Current state

- Web security depends on TLS connections
- Service Workers aim to support transitions from online to offline
  - Lots of push messaging, not as much offline content
- Two main drivers for real offline solutions
  - Lots of people who aren’t online much
  - Interest in new content delivery methods
Basic problem

- User finds USB drive in car park
- User plugs said USB drive into their computer
- Content arrives by something other than TLS
- Content needs to be usable
- User later goes online
- Content needs to be more usable after
The state problem

- The Web is a communications medium
- So assume that use of the Web offline means someone wants to communicate later
- Typically state about what happened is saved
- When someone goes online, that state has to be available for use
Challenge

- Authority on the Web is based on connections
- If someone can’t or won’t connect, how do we enable the full experience?
Necessary sacrifices

◦ What things do we have to lose?
  ◦ updates to server state
  ◦ communication with others
  ◦ real-time bidding for advertising
  ◦ tracking of user activity

◦ What else can we afford to sacrifice?
  ◦ This is a much harder question to answer
Option 1: Take Web origins offline

- In short, don’t sign connections, sign content
- A bundling format is critical
  - It’s largely uncontroversial, even good
  - It’s just an XKCD 927 problem
- Just sign the bundle… right?

https://example.com
Limitations

- It is hard to know what is safe to sign
- Potential weakening of the basis of authority
  - DNS lookups are seen as a weak second factor
- Revocation status cannot be communicated
  - Over-signing, compromise, or certificate miss-issuance all lead to a need to revoke
  - Bugs are exposed to exploitation by attackers
  - Content has a limited shelf-life to compensate
- A bunch of other minor issues
Option 2 (Proposal): Give content its own origin

- State for bundled content is saved in a store that is specific to that bundle
  - The identity of that origin can be meaningless
- A bundle can identify a target origin
- The target origin can accept the bundle
  - Content and state is transferred if successful
  - Origin aliases provide additional continuity
- A transfer can be rejected by a site
Offline Usage

- A bundle is given a new type of origin
  - ni:///sha-256;ypeBEsobvcr6wjGzmiPcTaeG7_gUfE5yuYB3ha_uSLs
- The browser treats this like any other origin
- Content can make HTTP requests
  (though these are unlikely to work if truly offline)
Transfer

- The bundle can designate a target URL
- The bundle requests a transfer to that URL
- The browser fetches the URL with a challenge
- If the site answers the challenge correctly…
  - Navigation to the target URL happens
  - State is transferred to the target origin
  - The content origin is aliased to the target origin
  - Content from the bundle can be used in place of making requests (performance gain)
Origin Aliasing

- New concept
- After transfer, the content origin becomes an alias for the target origin
- Messages sent to the content origin can be received by the target origin
Failed Transfer

- **A failed** transfer keeps the content origin
  - HTTP 503, connection failures, being offline still
- **A rejected** transfer is when the server fails the challenge sent by the browser
  - Manifests as a navigation to the target URL
  - No continuity
  - Navigation information passing options only: URL and maybe Referer
  - Useful if server believes content is somehow bad
Limitations

- Content can’t be attributed to its target origin
  - Content has a “potential” origin
  - This is really hard to explain
- Transition to online takes 1 round trip
- State transfer is non-trivial
  - One origin could have multiple bundles
  - Even 1:1 transfer is likely technically challenging
- Likely a bunch of minor issues
AMP usage

- AMP delivers content to an online recipient
  - The recipient is effectively offline by choice
- AMP is an offline case for a very short time
- Transfer happens immediately
  - State is likely zero
  - State is only created in case of a failed transfer
- This case is likely much easier to handle
THANK YOU
## Backup: Comparison

<table>
<thead>
<tr>
<th>Signed Exchanges</th>
<th>Content Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>◦ Requires a bundle format that includes signatures</td>
<td>◦ Requires a bundle format</td>
</tr>
<tr>
<td>◦ Decision about continuity made up front</td>
<td>◦ Decision about continuity made afterwards</td>
</tr>
<tr>
<td>◦ Limitations on what can be signed</td>
<td>◦ Potentially tricky transfers</td>
</tr>
<tr>
<td>◦ Time limited usage</td>
<td>◦ And maybe state merges</td>
</tr>
<tr>
<td>◦ Immediate transition</td>
<td>◦ Limitations on what can be signed</td>
</tr>
<tr>
<td></td>
<td>◦ No(fewer?) usage limitations</td>
</tr>
<tr>
<td></td>
<td>◦ Transition requires a request</td>
</tr>
<tr>
<td></td>
<td>◦ Possibly strange UX</td>
</tr>
</tbody>
</table>