EdgeBeacon: A Minimal Windows Batch Utility for Instant Service Availability Detection

Sudipta Kumar Das

Version 1.0.0

Abstract — We present *EdgeBeacon*, a minimal batch script that probes a target URL at $\sim$ 100ms intervals using Windows-bundled `curl`. Upon receiving HTTP 200 or any 3xx, it launches Microsoft Edge with the target URL. The utility is designed for day-to-day operational readiness checks (e.g., university portals) with zero install overhead and open access reproducibility.

Ask: 6–8 week RA support to harden, sign, and pilot *EdgeBeacon* campus-wide.

1 Introduction & Background

University *Open Credit Course* registrations rarely go live at the exact publicly announced time. In practice, portals are throttled or temporarily held back to prevent crashes under peak load; Single Sign-On (SSO) redirects are common; and students end up waiting in front of a PC, repeatedly pressing *Refresh* for minutes or even hours to learn when the server finally becomes reachable. This is stressful, wastes time, and still risks missing the first come–first served window.

Goal. Replace manual, error-prone refreshing with a small, dependable tool that detects actual availability *as soon as it happens* and then opens the portal instantly in the default (Edge) browser.

Motivating scenario

Admins may say "`12:00 AM`", yet the portal truly accepts requests at, say, `1:28 AM`. Students keep a tab open and refresh continuously to avoid missing the opening minute. **EdgeBeacon** automates this waiting: it probes the target URL at $\sim$ 100 ms intervals; the instant the server returns `200` or any `3xx` (common for SSO hand-offs), EdgeBeacon launches Microsoft Edge directly to the portal page.

2 Problem Statement & Objectives

Problem. Manual polling creates inequity (only the most persistent or lucky get through first), increases cognitive load late at night, and still fails if the opening moment is missed by seconds. **Objectives.**

- Hands-free detection of real server availability (HTTP `200`/`3xx`).
- Zero install overhead (Windows built-ins only), suitable for any lab/hostel PC.
- Clear console feedback (attempt count, timestamps, status codes).
- Open, reproducible implementation and documentation for audit and reuse.

3 Contributions

- **Practical automation:** A minimal batch utility that monitors availability and triggers Edge at the exact go-live moment.
- **Redirect-aware:** Treats `3xx` as UP, matching real portals that bounce through SSO.
- **Zero-dependency:** Uses Windows' built-in `curl`; no external installer or admin rights.
- **Open science:** Public repository, MIT license, `CITATION.cff`, LaTeX tech note, and versioned releases for replicability.

4 Expected Impact

For students, it eliminates repetitive refreshing, reduces late-night fatigue, and decreases missed-slot anxiety; the moment the site is live, the browser opens and students act. **For administrators.** Encourages gentler traffic patterns than manual "refresh storms," while remaining fully compliant with existing authentication (no bypassing). **For teaching/research.** A concrete example of using system scripting to solve a day-to-day operational bottleneck with rigorous, open documentation.

5 Evaluation Plan

We will report:

- **Detection latency:** time from first `200/3xx` to Edge launch.
- **Resource footprint:** CPU/network impact across intervals (e.g., 50–500 ms).
- **Robustness:** behavior across common failure modes (DNS fail, TLS errors, `000` connection refused).
- **Usability notes:** student feedback from a mock registration window.

Success metrics:

- Average time to open (ms) from first 200/3xx to Edge launch
- Pilot usage & short student feedback
- Fewer help-desk "refresh storm" tickets in reg windows

6 Roadmap

- Configurable interval & polite backoff profiles (admin-friendly).
- Optional PowerShell GUI wrapper (progress UI, start/stop).
- Signed portable EXE build and checksums in Releases.
- Accessibility polish (high-contrast console, larger fonts).

7 Method

We sample the target URL with Windows `curl` and inspect only the HTTP code:

curl −s −o NUL −w "%{http_code}" <URL>

Codes **200** or any **3xx** are treated as UP (covers SSO redirects). Between probes we sleep briefly to avoid CPU saturation:

ping 127.0.0.1 −n 1 −w <ms> >NUL

Pseudocode (single-threaded probe)

```
Input: URL u, interval Dt (ms), Edge path E
repeat
c := http_code(u) // curl −s −o NUL −w "%{http_code}"
if c == 200 or (300 <= c && c < 400) then
launch_edge(E, u) // start "" "%E%" "%u%"
exit
end if
sleep(Dt) // ping 127.0.0.1 −n 1 −w Dt >NUL
forever
```

Deterministic bounds

Let the probe period be Δt ms and the portal goes live at time T.

- Sampling rate $f = \frac{1000}{\Delta t}$ Hz; requests per second $r = f$.
- Worst-case detection delay $< \Delta t$; if the go-live time is uniform within a sampling bin, $\mathbb{E}[\text{delay}] \approx \frac{\Delta t}{2}$.
- Probes issued by time T: $N = \lceil T/\Delta t \rceil$.

8 Usage

Double-click the batch file, paste the URL, and wait. The console prints attempts and HTTP codes; upon UP, Edge launches immediately.

8.1 Simple math view

We check the website every Δt milliseconds (for example, $\Delta t = 100$ ms $= 0.1$ s).

- **Worst extra wait after the site goes live:** $< \Delta t$.
- **Average (typical) extra wait:** $\approx \Delta t/2$.

Example. If $\Delta t = 100$ ms, the longest extra wait is under 100 ms, and on average it is about 50 ms.

9 Limitations

Aggressive intervals can increase network/CPU load. For captive portals that block ICMP/HTTP until authentication, behavior may vary.

9.1 Quantified load

Rate $r = 1000/\Delta t$ checks/s; data $\approx r\,h$ bytes/s for header size h. *Eg.* $\Delta t = 200$ ms $\Rightarrow r = 5$; if $h \approx 800$ B, data ~ 4 KB/s.

10 Ethical/Operational Notes

Use against targets you own or are authorized to probe. The script performs benign HTTP status checks only.

10.1 Operational rules (easy)

Use only on allowed sites. Respect a site cap $r_{\max}$ by choosing $\Delta t \geq 1000/r_{\max}$ ms (e.g., $r_{\max} = 5 \Rightarrow \Delta t \geq 200$ ms). Tool only reads HTTP status; no auth bypass.

11 Funding Rationale & Deliverables

Ask: 6–8 week RA support (USD \$600–\$1,200) to harden and pilot *EdgeBeacon* campus-wide. **Why:** reduces late-night refresh load; last-mile work (signing, guide, eval) enables safe rollout.

D1. Evaluation kit + report: latency curve; CPU/network vs. interval (scripts included).

D2. GUI wrapper; **signed** portable EXE; checksum + release notes.

D3. Admin guide: recommended intervals/rate caps; deployment steps.

D4. Reproducibility bundle archived with DOI (code, data, LaTeX PDF).

Timeline: W1–2 D1, W3–4 D2, W5 D3, W6 D4. **Success:** signed public release + admin guide used in next registration window.

Institutional fit: reduces late-night refresh load, respects auth, and deploys on lab PCs with zero install.