

Implementing Soft Preemption for Audio Narrators

Setting the Stage: Context for the Curious Book Reader

This entry serves as a practical blueprint for handling audio interruptions in stream processing. It demonstrates how to shift from hard, destructive interruptions to a more graceful, 'soft' preemption method—a technique important to know in the Age of AI as we optimize how agents interact with the flow of information.

Technical Journal Entry Begins

 Verified Pipulate Commits:

- [8a3691b8](#) ([raw](#))
- [4a6b0953](#) ([raw](#))

MikeLev.in: That was a good job right there! That worked exactly as intended. Now do you notice how the station identification breaks immediately cut off the talking? Do you think they could let the sentence finish? I know wording this makes a lot of difference because we're looking for the easiest and safest to implement approach and not any major refactor. What is the lightest touch possible thing we can do to prevent that immediate cutting off the words?

Claude Opus 4.8 Max: Context verified. The change lives entirely in `stream.py`; `forest.py`, `score.py`, and `content_loader.py` are all present, and `queue` is already imported at the top. The cutoff has a one-line root cause. `run_station_break` opens with `narrator.interrupt()`, and `interrupt()` does *two* things: it drains the backlog **and** kills the audio playing right now (`p.kill()`). That second half is what clips the article mid-word. You don't want to weaken `interrupt()` itself — the standby and breaking-news paths call it too, and those *should* cut instantly (a deploy is happening; silence now). So the fix is to give the station break a gentler sibling and leave the urgent path alone.

Identifying the Clipping Root Cause

The lightest safe touch: add `flush_queue()` — literally `interrupt()` minus the kill block — and call it at the break lead-in instead. The in-progress line finishes, only the queued backlog is dropped, and the existing `queue.join()` already waits on the real voice clock so "We interrupt this program..." lands *after* the current sentence ends. (I picked a named sibling over an `interrupt(kill_current=False)` flag specifically so the urgent call sites stay byte-for-byte untouched — and as a bonus this finally makes the implementation match `run_station_break`'s own docstring, which already claims it merely "flushes the article backlog.")

Engineering the Soft Sibling

The one real risk was the `task_done` accounting: get that wrong and `queue.join()` either hangs or returns early. So I simulated the Narrator's queue lifecycle in isolation — in-progress line + backlog, then flush → say → join. Result: ['CURRENT', 'WE_INTERRUPT'] spoken in order, backlog dropped, nothing killed, join returned without deadlock. The behavior holds.

Validating Queue Integrity

Two surgical edits, both to `remotes/honeybot/scripts/stream.py`:

Target: `remotes/honeybot/scripts/stream.py`

```
[[[SEARCH]]]
    with self._proc_lock:
        for p in self._active_procs:
            try:
                p.kill()
            except Exception:
                pass
        self._active_procs = []

    def run(self):
[[[DIVIDER]]]
        with self._proc_lock:
            for p in self._active_procs:
                try:
                    p.kill()
                except Exception:
                    pass
            self._active_procs = []

    def flush_queue(self):
        """Soft sibling of interrupt(): drop the queued-but-unspoken
backlog
        but DO NOT kill the audio playing right now. The in-progress
line
        finishes cleanly, then the (now-empty) queue takes the next
thing to
        say. Used for station breaks, which should never clip a word;
the hard
        interrupt() stays reserved for genuinely urgent cuts
(deploy/breaking)."""
        try:
            while True:
                self.queue.get_nowait()
                self.queue.task_done()
        except queue.Empty:
            pass
```

```

    def run(self):
[[[REPLACE]]]

Target: remotes/honeybot/scripts/stream.py
[[[SEARCH]]]
    # --- Stage 1: The Interruption Sentinel ---
    narrator.interrupt()
    narrator.say("We interrupt this program for a station
identification break.")
    narrator.queue.join()
    time.sleep(1.2)
[[[DIVIDER]]]
    # --- Stage 1: The Soft Preemption ---
    # flush_queue (not interrupt) so the article sentence playing
RIGHT NOW
    # finishes instead of being clipped mid-word — only the queued
backlog is
    # dropped. The queue.join() below then rides the real voice clock,
so the
    # lead-in lands after the current line ends, not on top of it.
    narrator.flush_queue()
    narrator.say("We interrupt this program for a station
identification break.")
    narrator.queue.join()
    time.sleep(1.2)
[[[REPLACE]]]

```

One thing worth naming so it isn't a surprise on the next listen: the backlog *is* still dropped (same as before — interrupt() always drained it), so if the director thread has run a sentence or two ahead of the voice, the article skips those when the break lands. You'll now hear the current sentence land cleanly and then the break, rather than a hard clip — but the lead-in still won't be a seamless continuation of the article. If that backlog-skip ever bugs you, that's a separate, larger conversation about the two-clock desync, not this patch.

MikeLev.in: Okay, I understand the trade-off. To get a smooth interruption, a few sentences might be dropped between the cut-off and the article continuation. That's fine. That cut-off effect is too jarring and smoothing it out will be a worthwhile improvement. And this touches stream.py which only takes effect on an outerloop restart, so it's already a use-case for my --reboot flag on publish. However, I won't see (hear) it take effect until the next article push *AFTER* this one. C'est la vie.

The Deployment Workflow

```

$ git status
On branch main
Your branch is up to date with 'origin/main'.

```

```

nothing to commit, working tree clean
(nix) pipulate $ patch
(nix) pipulate $ cat patch | app
✓ DETERMINISTIC PATCH APPLIED: Successfully mutated
'remotes/honeybot/scripts/stream.py'.
(nix) pipulate $ d
diff --git a/remotes/honeybot/scripts/stream.py
b/remotes/honeybot/scripts/stream.py
index dc8a8cb6..78d601ab 100644
--- a/remotes/honeybot/scripts/stream.py
+++ b/remotes/honeybot/scripts/stream.py
@@ -201,6 +201,19 @@ class Narrator(threading.Thread):
         pass
         self._active_procs = []

+    def flush_queue(self):
+        """Soft sibling of interrupt(): drop the queued-but-unspoken
backlog
+        but DO NOT kill the audio playing right now. The in-progress
line
+        finishes cleanly, then the (now-empty) queue takes the next
thing to
+        say. Used for station breaks, which should never clip a word;
the hard
+        interrupt() stays reserved for genuinely urgent cuts
(deploy/breaking)."""
+        try:
+            while True:
+                self.queue.get_nowait()
+                self.queue.task_done()
+        except queue.Empty:
+            pass
+
     def run(self):
         while not self.stop_event.is_set():
             try:
(nix) pipulate $ m
📝 Committing: feat(honeybot): Implement flush_queue for stream
processing
[main 8a3691b8] feat(honeybot): Implement flush_queue for stream
processing
1 file changed, 13 insertions(+)
(nix) pipulate $ patch
(nix) pipulate $ cat patch | app
✓ DETERMINISTIC PATCH APPLIED: Successfully mutated
'remotes/honeybot/scripts/stream.py'.
(nix) pipulate $ d
diff --git a/remotes/honeybot/scripts/stream.py

```

```

b/remotes/honeybot/scripts/stream.py
index 78d601ab..0571e5bb 100644
--- a/remotes/honeybot/scripts/stream.py
+++ b/remotes/honeybot/scripts/stream.py
@@ -491,8 +491,12 @@ def run_station_break(env, profile_dir):
     bead = STATION_SEGMENTS[_station_index % len(STATION_SEGMENTS)]
     _station_index += 1

-    # --- Stage 1: The Interruption Sentinel ---
-    narrator.interrupt()
+    # --- Stage 1: The Soft Preemption ---
+    # flush_queue (not interrupt) so the article sentence playing
RIGHT NOW
+    # finishes instead of being clipped mid-word – only the queued
backlog is
+    # dropped. The queue.join() below then rides the real voice
clock, so the
+    # lead-in lands after the current line ends, not on top of it.
+    narrator.flush_queue()
    narrator.say("We interrupt this program for a station
identification break.")
    narrator.queue.join()
    time.sleep(1.2)
(nix) pipulate $ m
📝 Committing: fix: Refactor stream.py - Implement soft preemption
[main 4a6b0953] fix: Refactor stream.py - Implement soft preemption
1 file changed, 6 insertions(+), 2 deletions(-)
(nix) pipulate $ git push
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 48 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (12/12), 1.43 KiB | 1.43 MiB/s, done.
Total 12 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 4 local objects.
To github.com:pipulate/pipulate.git
e7ddcb6b..4a6b0953 main -> main
(nix) pipulate $

```

And now we turn this into an article and publish it with the --reboot flag. And then the article after this one, the next one I publish, I'll know if it actually worked, haha!

Book Analysis

Ai Editorial Take

What surprised me is the shift from treating the 'narrator' as a simple service to treating it as an asynchronous entity where queue management is now a primary UX factor. It highlights a transition in AI agent development: moving from 'does it work?' to 'how does it feel to listen to it?'

Content Potential And Polish

- **Core Strengths:**
- Clearly defines the trade-offs between 'hard' and 'soft' interrupts.
- Provides verifiable, step-by-step technical implementation.
- Transparently discusses the limitation regarding backlog skipping.
- **Suggestions For Polish:**
- Expand slightly on the 'two-clock desync' mentioned as a secondary issue.
- Ensure the distinction between the urgent and soft path is emphasized for future maintainers.

Next Step Prompts

- Analyze the current state of the two-clock desync and propose a strategy to sync the 'narrator' voice clock with the backlog processor.
- Draft a follow-up article detailing how to implement a more complex buffer management system that prevents article sentence skipping entirely.