)
end if
```

Grant the Access

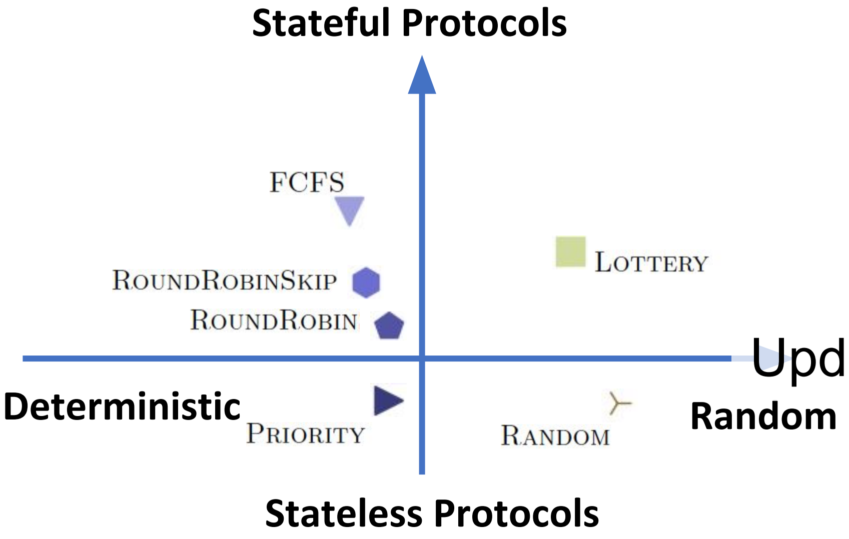
```
if $pid \neq$ NULL then
    $\boldsymbol{G}[pid] \leftarrow \top$
```

Update the Global Data

```
    $\boldsymbol{W}[pid] \leftarrow 0$
end if
return $\boldsymbol{G}$
```



Stateful Protocols

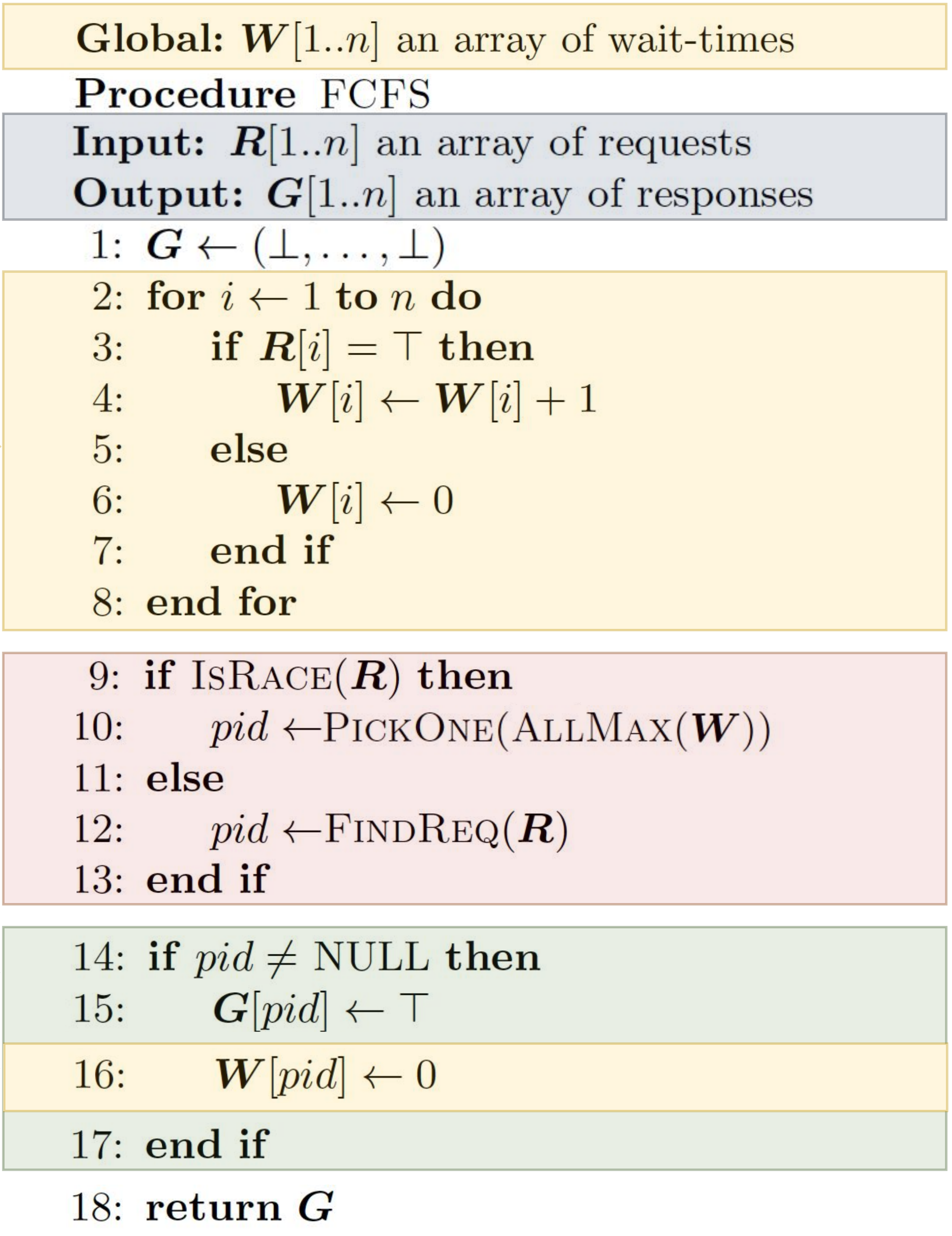


Update the Global Data

Select the Process

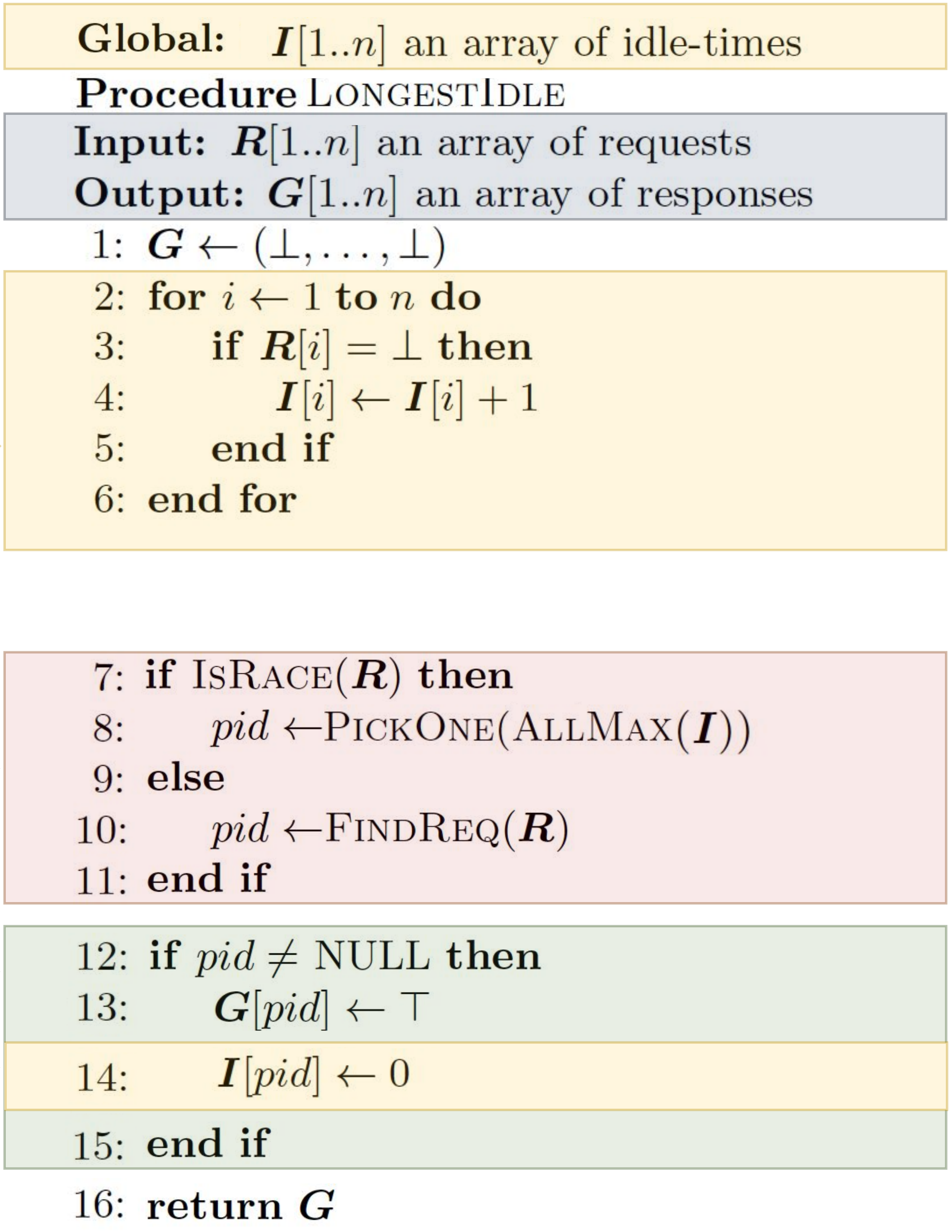
Grant the Access

Update the Global Data



```
Global: $W[1..n]$ an array of wait-times
Procedure FCFS
Input: $R[1..n]$ an array of requests
Output: $G[1..n]$ an array of responses
G ← $(\bot, \ldots, \bot)$
for $i \leftarrow 1$ to $n$ do
    if $R[i] = \top$ then
        $W[i] \leftarrow W[i] + 1$
    else
        $W[i] \leftarrow 0$
    end if
end for
if IsRace($R$) then
    $pid \leftarrow$PickOne(AllMax($W$))
else
    $pid \leftarrow$FindReq($R$)
end if
if $pid \neq$ NULL then
    $G[pid] \leftarrow \top$
    $W[pid] \leftarrow 0$
end if
return $G$
```

Stateful Protocols



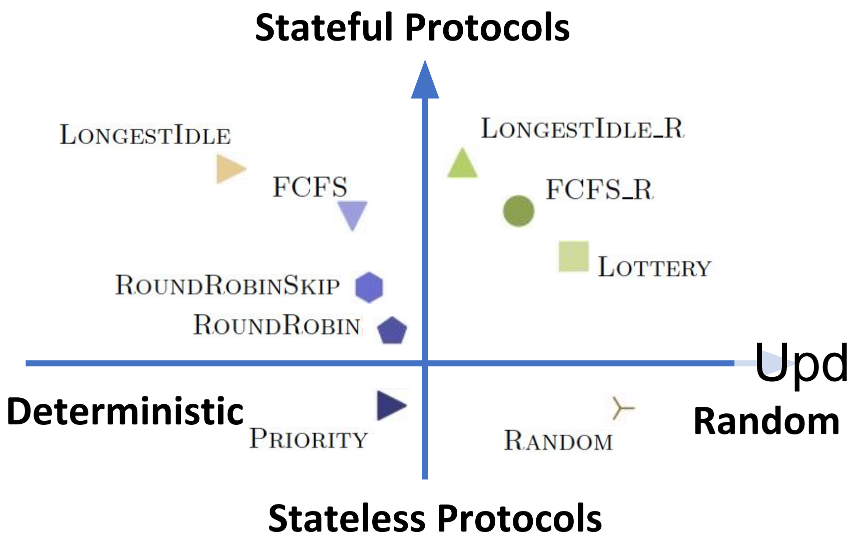
```
Global: $\boldsymbol{I}[1..n]$ an array of idle-times
Procedure LongestIdle
Input: $\boldsymbol{R}[1..n]$ an array of requests
Output: $\boldsymbol{G}[1..n]$ an array of responses
$\boldsymbol{G} \leftarrow (\bot, \ldots, \bot)$
for $i \leftarrow 1$ to $n$ do
    if $\boldsymbol{R}[i] = \bot$ then
        $\boldsymbol{I}[i] \leftarrow \boldsymbol{I}[i] + 1$
    end if
end for
if IsRace($\boldsymbol{R}$) then
    $pid \leftarrow$ PickOne(AllMax($\boldsymbol{I}$))
else
    $pid \leftarrow$ FindReq($\boldsymbol{R}$)
end if
if $pid \neq$ NULL then
    $\boldsymbol{G}[pid] \leftarrow \top$
    $\boldsymbol{I}[pid] \leftarrow 0$
end if
return $\boldsymbol{G}$
```

Update the Global Data

Select the Process

Grant the Access

Update the Global Data



Information Leakage in **Arbiter Protocols**

Arbiter Protocol Model

H – a **secret/private input**, **high-security** input: **Victim's Requests** {**R1**, **R2**, ..., **Rn**};

L – a **public input**, the **low security** input: **Adversary's Requests**: {**R1**, **R2**, ..., **Rn**};

O – a **output observation**: **Adversary's Access Grants**: {**G1**, **G2**, ..., **Gn**};

Before invoking the system: the adversary has some **initial uncertainty** about the value of H .

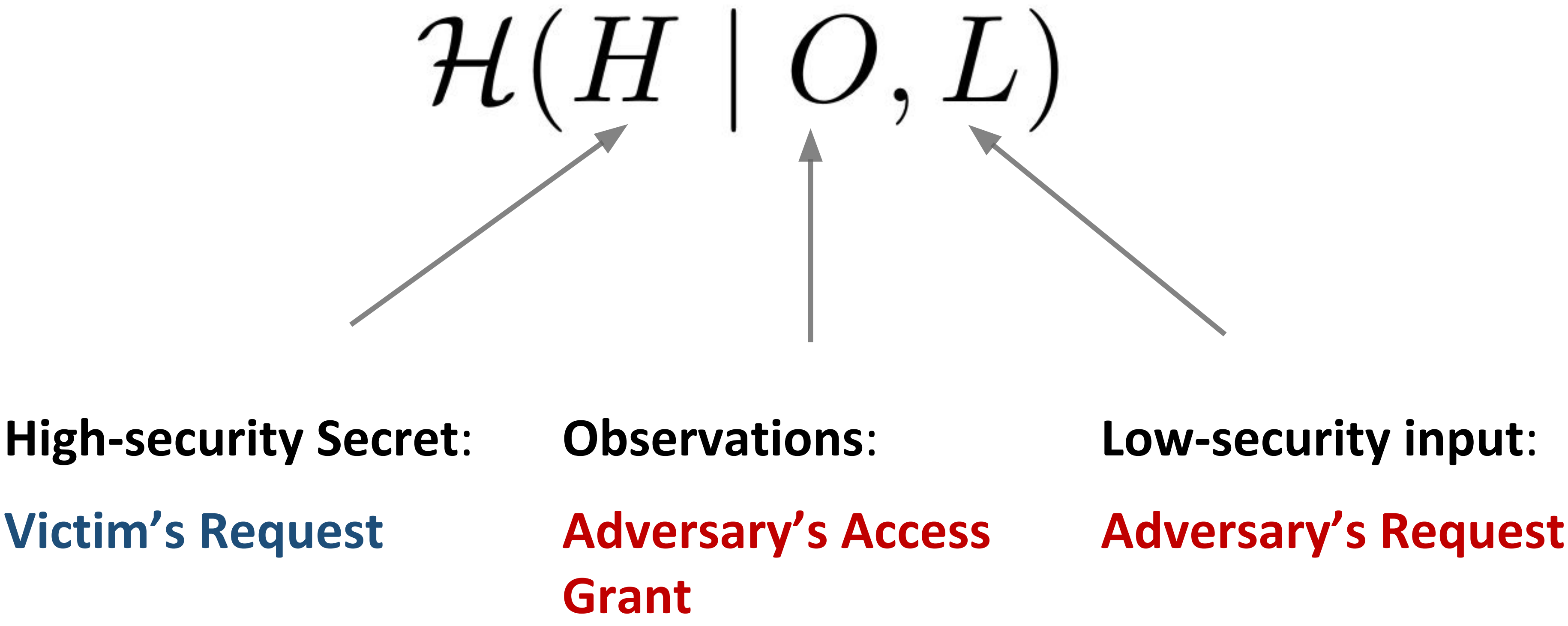
After observing O : some amount of information is **leaked**,

the adversary's **uncertainty** about H is **reduced**.

Information Entropy as a **measurement** of **uncertainty**

Shannon Entropy

Quantifies Information Gain:



$$\mathcal{H}(H \mid O, L)$$

High-security Secret:

Victim's Request

Observations:

Adversary's Access Grant

Low-security input:

Adversary's Request

Shannon Entropy

$$\mathcal{H}(H \mid O, L) = \sum_{\omega,l} P(\omega \mid l) \sum_{h} P(h \mid \omega, l) \log_2 \frac{1}{P(h \mid \omega, l)}$$

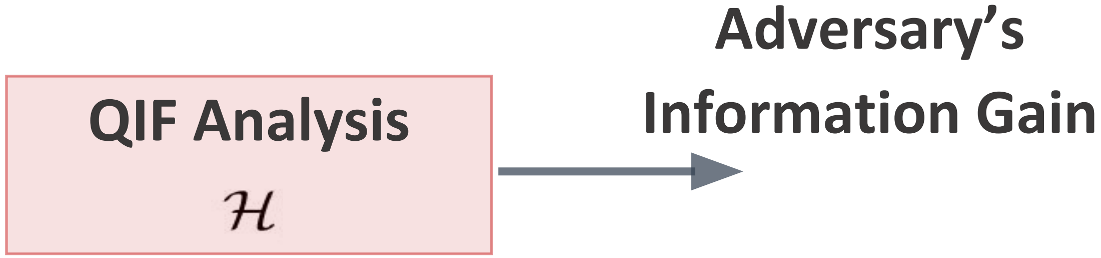
Expected maximal amount of information leaked:

$$\mathcal{I}(H, O, L) = \max_{l}(\mathcal{H}_{init}(H \mid l) - \mathcal{H}_{fin}(H \mid O, l))$$

$$\mathcal{H}_{init}(H \mid l) = \sum_{h} P(h \mid l) \log_2 \frac{1}{P(h \mid l)}$$

$$\mathcal{H}_{fin}(H \mid O, l) = \sum_{\omega} P(\omega \mid l) \sum_{h} P(h \mid \omega, l) \log_2 \frac{1}{P(h \mid \omega, l)}$$

Information Entropy as a **measurement** of **uncertainty**



QIF Analysis

$\mathcal{H}$

Adversary's Information Gain

- **How** do we capture **all behaviors** of a protocol**?**

Symbolic Execution

Extracts path constraints from a system by executing it on symbolic inputs, as opposed to concrete inputs.

We **adopt** and **extend symbolic execution** techniques to **automatically extract constraints** that relate **secret values** with **adversary's observations**.

Symbolic Execution

$\phi(H, L)$ – a **path constraint** from a **traditional Symbolic Execution** tool.

Extend $\phi(H, L)$ with an **event constraint**:

- **How** do we **handle random** components in **symbolic analysis?**

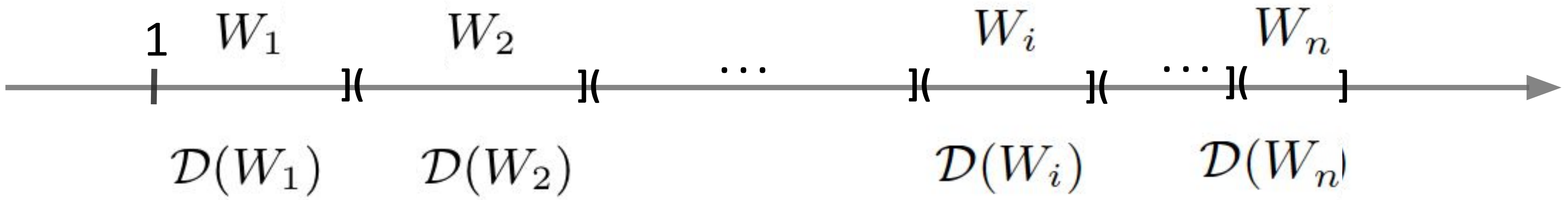
Random in Symbolic Analysis

R – a **random variable**;

$(R_1, \ldots, R_n)$ – the **domain** of R .

$\mathbf{W} = (W_1, \ldots, W_n)$ – **probability weights** for R , with $W_i \in \mathbb{Z}^+$.

Domain Interval



$$\mathcal{D}(W_i) = \begin{cases} [1, W_i], & i = 1 \\ \left(\sum_{j=1}^{i-1} W_j, \sum_{j=1}^{i} W_j\right], & 1 < i \leq n \end{cases}$$

Random in Symbolic Analysis

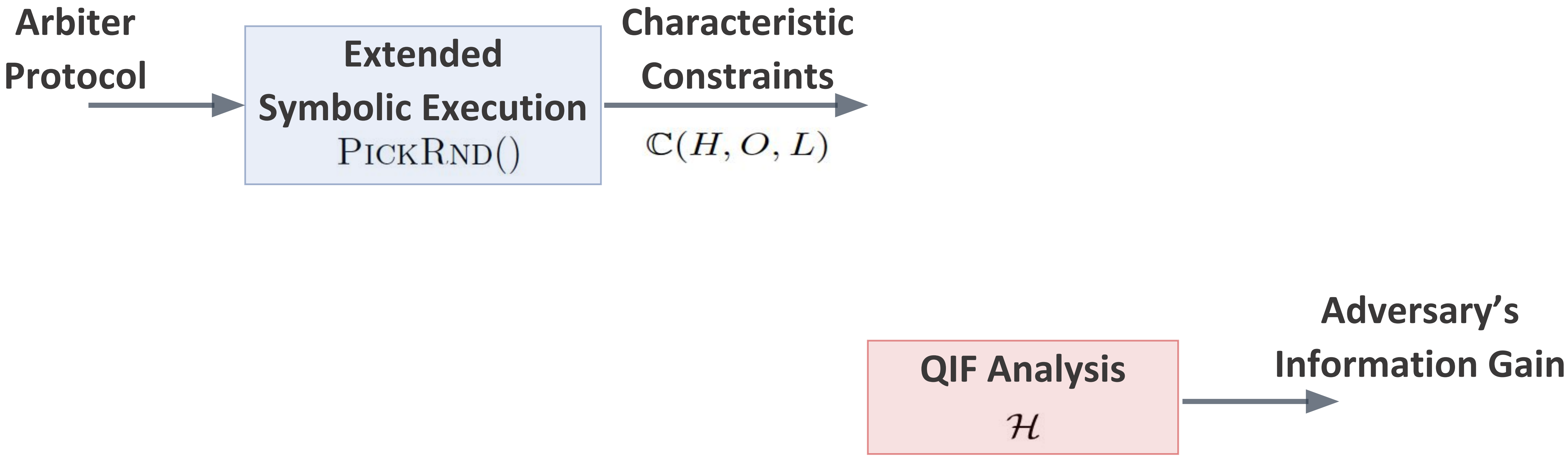
sym_R – a **fresh symbolic integer variable**;

$\textsc{PickRnd}()$ – **Selects a value** from a domain with a weighted-random distribution;
Simulates the desired **random generator behavior**;
Extends **path constraints** to reflect the relation between sym_R and R.

```
for $id \leftarrow 1$ to $id \leq n$ do
    if $\boldsymbol{W}[id] > 0$ and $sym\_R \in \mathcal{D}(\boldsymbol{W}[id])$ then
        return $id$
    end if
end for
return NULL
```

- **How** do we **handle random** components in **symbolic analysis?**

- **How** do we capture **all behaviors** of a protocol**?**



Arbiter Protocol

Extended Symbolic Execution

$\textsc{PickRnd}()$

Characteristic Constraints

$\mathbb{C}(H, O, L)$

QIF Analysis

$\mathcal{H}$

Adversary's Information Gain

- **How** do we **compute** the **probabilities**?

Model Counting

$$P(\omega|l) = \frac{\text{\# cases adversary \textbf{observes} } \omega \text{ after \textbf{requesting} } l}{\text{Total \# cases when adversary \textbf{requests} } l}$$

$$P(h \mid \omega, l) = \frac{\text{\# cases the \textbf{secret is} } h \text{ when adversary \textbf{observes} } \omega \text{ after \textbf{requesting} } l}{\text{Total \# cases when adversary \textbf{observes} } \omega \text{ after \textbf{requesting} } l}$$

Model Counting: Range Constraints $\mathcal{RC}$

Grammar

$$\begin{array}{lcl} \mathcal{C} & \rightarrow & \mathcal{C} \wedge \mathcal{C} \mid \mathcal{R} \\ \mathcal{R} & \rightarrow & \mathcal{B} = \top \mid \mathcal{B} = \bot \mid \mathcal{I} \in [a, b] \end{array}$$

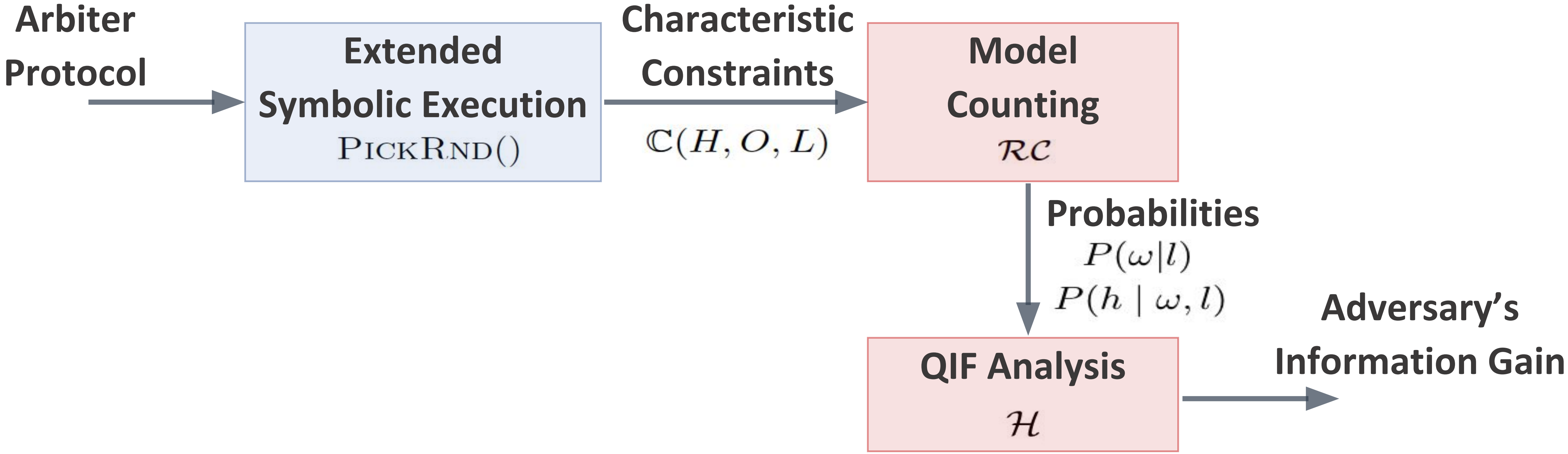
$\mathcal{B}$ - **Boolean** variables

$\mathcal{I}$ - **Integer** variables

Model Counter

```
for each $\mathbb{C}_\phi$ in $\mathbb{C}$ do
    $m \leftarrow \#\textsc{FreeVars}(\mathbb{C}_\phi, P_\mathcal{V}, P_\mathcal{A})$
    $s \leftarrow 2^m$
    for each $(r \in [a, b])$ in $I$ do
        $s \leftarrow (b - a + 1) \times s$
    end for
end for
```

- **How** do we **compute** the **probabilities**?



Arbiter Protocol

Extended Symbolic Execution

$\textsc{PickRnd}()$

Characteristic Constraints

$\mathbb{C}(H, O, L)$

Model Counting

$\mathcal{RC}$

Probabilities

$P(\omega|l)$

$P(h \mid \omega, l)$

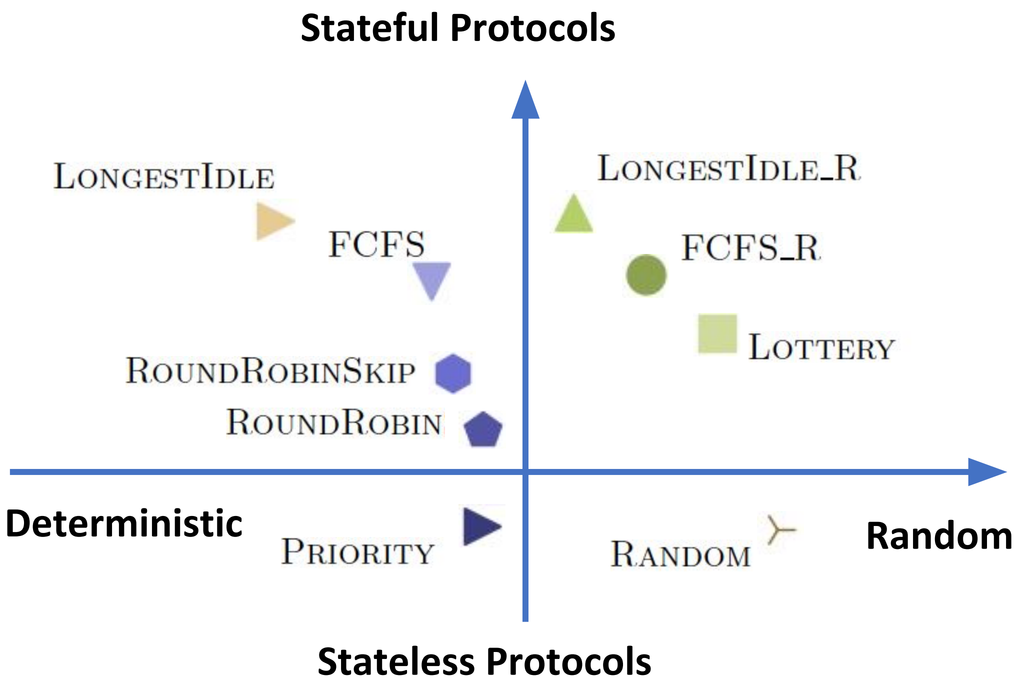
QIF Analysis

$\mathcal{H}$

Adversary's Information Gain

Experiments

- **Arbiter Protocols** (in Java):

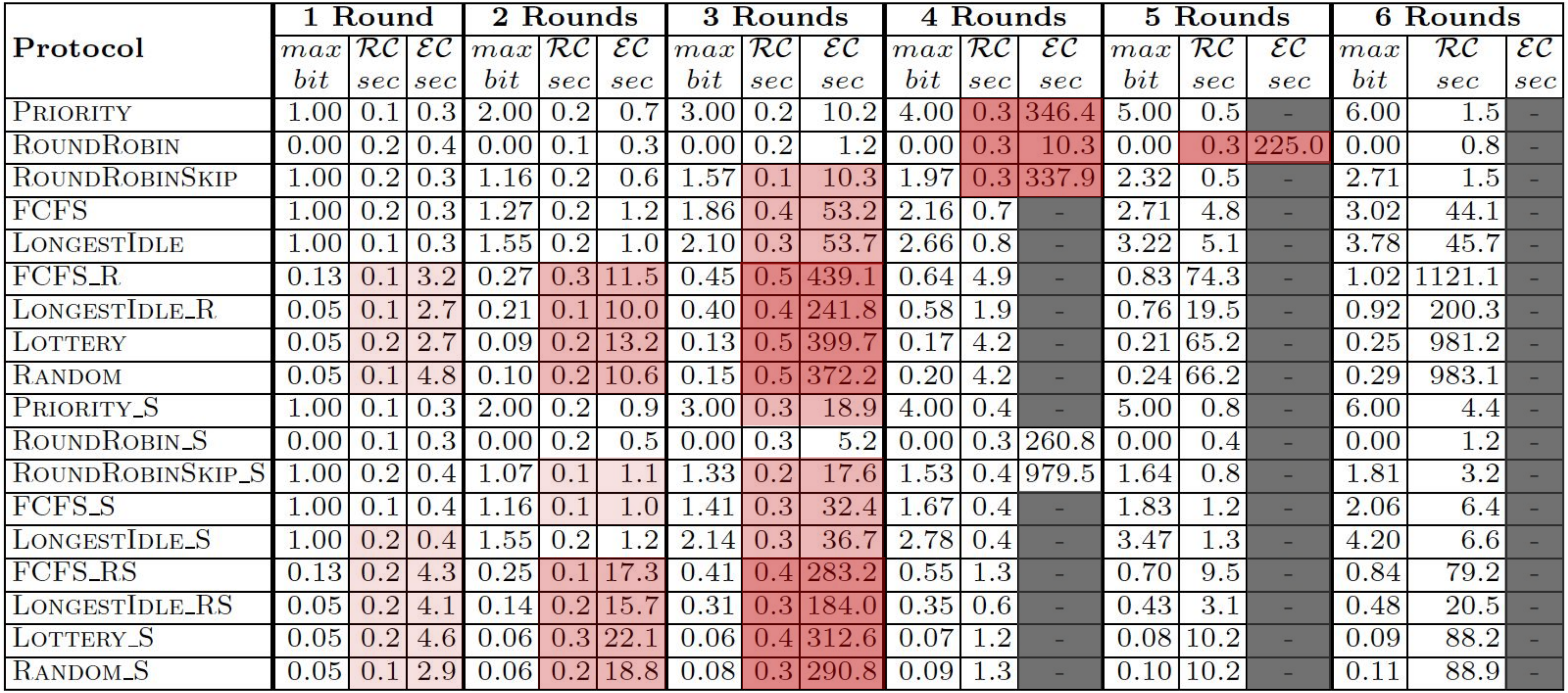


- **Processes:** **Victim Process**, **Adversary Process**, **Benign Process**.
- **Rounds:** **1 to 6**.

We **extended** SPF (Symbolic Java PathFinder). **Implemented** $\textsc{PickRnd}()$ and $\mathcal{RC}$.

Max Leakage (*bits*) **and Execution Time** (*seconds*)

$\mathcal{RC}/\mathcal{EC}$ ~ 1.4x – 2,647x faster.
Avg. speedup: 250x (excluding $\mathcal{EC}$ time outs).



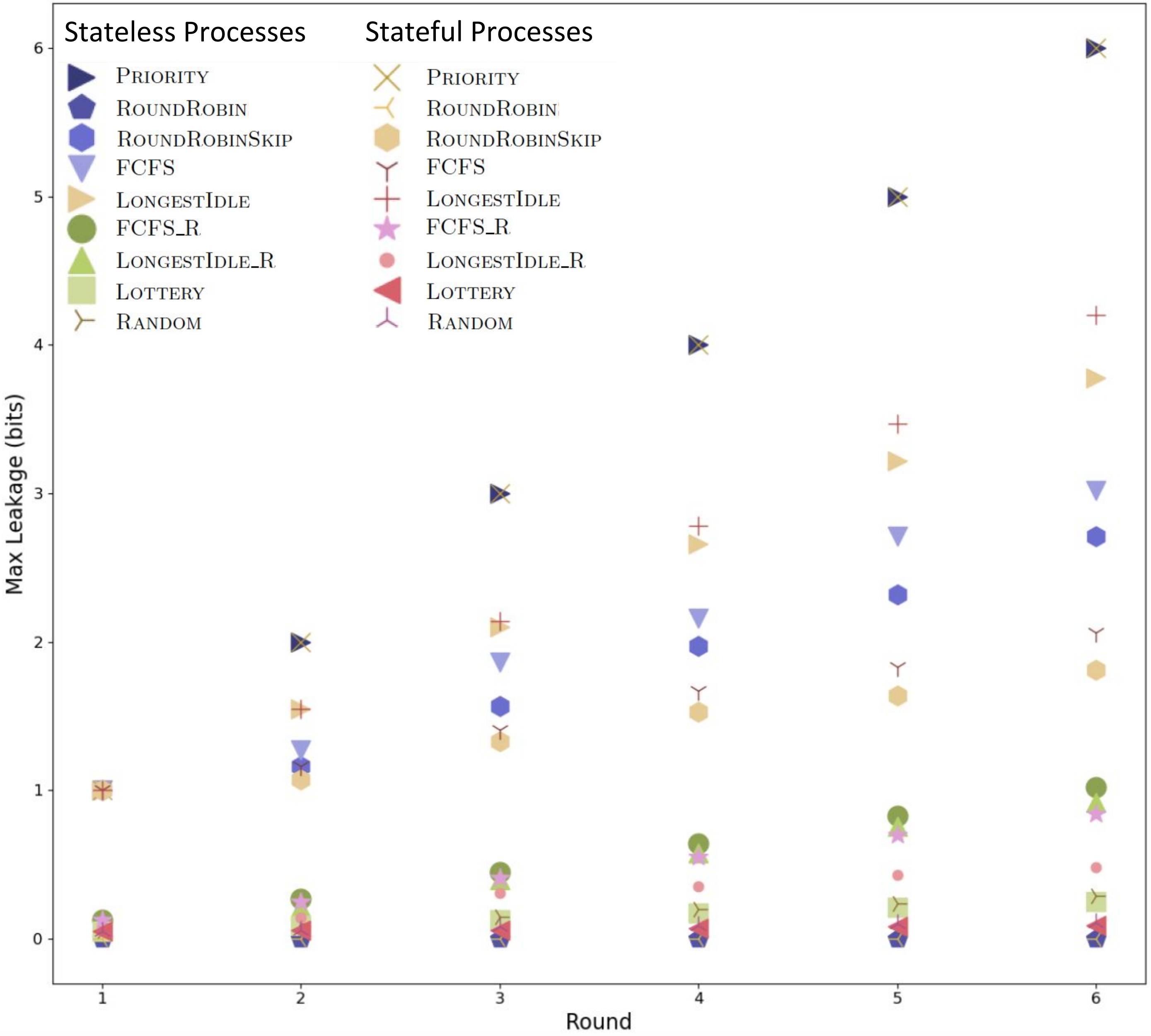
Protocol	1 Round			2 Rounds			3 Rounds			4 Rounds			5 Rounds			6 Rounds		
	max *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*	*max* *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*	*max* *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*	*max* *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*	*max* *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*	*max* *bit*	$\mathcal{RC}$ *sec*	$\mathcal{EC}$ *sec*
Priority	1.00	0.1	0.3	2.00	0.2	0.7	3.00	0.2	10.2	4.00	0.3	346.4	5.00	0.5	-	6.00	1.5	-
RoundRobin	0.00	0.2	0.4	0.00	0.1	0.3	0.00	0.2	1.2	0.00	0.3	10.3	0.00	0.3	225.0	0.00	0.8	-
RoundRobinSkip	1.00	0.2	0.3	1.16	0.2	0.6	1.57	0.1	10.3	1.97	0.3	337.9	2.32	0.5	-	2.71	1.5	-
FCFS	1.00	0.2	0.3	1.27	0.2	1.2	1.86	0.4	53.2	2.16	0.7	-	2.71	4.8	-	3.02	44.1	-
LongestIdle	1.00	0.1	0.3	1.55	0.2	1.0	2.10	0.3	53.7	2.66	0.8	-	3.22	5.1	-	3.78	45.7	-
FCFS_R	0.13	0.1	3.2	0.27	0.3	11.5	0.45	0.5	439.1	0.64	4.9	-	0.83	74.3	-	1.02	1121.1	-
LongestIdle_R	0.05	0.1	2.7	0.21	0.1	10.0	0.40	0.4	241.8	0.58	1.9	-	0.76	19.5	-	0.92	200.3	-
Lottery	0.05	0.2	2.7	0.09	0.2	13.2	0.13	0.5	399.7	0.17	4.2	-	0.21	65.2	-	0.25	981.2	-
Random	0.05	0.1	4.8	0.10	0.2	10.6	0.15	0.5	372.2	0.20	4.2	-	0.24	66.2	-	0.29	983.1	-
Priority_S	1.00	0.1	0.3	2.00	0.2	0.9	3.00	0.3	18.9	4.00	0.4	-	5.00	0.8	-	6.00	4.4	-
RoundRobin_S	0.00	0.1	0.3	0.00	0.2	0.5	0.00	0.3	5.2	0.00	0.3	260.8	0.00	0.4	-	0.00	1.2	-
RoundRobinSkip_S	1.00	0.2	0.4	1.07	0.1	1.1	1.33	0.2	17.6	1.53	0.4	979.5	1.64	0.8	-	1.81	3.2	-
FCFS_S	1.00	0.1	0.4	1.16	0.1	1.0	1.41	0.3	32.4	1.67	0.4	-	1.83	1.2	-	2.06	6.4	-
LongestIdle_S	1.00	0.2	0.4	1.55	0.2	1.2	2.14	0.3	36.7	2.78	0.4	-	3.47	1.3	-	4.20	6.6	-
FCFS_RS	0.13	0.2	4.3	0.25	0.1	17.3	0.41	0.4	283.2	0.55	1.3	-	0.70	9.5	-	0.84	79.2	-
LongestIdle_RS	0.05	0.2	4.1	0.14	0.2	15.7	0.31	0.3	184.0	0.35	0.6	-	0.43	3.1	-	0.48	20.5	-
Lottery_S	0.05	0.2	4.6	0.06	0.3	22.1	0.06	0.4	312.6	0.07	1.2	-	0.08	10.2	-	0.09	88.2	-
Random_S	0.05	0.1	2.9	0.06	0.2	18.8	0.08	0.3	290.8	0.09	1.3	-	0.10	10.2	-	0.11	88.9	-

$\mathcal{RC}$ – with the Range-Constraint Counting, $\mathcal{EC}$ – with the Enumerative Counting methods;

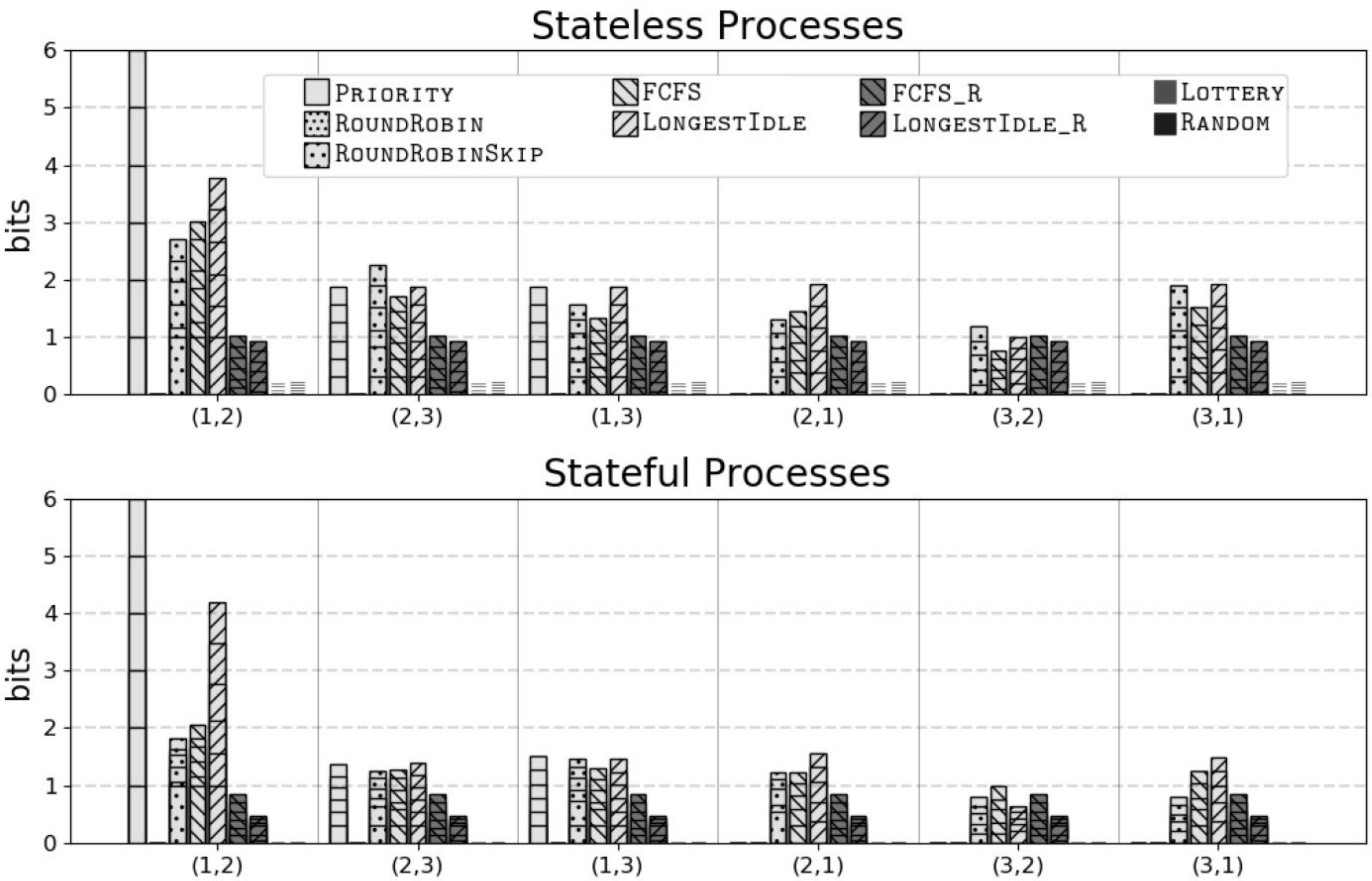
Timeout – 20 minutes (1200 s);

(S) – **stateful processes**; (R) – resolving wait-time and idle-time concurrences randomly.

Worst-case Leakage (*bits*) for **each protocol** as a function of the **round number**.



Leakage (*in bits*) for **each protocol** per **rounds 1-6**



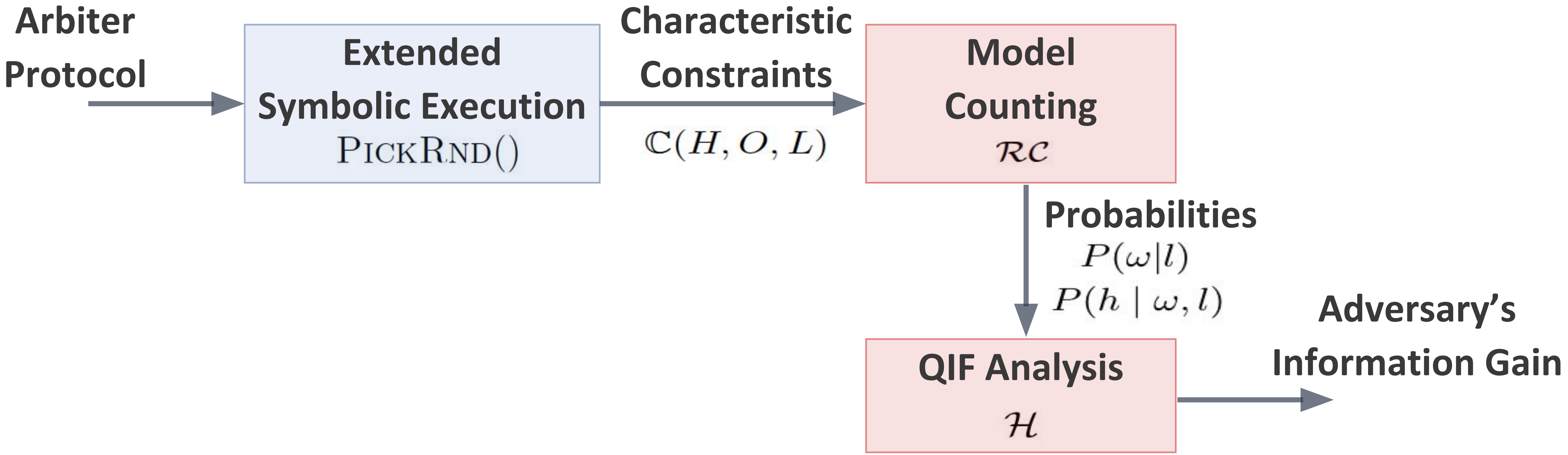
Leakage for each (victim; adversary) process pair.

Cumulative leakage is shown for 6 rounds.

Summary

A new approach for **automatically** **identifying** and **quantifying** the **information leakage** in protocols that **arbitrate** utilization of **shared resources** between processes.

Provides protocol designers and users a **new dimension** in **assessment** and **comparison** of protocols in terms of the **amount of information leaked over time**.



Summary

A new approach for **automatically** **identifying** and **quantifying** the **information leakage** in protocols that arbitrate utilization of shared resources between processes.

Provides protocol designers and users a **new dimension** in **assessment** and **comparison** of protocols in terms of the amount of information leaked over time.

The novel QIF analysis technique:

- **Combines** and **extends** **symbolic execution** and **model counting** techniques:
  - We extend **symbolic execution** to extract constraints characterizing relationships between the secret and the adversary-observable events.
  - With **model counting** constraint solvers, we quantify the amount of information leaked, in terms of entropy, by observable events:

    A **novel**, **efficient** and **exact model counting** technique for a class of constraints extracted during QIF analysis of arbiter protocols.
- **Supports randomized** protocols.