Computer Communication Workshop: DNY - Dynamic Domain Name for You:

Abstract Idea Description

1. **Team members:** Dana Haham, Noa Pinsler, Yarin Baslo.
2. **The Problem:** In many regions, access to specific websites and online services is restricted due to governmental censorship, ISP-level blocking, or geographical content restrictions. These limitations can hinder user productivity, access to knowledge, and internet freedom. Additionally, DNS servers can sometimes experience downtime or overload, resulting in widespread connectivity issues and leaving many systems unable to resolve domain names properly. Most users are unaware of DNS or how changing it can help bypass these restrictions. Even if they are, manually configuring DNS is time-consuming and technical, creating a barrier to usability.
3. **Idea:** A tool that dynamically changes the DNS for a specific request or session to provide seamless access to websites blocked or restricted by their current DNS.
4. **Added Value:**
   a. **Precision and Non-Intrusiveness:** Dynamically switches DNS only for the specific blocked request or session, ensuring the user's global DNS settings remain untouched, preserving their preferred speed, security, and reliability for all other activities.
   b. **Fully Automated Detection:** Automatically detects when a website or service is inaccessible due to DNS-related issues and initiates the DNS switch without any manual input from the user.
   c. **Zero Technical Knowledge Required:** Requires no expertise in networking or DNS management, providing an easy and seamless experience for everyone.
   d. **Cross-Platform Compatibility:** The tool is compatible across various platforms and devices, making it a versatile solution for users on different operating systems and environments
5. **Target Market:**
   a. **General Internet Users:** Non-technical individuals who need seamless access to websites blocked by DNS issues.
   b. **People in Authoritarian Regimes:** Individuals living under restrictive governments who face censorship and need tools to bypass blocked websites and access unrestricted information.
   c. **IT Professionals and System Administrators:** Experienced individuals who manage and configure DNS settings across networks and devices, seeking an automated solution to streamline DNS management and ensure uninterrupted access for users.

6. **Strengths:**
    a. **Automation**: Eliminates the need for manual DNS configuration, making it highly user-friendly.
    b. **Accessibility**: Simplifies access to restricted websites without requiring technical expertise.
    c. **Convenience**: Operates seamlessly in the background, providing a hassle-free experience.
    d. **Cross-Platform Potential**: Can be developed to support various operating systems like Windows, macOS, and Linux.
    e. **Broad Appeal**: Useful for a wide range of users, including casual internet users, remote workers, and travelers.
7. **Weaknesses:**
    a. **Limited Scope**: Focuses only on DNS-related blocks and won't work for more advanced restrictions like IP-based or deep packet inspection.
    b. **Technical Challenges**: interacting with the network layer of the operating system, which requires precise handling of system-level APIs, permissions, and network configurations.
    c. **Dependency on DNS Servers**: Effectiveness relies on the availability and performance of external DNS servers like Google DNS or Cloudflare.
    d. **Potential User Confusion**: Users may not understand the tool's function or why their DNS settings are being changed.
8. **Competition:**
    a. **Manual DNS Switching:** Users can manually change DNS settings on their devices, but this is time-consuming and requires technical knowledge.
    b. **Browser Extensions:** Extensions like those supporting DNS-over-HTTPS (DoH) can resolve DNS issues for web browsers but lack system-wide coverage.
    c. **VPN Services:** VPNs can bypass geo-restrictions by rerouting traffic, but they are more complex, expensive, and may slow down the connection.
    d. **Third-Party DNS Changers:** Existing tools may offer DNS switching but lack automation triggered by website accessibility issues.
    e. **Our tool differentiates itself by** automating the process based on user activity and requiring no manual intervention.
9. **Main Features:**
    a. **Automatic Detection**: Recognizes when a website is inaccessible due to DNS issues.
    b. **Dynamic Switching**: Automatically changes the DNS to an alternative, more accessible server.

c. **User Feedback**: Notifies users about the DNS change and its success.

d. **Ethical Content Check:** Utilizes AI to analyze the website the user is trying to access and provides a warning if the site is identified as unethical, harmful, or potentially unsafe.

10. **User Flow:**

    a. The user tries to access a blocked website.

    b. The tool detects the DNS resolution failure and switches to a preconfigured or tested DNS server.

    c. The user gains access to the website without needing to take additional steps.

    d. The tool logs the activity and reverts to the original DNS attest.

11. **Dependencies:**

    a. **Public DNS Servers**: Requires access to alternative DNS providers like Google DNS, Cloudflare DNS, or OpenDNS.

    b. **System-Level APIs**: Uses operating system APIs to change DNS settings dynamically.

    c. **Internet Connectivity**: Relies on an active internet connection to test and switch DNS servers.

High Level Architecture

1. **System Main Components:** The DNS Interception Tool consists of several key components working together to enable seamless, dynamic DNS resolution that circumvents blocked websites. The main components of the system are as follows:
   a. **Local DNS Server:** Acts as a local DNS server running on the user's device. It intercepts all outgoing DNS requests, determines whether the domain is accessible, and dynamically reroutes blocked requests to an external DNS provider when necessary.
      Responsibility**:**
      1) Handle all DNS requests from the user's system.
      2) Identify whether a domain is blocked by analyzing DNS responses or patterns.
      3) Redirect blocked domains to an alternative DNS provider.
      4) Respond to the user's system with appropriate results.

   b. **DNS Blocking Detection Mechanism:** Identifies whether the requested domain is blocked based on DNS response codes or by matching the resolved IP address against known censorship patterns.
      Responsibility**:**
      1) Monitor DNS response data for indications of blocks or failures.
      2) Analyze patterns or IP addresses returned for blocked domains.
      3) Automatically trigger the DNS switching mechanism when a block is detected.

   c. **External DNS Provider:** Acts as a fallback for blocked DNS requests. It uses third-party DNS servers to resolve requests that the local DNS provider fails to handle.
      Responsibility:
      1) Provide DNS resolution for domains that are blocked or inaccessible via the user's default DNS provider.

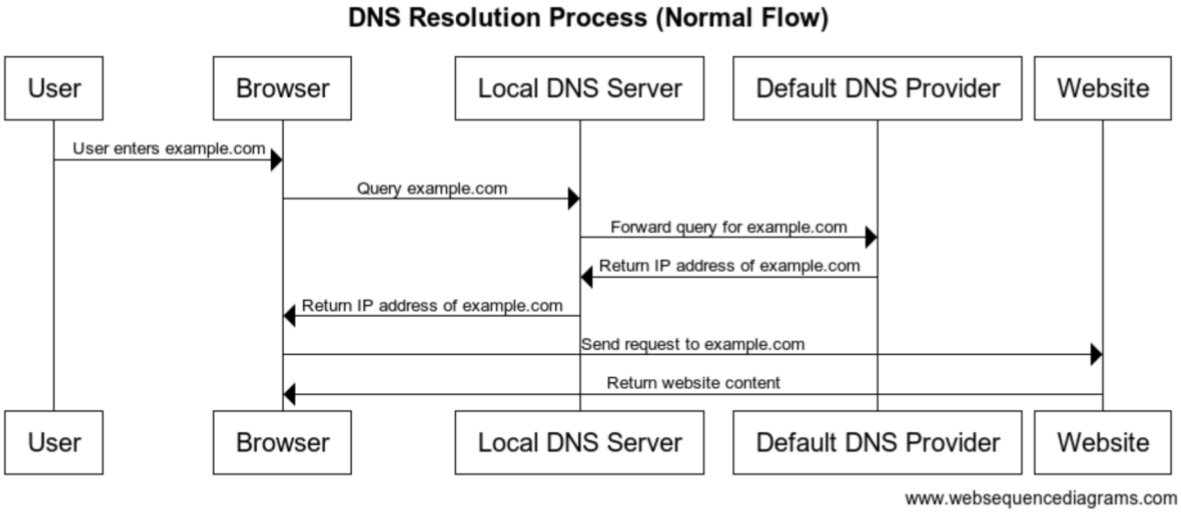
   d. **Ethical Content Check:** Analyzes the content of the requested domain to ensure it adheres to ethical standards and is not harmful, malicious, or otherwise unsafe.
      Responsibility**:**
      1) Use AI-based or external APIs to classify the website as safe, harmful, or unethical.
      2) Warn the user if the domain falls into a high-risk or unethical category, providing options to proceed or block

   e. **Notification:** Notifies users of DNS resolution activity, including when a domain is blocked, resolved using an external DNS provider, or flagged for ethical concerns.
      Responsibility**:**

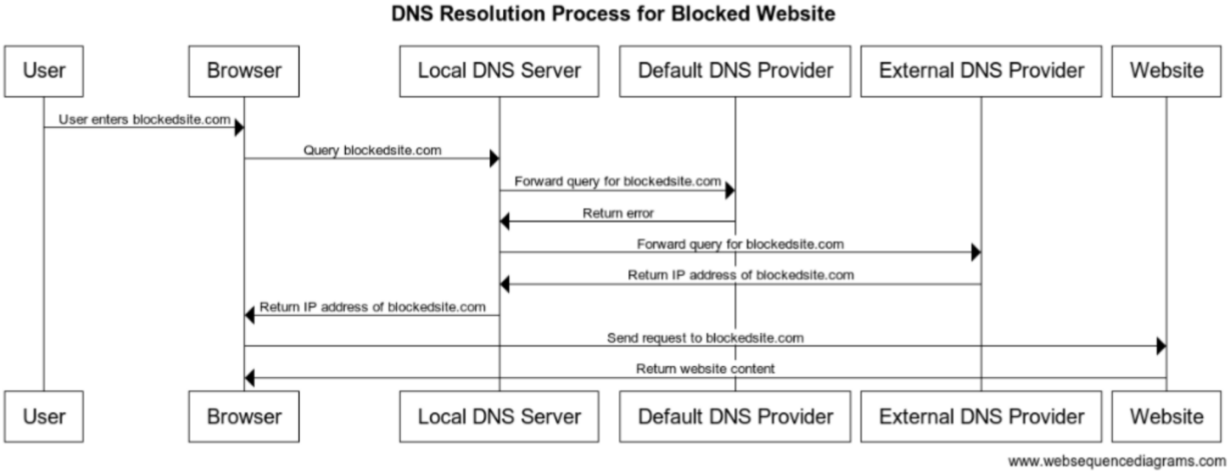
1) Display pop-up or tray notifications for the following events:
    a) A domain is detected as blocked.
    b) DNS resolution is successfully bypassed using an external DNS.
    c) A site is flagged as unsafe or unethical after ethical content analysis.
2) Offer clear actions to the user, such as "Proceed Anyway" or "Block Site."

2. <u>**Main User's Use Cases:**</u>

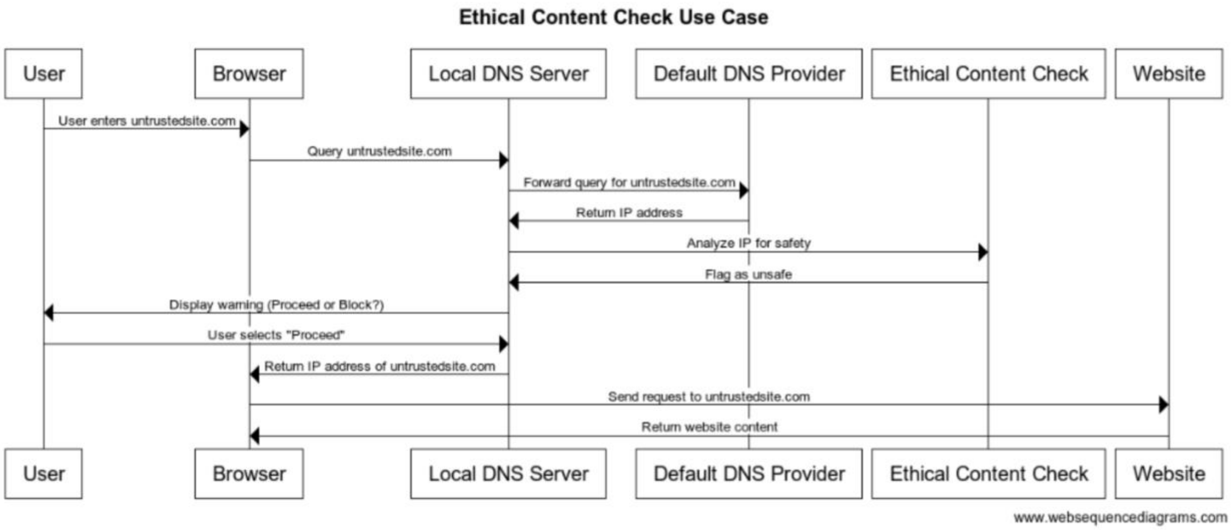
a. **Use Case 1 - Normal Domain Access (No Block):**


DNS Resolution Process (Normal Flow)
User
Browser
Local DNS Server
Default DNS Provider
Website
User enters example.com
Query example.com
Forward query for example.com
Return IP address of example.com
Return IP address of example.com
Send request to example.com
Return website content
User
Browser
Local DNS Server
Default DNS Provider
Website
www.websequencediagrams.com

b. **Use Case 2 - Blocked Domain Detection and DNS Switch:**


DNS Resolution Process for Blocked Website
User
Browser
Local DNS Server
Default DNS Provider
External DNS Provider
Website
User enters blockedsite.com
Query blockedsite.com
Forward query for blockedsite.com
Return error
Forward query for blockedsite.com
Return IP address of blockedsite.com
Return IP address of blockedsite.com
Send request to blockedsite.com
Return website content
User
Browser
Local DNS Server
Default DNS Provider
External DNS Provider
Website
www.websequencediagrams.com

c. **Use Case 3 - Unsafe or Unethical Website Detection:**



3. **<u>Back - End Technology:</u>** The local DNS server will be implemented using a lightweight framework that can handle DNS requests and responses efficiently. This could be built in the following technologies:
   a. **Python (dnslib library):**
      1) Python's dnslib library for creating a custom DNS server.
      2) It provides a framework to listen for DNS queries, send requests to other DNS servers, and return responses.
      3) You can implement the detection logic for blocked domains, and redirect to external DNS servers when necessary.

   b. **Node.js (dns module):**
      1) Using Node.js with the built-in dns module for managing DNS queries.
      2) Node.js is lightweight, efficient, and supports asynchronous operations, which makes it ideal for handling DNS requests.
4. **<u>Communication Protocols:</u>** DNS typically uses UDP for its communication, especially on port 53. We will handle DNS requests over UDP to ensure compatibility with existing DNS infrastructure.
5. **<u>Front-End Technologies:</u>**
   a. **Electron.js** is a framework that allows you to build desktop applications using web technologies (HTML, CSS, and JavaScript). This could be used to create a native application.