3-Tier Server Demo Guide

This guide focuses on visibly demonstrating blocking vs concurrent server behavior.

One-command demo (recommended)

Run:

```
./scripts/demo_servers.sh
```

What it does: - Builds everything (`make -j4`) - Runs three experiments against each server variant (`sync`, `threaded`, `select`, `event`) - Prints timing summaries - Saves logs under `/tmp/3tier-demo`

What to point out during the demo

Experiment 1: Baseline

- Setup: one normal client, fast DB.
- Strong visual:
  - All variants are fast in uncontended conditions.
  - Useful as a control before introducing contention.

Experiment 2: Slow sender interference

- Setup: one client sends a valid request frame one byte at a time; then a normal client is started.
- Strong visual:
  - `sync`: normal client is delayed a lot (head-of-line blocking).
  - `threaded`, `select`, `event`: normal client remains fast.

Experiment 3: DB bottleneck control

- Setup: fixed DB delay with two simultaneous clients.
- Strong visual:
  - All variants look similarly slow because tier 3 is serialized.
  - Good teaching point: server concurrency cannot fix a downstream bottleneck.

Manual walkthrough (if you want to narrate each step)

Use separate terminals.

1. Build: `bash make -j4`

2. Start DB: `bash ./db/db --port 24100`

3. Start one server (change binary per variant): bash ./server_sync/server –port 24000 –db-port 24100

   ## or: ./server_threaded/server –port 24000 –db-port 24100

   ## or: ./server_select/server –port 24000 –db-port 24100

   ## or: ./server_event/server –port 24000 –db-port 24100

4. Baseline run: `bash ./client/client --port 24000 --n 1`

5. For head-of-line blocking: `bash ./scripts/demo_servers.sh `The script injects a byte-drip slow sender and measures how long a normal client waits.

6. For DB bottleneck behavior, run the same script and look at `db_bottleneck_total_ms`.

Optional knobs

Override defaults when running the script:

```
DB_DELAY_MS=500 DRIP_BYTE_DELAY_MS=100 ./scripts/demo_servers.sh
```

- `DB_DELAY_MS`: controls severity of the downstream bottleneck experiment.
- `DRIP_BYTE_DELAY_MS`: controls how aggressively the slow sender stalls a blocking server.