Emerald Image Mesh Flow

Participants: User, Web UI, Provider Node, Discovery Cache, iroh Network, Peer Node

User → Web UI: Select image & click upload

Web UI → Provider Node: POST /upload (multipart file)

Provider Node → Provider Node: Store blob in iroh-blobs & export current.img

Provider Node → Web UI: 200 OK (hash, ticket, metadata)

Web UI → Web UI: Set current upload context & checklist state

Provider Node → Discovery Cache: Update NodeState with hash & provider node_id

Discovery Cache ⇢ Provider Node: Cached peer NodeAddrs (via peer_addr_refresher)

alt [P2P notify succeeds]

Provider Node → iroh Network: Notify peers (NotifyMsg)

iroh Network ⇢ Peer Node: Notify (hash, filename, provider id)

[NodeAddr missing or notify fails]

Provider Node → Peer Node: HTTP POST /receive (hash fallback)

loop [For each peer]

Peer Node → Peer Node: Reset NodeState (has_image=false, progress=0)

Peer Node → Discovery Cache: Poll provider /status until NodeAddr known

Discovery Cache ⇢ Peer Node: Provider NodeAddr cached

Peer Node → iroh Network: downloader.download(hash, provider_id)

iroh Network ⇢ Provider Node: Request blob slices over iroh-blobs

Provider Node ⇢ Peer Node: Blob parts streamed (Progress events)

Peer Node → Peer Node: Update bytes_received & progress from events

Peer Node → Peer Node: Export blob to current.img and set has_image=true

Peer Node → Discovery Cache: /status reports completion

Web UI → User: Update graph & previews with live progress

Note: iroh-blobs can stripe data across providers (SplitStrategy), but the demo pins each download to the discovered provider.