

# 08 – Timers & Think Time

---

## Objective

Understand how to make your Virtual Users behave like Real Humans, not robots. Learn why "Think Time" is critical for accurate capacity planning and avoiding false positives.

### What is "Think Time"?

Real users do not click buttons instantly.

- They read the page.
- They fill out a form.
- They scroll.
- **Then** they click "Submit".

The delay between two actions is called **Think Time**.

### Without Think Time (Robot Behavior)

- User 1 logs in → 0ms delay → Profile → 0ms delay → Submit.
- **Result:** The server gets hammered with unrealistic intensity.
- **Risk:** You overestimate the server load and crash the system artificially.

### With Think Time (Human Behavior)

- User 1 logs in → waits 3s → Profile → waits 5s → Submit.
- **Result:** Realistic concurrency and throughput.

---

## How JMeter Handles Delays (Timers)

In JMeter, **Timers** add delays *before* a sampler is executed.

### Important Rule of Scope:

If you place a Timer inside a Thread Group, it applies to **EVERY** sampler in that group.

**Visual Example:**

```
Thread Group
├─ Constant Timer (3000ms)
├─ Login API (waits 3s)
├─ Profile API (waits 3s)
└─ Submit API (waits 3s)
```

---

## 3 Common Types of Timers

### ◆ Constant Timer

- **Behavior:** Pauses for a fixed amount of time.
- **Use Case:** Debugging or very specific pacing requirements.
- **Example:** `3000 ms` (3 seconds).

### ◆ Uniform Random Timer (Recommended for Load)

- **Behavior:** Pauses for a random time within a range.
- **Why?** Real users don't all wait exactly 3.00 seconds. Some wait 2s, some 4s. This prevents "harmonic" spikes where all threads hit the server at the exact same millisecond.
- **Config:**
  - Constant Delay Offset: `2000`
  - Random Delay Maximum: `2000`
  - *Result: Delays between 2s and 4s.*

### ◆ Gaussian Random Timer

- **Behavior:** Uses a bell curve distribution (most users wait around the average, few wait very short or very long).
- **Use Case:** Highly realistic simulation.

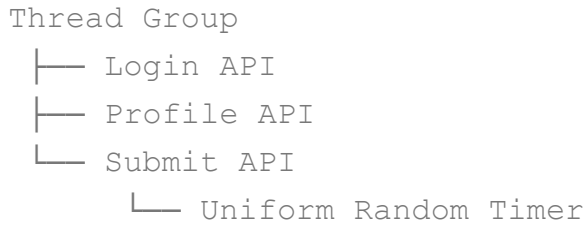
### ◆ Constant Throughput Timer (Pacing)

- **Behavior:** Controls the **Requests Per Minute (RPM)** rather than just pausing.
  - **Use Case:** "I want to force exactly 60 requests per minute."
  - **Note:** This is an advanced timer used for target-based testing.
- 

## 4 Where to Place Timers

## ✓ Best Practice: As a Child of a Sampler

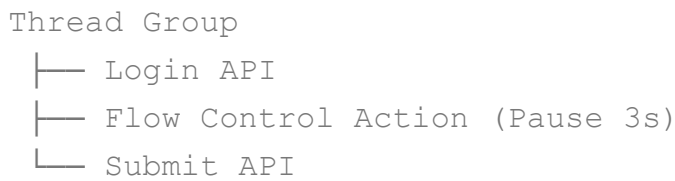
If you want a delay *only* before the "Submit" action:



Result: Login (instant) → Profile (instant) → Wait → Submit.

## ✓ Best Practice: Using "Flow Control Action"

You can add a **Flow Control Action** (previously called Test Action) sampler to insert a pause without attaching it to a specific request.



---

## 5 Real World Strategy

Test Type	Think Time Strategy
Smoke / Debug	<b>0 ms</b> (Disable timers to run fast)
Load Test	<b>2-5 sec</b> (Randomized to mimic humans)
Stress Test	<b>0-1 sec</b> (Minimize to crush the server)
Soak Test	<b>High</b> (Realistic daily usage)

---

## 6 Calculating Throughput (The Math)

If you have **100 Users**:

- **Scenario A (0s Think Time):**

- Transaction takes 1s.
- Throughput = 100 req/sec.
- **Scenario B (9s Think Time):**
- Transaction takes 1s + 9s wait = 10s total.
- Throughput = 100 users / 10s = **10 req/sec**.

👉 **Think Time dramatically affects the load you generate.**

---

## Interview Question

**"Why did your test pass with 1000 users, but production crashed with 1000 users?"**

**Answer:**

"Likely because the load test had **zero think time**, creating artificial throughput that didn't match real session duration, causing connection queuing issues differently than production. Or, the test had **too much think time**, and didn't generate enough Requests Per Second (RPS) to stress the DB."

---

## Mini Exercise

1. Add a **Uniform Random Timer** to your Thread Group.
2. Set it to random 1s-3s.
3. Run the test and observe the "Start Time" of requests in **View Results Tree**.
4. Notice they are no longer firing instantly.