Architecture of the Upload Flow
Overview

- PPM is three "sub-protocols" executed simultaneously
  - *Upload Flow* – Client pushes report (encrypted input shares) to the Leader
  - *Aggregate Flow* – Leader and Helper(s) interact to verify and aggregate reports and compute aggregate shares
  - *Collect Flow* – Collector pulls encrypted aggregate shares from the Leader
Leader-Upload / Split-Upload

**Leader-Upload** (status quo) – Report contains all encrypted input shares

1. **Client** reports encrypted input shares to **Leader**
2. **Leader** aggregates the reports
3. **Helper** aggregates the reports

**Split-Upload** (PR #174) – Report split into report shares, each containing the encrypted input share of the recipient

1. **Client** reports encrypted input shares to **Leader**
2. **Leader** distributes report shares
3. **Helper** aggregates the report shares
Motivations for Leader-Upload

- #1 Only the leader has high capacity requirements
  - **Upload Flow** – **HIGH** capacity
    - Bandwidth = report_size * num_clients * reports_per_sec
    - Clients are online, so needs to be fast
  - **Aggregate Flow** – **MODERATE** capacity
    - Bandwidth reduced by factor of $O(1)$ to $O(report_size)$, depending on the VDAF
    - Leader can throttle traffic if needed
  - **Collect Flow** – **LOW** capacity

![Diagram of Leader-Upload and Split-Upload](image-url)
Motivations for Leader-Upload

- **#2** Resolves data race in Split-Upload
  - Between:
    - Leader receives report share and initiates aggregation flow (doesn't know if the helper has received its share yet)
    - Helper receives report share
  - Split-Upload requires additional retry logic to resolve this (or else tolerate additional data loss)
    - We have other sources of data loss already, so maybe not so bad?
Motivations for Leader-Upload

- #3 In Split-Upload, upload flow is more likely to fail since there are two HTTP requests instead of just one.
Downside of Leader-Upload

- Aggregation flow has higher-than-necessary bandwidth
  - Significant problem for Poplar [BBCG+21]
    - Size of both input shares are $O(N)$ where $N$ is the length (in bits) of the input strings. Concretely:
      - $N=32 \Rightarrow \sim 2\text{KB/share}$
      - $N=64 \Rightarrow \sim 4\text{KB/share}$
      - $N=128 \Rightarrow \sim 8\text{KB/share}$
    - Poplar requires $N$ runs to compute heavy hitters (spec currently requires retransmitting report shares at the start of each aggregation run)
      - Higher bandwidth $\Rightarrow$ higher egress cost (issue #130)
Options

- **Option #1** – Stick with Leader-Upload, but mitigate its downside
  - Change the protocol so that report shares need only be transmitted once (in the first aggregation run)
    - Question: Is this enough?

- **Option #2** – Take Split-Upload (PR #174) and leave mitigation of downsides up to the deployment
  - One can "emulate" Leader-Upload by putting an *Ingestor* between Client and Aggregators
    - Question: In what sense is the Ingestor *trusted* or *untrusted*?