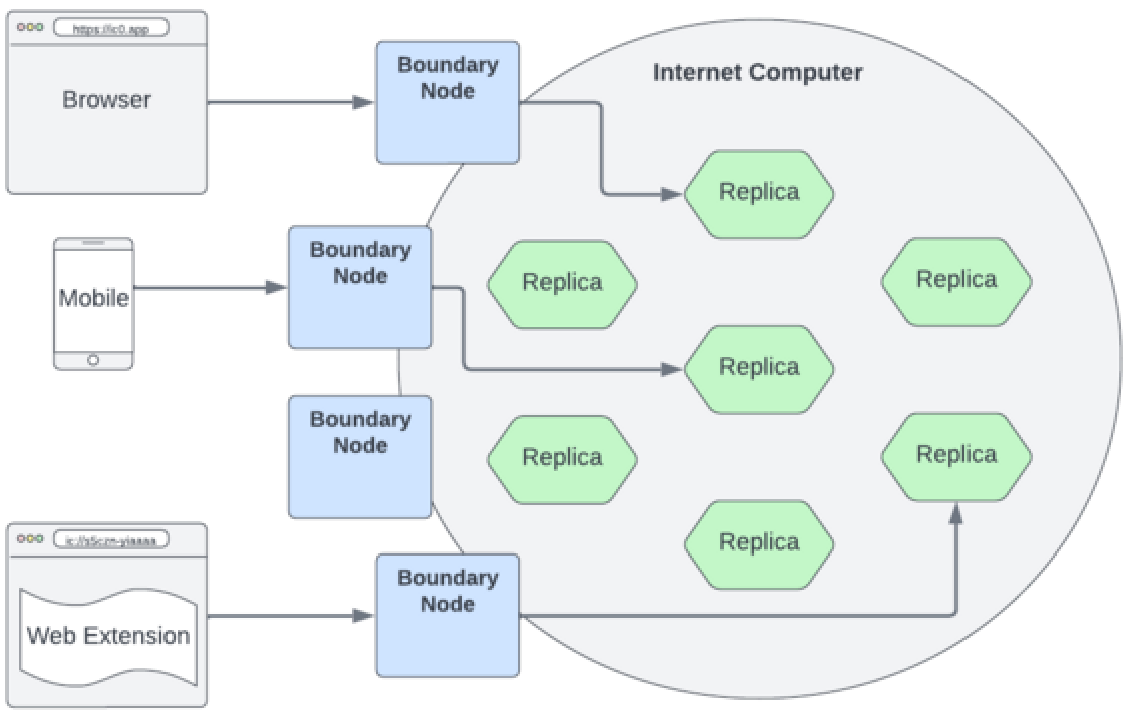
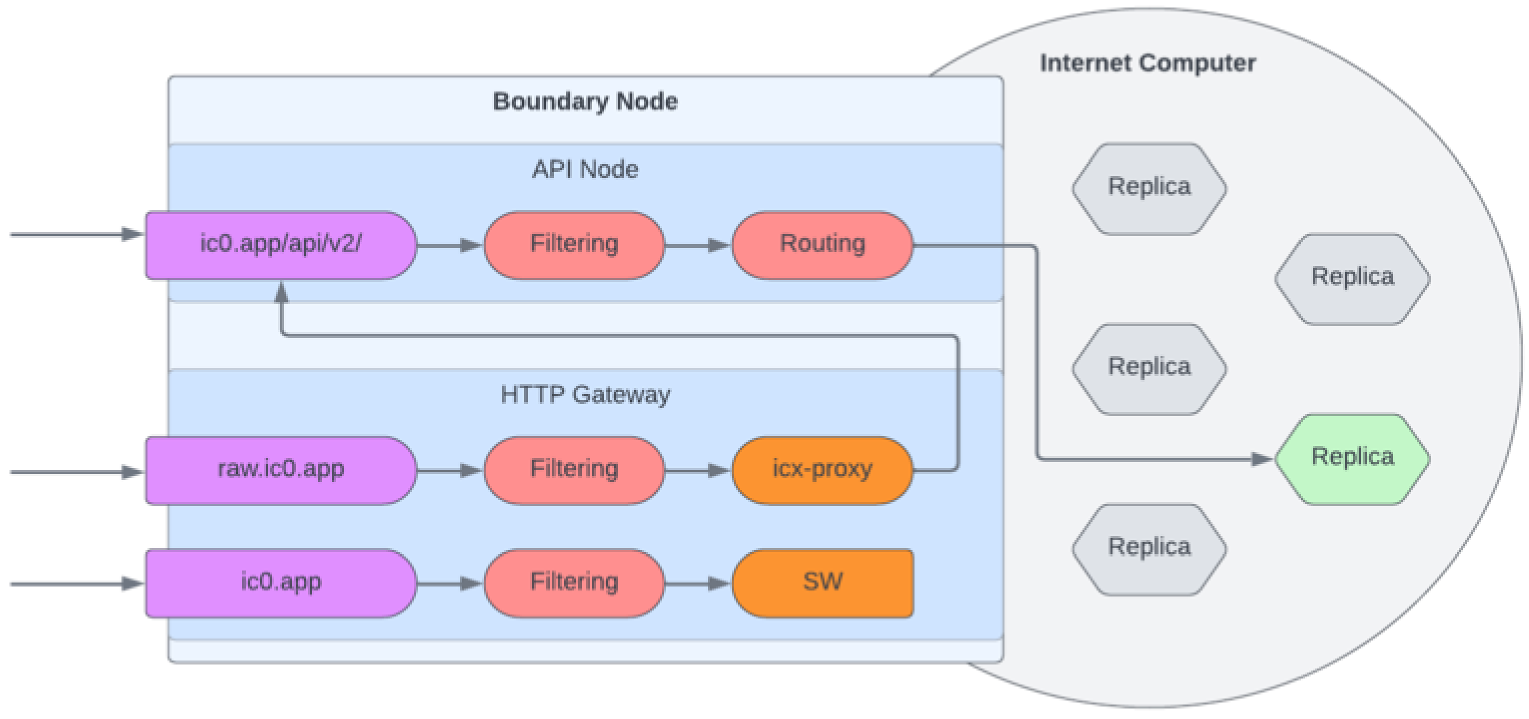
Boundary Nodes

The boundary nodes are the gateway to the Internet Computer (IC), and allow users to seamlessly access the canister smart contracts running on it. The following figure shows how the boundary nodes form the edge of the IC and all accesses to the IC have to go through one of the boundary nodes.



Boundary Node Internals

At a closer look, boundary nodes consist of two parts: the API node, which provides an endpoint for API canister calls (https://internetcomputer.org/docs/current/references/ic-interface-spec/#http-interface), and the HTTP gateway, which provides an HTTP endpoint for users to access the canisters hosted on the IC with their stock browser.



API Node

The API endpoint resides at `icp-api.io/api/v2` and is specified in the IC’s Interface Specification (https://internetcomputer.org/docs/current/references/ic-interface-spec/#http-interface).

Whenever a boundary node receives an API canister call, it passes it through a filter and then routes it to a replica node on the correct subnet in the IC.

Filtering within the boundary node consists only of rate-limiting. The rate-limits are in place to protect the IC from being overwhelmed with external accesses.

After an API canister call passed the filtering stage, the boundary node infers the destination canister ID and uses the routing table to look up the subnet in which this canister is hosted. It then randomly chooses a replica within that subnet to which it forwards the API call. The random selection of the target replica ensures that an API call eventually reaches an honest node when the client keeps retrying.

Finally, the boundary node forwards the API canister call to the selected replica node in the core of the IC.

Since the API Node is simply passing the API canister call on to the IC, no trust is required.

HTTP Gateway

The HTTP endpoint is served through two main domains: `ic0.app` and `icp0.io`. In the following, we use for simplicity only `ic0.app`, even though both domains could be used equally.

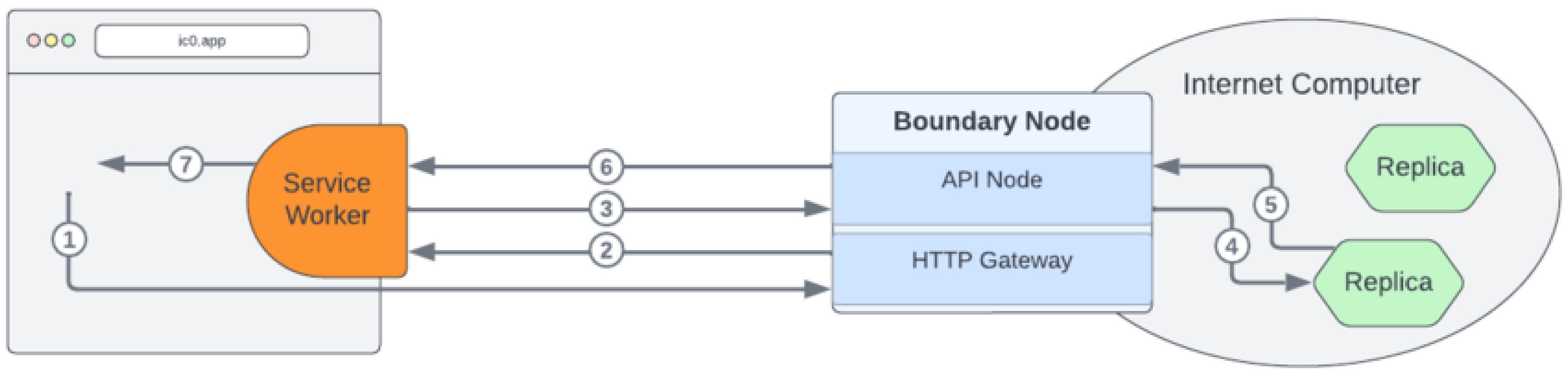
Unfortunately, browsers don't natively support API canister calls and therefore cannot directly talk to the canisters hosted on the IC. The HTTP gateway protocol (https://internetcomputer.org/docs/current/references/ic-interface-spec/#http-gateway) bridges that gap by providing a mechanism to translate HTTP requests into API canister calls allowing a client to interact with canisters. A gateway can be implemented in various forms (e.g., as a stand-alone proxy, as a browser plugin, or as a service worker (https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API)).

The boundary nodes provide two different implementations of the HTTP gateway protocol:

- Service worker: under `<canister_id>.ic0.app`, the boundary nodes serve a service worker (https://www.npmjs.com/package/@dfinity/service-worker), which is installed in the browser and acts as an HTTP gateway directly in the user's browser;
- `icx-proxy`: under `<canister_id>.raw.ic0.app`, the boundary node runs `icx-proxy` (https://github.com/dfinity/icx-proxy), an HTTP gateway implementation suitable for all clients that do not support a service worker.

Before serving any HTTP request, the HTTP Gateway passes the incoming request through a filter, which can be used to implement content filtering allowing operators to comply with local legal frameworks (e.g., blocking gambling services in a particular geography). To learn more about content filtering see Content Filtering via The Boundary Nodes.

Service Worker



When accessing `<canister_id>.ic0.app`, the boundary node returns a service worker implementing the HTTP gateway protocol, which is installed directly in the user's browser (step 1 and 2). From then on, the service worker will intercept all HTTP requests and translate them to API canister calls (step 3). These API canister calls will then go through the API endpoint of the boundary node to the IC (step 4). The replica sends the response back through the boundary node to the service worker (step 5 and 6). For all responses, the service worker verifies the certificate of the response and only translates it into a proper HTTP response for the browser if it passes all the checks (step 7).

`icx-proxy`

The HTTP Gateway endpoint implements the HTTP gateway protocol, which translates between HTTP requests and API canister calls. This endpoint resides at `<canister_id>.raw.ic0.app`. Whenever a boundary node receives such a request, it forwards it to `icx-proxy`, a service running directly on the boundary node that implements the HTTP gateway protocol. `icx-proxy` translates the HTTP requests into API canister calls and forwards them to the API endpoint of the boundary node. It verifies the certificates of the responses and constructs an HTTP response to send back to the client. Here, the user needs to trust the boundary node as the boundary node is constructing the API calls and verifying the correctness of the IC’s response.

Additional Features of the Boundary Nodes

Globally-Distributed

The boundary nodes serving `ic0.app` are globally distributed and organized in regional pools. All requests are directed to the geographically closest pool and load balanced over the instances within that pool. The health of the boundary nodes is constantly monitored and in case of failure, boundary nodes will be removed from the pools.

SEO

Bots and crawlers, such as the ones used by search engines, do not support service workers. In order for them to index content hosted under `ic0.app` their requests are internally redirected to `icx-proxy`. This allows the dapps running on the Internet Computer to seamlessly integrate into the Web 2.0 world. These dapps can be indexed by search engines and their metadata can be read in order to generate previews and cards on social platforms.

Caching

To improve the user-perceived performance of the dapps hosted on the IC, the boundary nodes currently provide response caching. Responses to requests are cached for 1s.

Future Boundary Node Developments

To follow future boundary node developments check out the public roadmap (https://internetcomputer.org/roadmap/), the IC developer forum (https://forum.dfinity.org/) and the thread on the future boundary node architecture (https://forum.dfinity.org/t/boundary-node-roadmap/15562).

See Also

- **The Internet Computer project website (hosted on the IC): internetcomputer.org (https://internetcomputer.org/)**
- Content Filtering via Boundary Nodes