DAP updates and open issues

IETF 120 – PPM

Changes since IETF 118

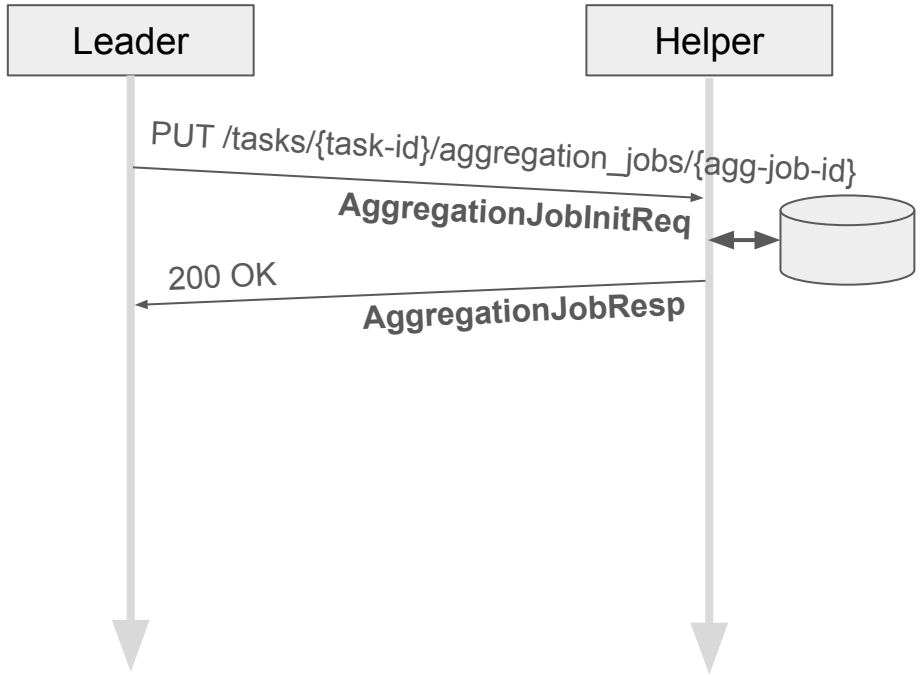
- draft-ietf-ppm-dap-09: `fixed_size` queries: Make max batch size optional
- draft-ietf-ppm-dap-10: Incorporate feedback from early HTTPDIR review (Mark Nottingham)
- draft-ietf-ppm-dap-11: Result of consensus call following discussion at the interim: remove support for multi-collection. DAP no longer supports heavy hitters via Poplar1.
- Closed the drill down issue (#490) without action. Options for Prio3:
  - Specify variant with plaintext labels
  - Use Mastic instead (under development)

PR #563: Remove `max_batch_size` task parameter

- Context: The `fixed_size` query type is so named because it has both a minimum and maximum batch size. The maximum batch size turned out to be slightly annoying to implement, as well as not particularly useful. In fact, we made it optional in draft-ietf-ppm-dap-09.
- Proposal: Remove the maximum batch size. Related nits:
  - Rename `QueryType` to `BatchMode`
    - Aligns better with intended semantics: it refers *both* to how reports may be partitioned during aggregation *and* to how they're addressed during collection
  - Rename the query type to `leader_selected`
  - Security considerations: Describe Sybil-like attack in which the Leader attempts to partition reports by Client

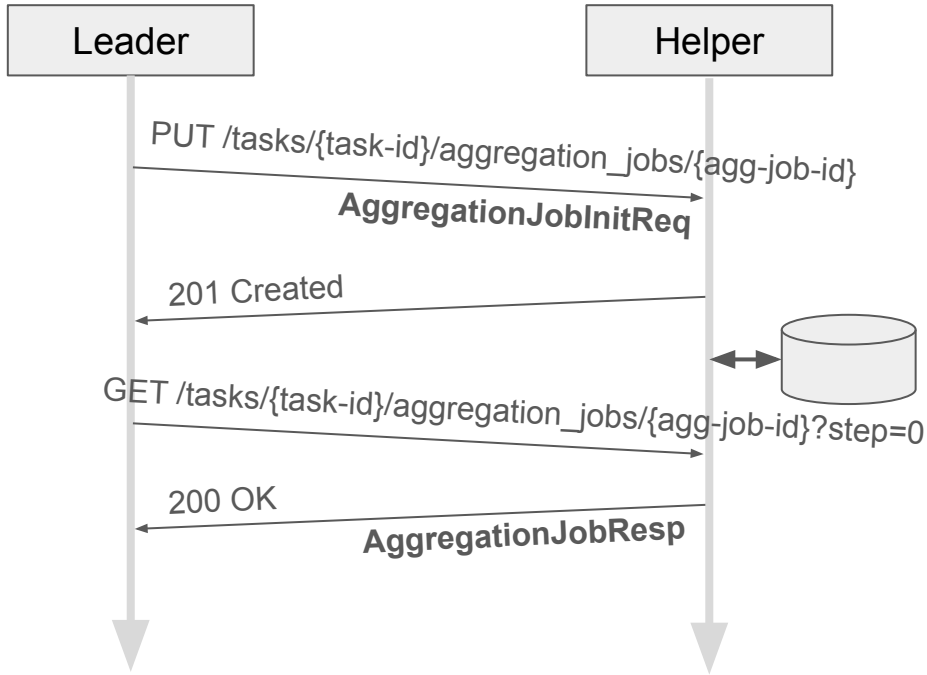
PR #564: Make aggregation asynchronous

- Context: Aggregation is "synchronous": each HTTP request blocks until Helper completes VDAF preparation and commits states changes (record report IDs for anti-replay, update aggregate share).
    - Deployment experience: a small number of requests take 10+ seconds to resolve or need to be aborted and retried (resource contention).



PR #564: Make aggregation asynchronous

- Context: Aggregation is "synchronous": each HTTP request blocks until Helper completes VDAF preparation and commits states changes (record report IDs for anti-replay, update aggregate share).
  - Deployment experience: a small number of requests take 10+ seconds to resolve or need to be aborted and retried (resource contention).
- Proposal: Allow Helper to process job asynchronously
  - Helper responds to PUT with 201 Created
  - Leader sends GET to poll for result



PR #566: Document deviations from RFC 8446

Context: DAP implicitly overloads RFC 8446 (TLS 1.3) presentation language. The following is for fields with fixed constants (Section 3.7). We use it for specifying how a message is formatted ("the Helper replies with the following message")

```
struct {
  PrepareStepState prepare_step_state = 2; /* reject */
  ReportId report_id;
  ReportShareError report_share_error;
} PrepareStep;
```

Proposal: Make this syntax explicit with new `struct variant` notation.

```
struct variant {
  PrepareStepState prepare_step_state = 2; /* reject */
  ReportId report_id;
  ReportShareError report_share_error;
} PrepareStep;
```

PR #567: Drop `task_id` param from HPKE config endpoint

- Context: DAP Aggregators may have per-task HPKE configurations to reduce risk of key compromise. To support this, Clients indicate the task ID when requesting the configs.
  - This might complicate applications that need anonymity (e.g., DAP over OHTTP): Aggregators know which tasks a Client participates in.
  - This feature is not used in any known implementation.
- Proposal: Remove the optional `task_id` parameter from the `/hpke_config` endpoint, forcing an Aggregator endpoint to use the same set of HPKE configs for all tasks.
  - Can still do key separation across endpoints

PR #568: Content type versioning

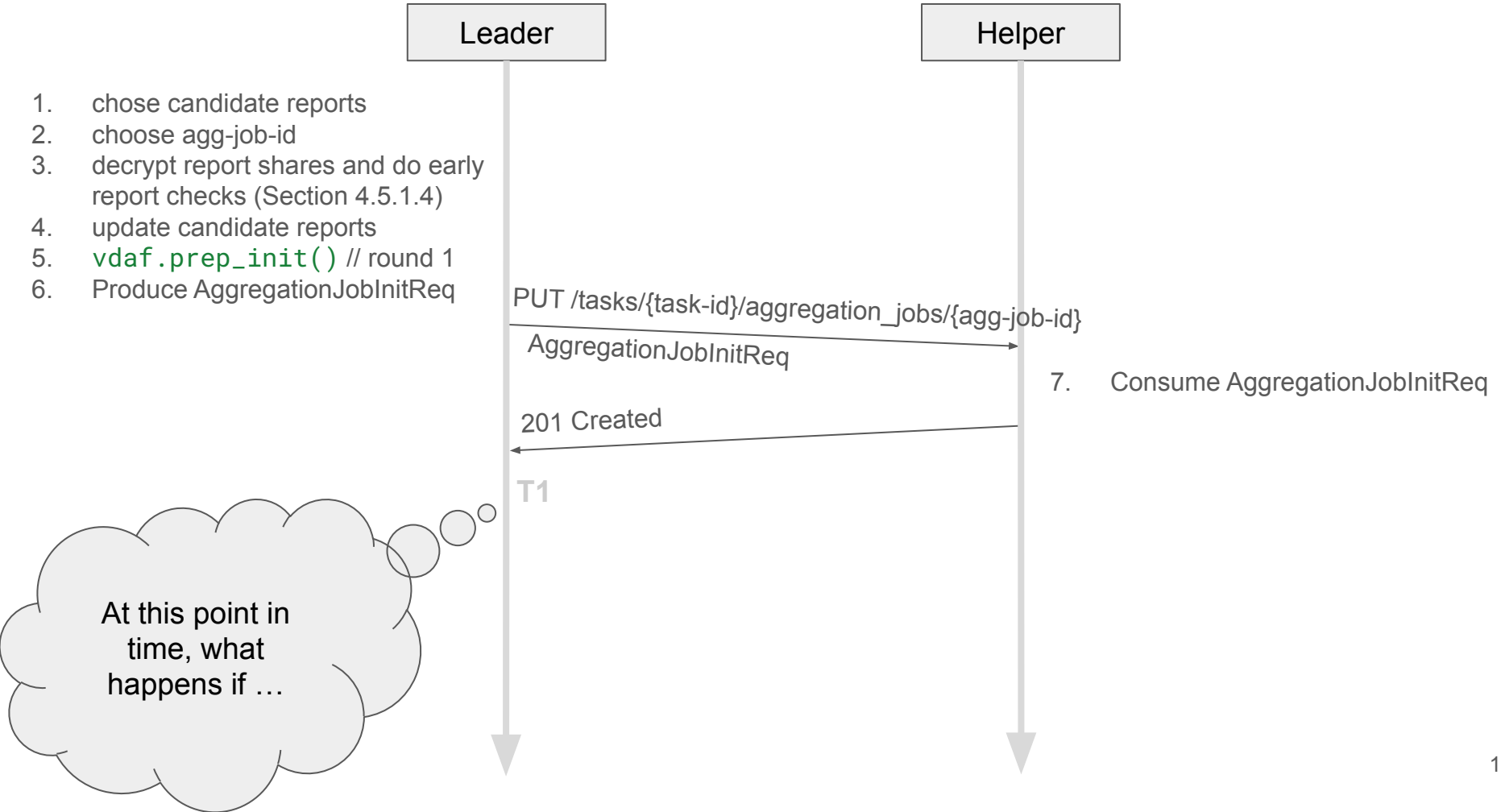
- Context: the content type for an HTTP request (eg., `"application/dap-aggregation-job-init-req"`) is meant to convey how the request body is parsed. However:
  - "DAP v2" will likely be wire-incompatible with the current protocol
  - Drafts of the current protocol are often wire-incompatible
- Proposal: Clarify that major revisions to DAP will use new media types (e.g., `"dap"` → `"dap2"`). Also, note that media types have an optional parameter that can be used to convey the draft number (may be useful for debugging).

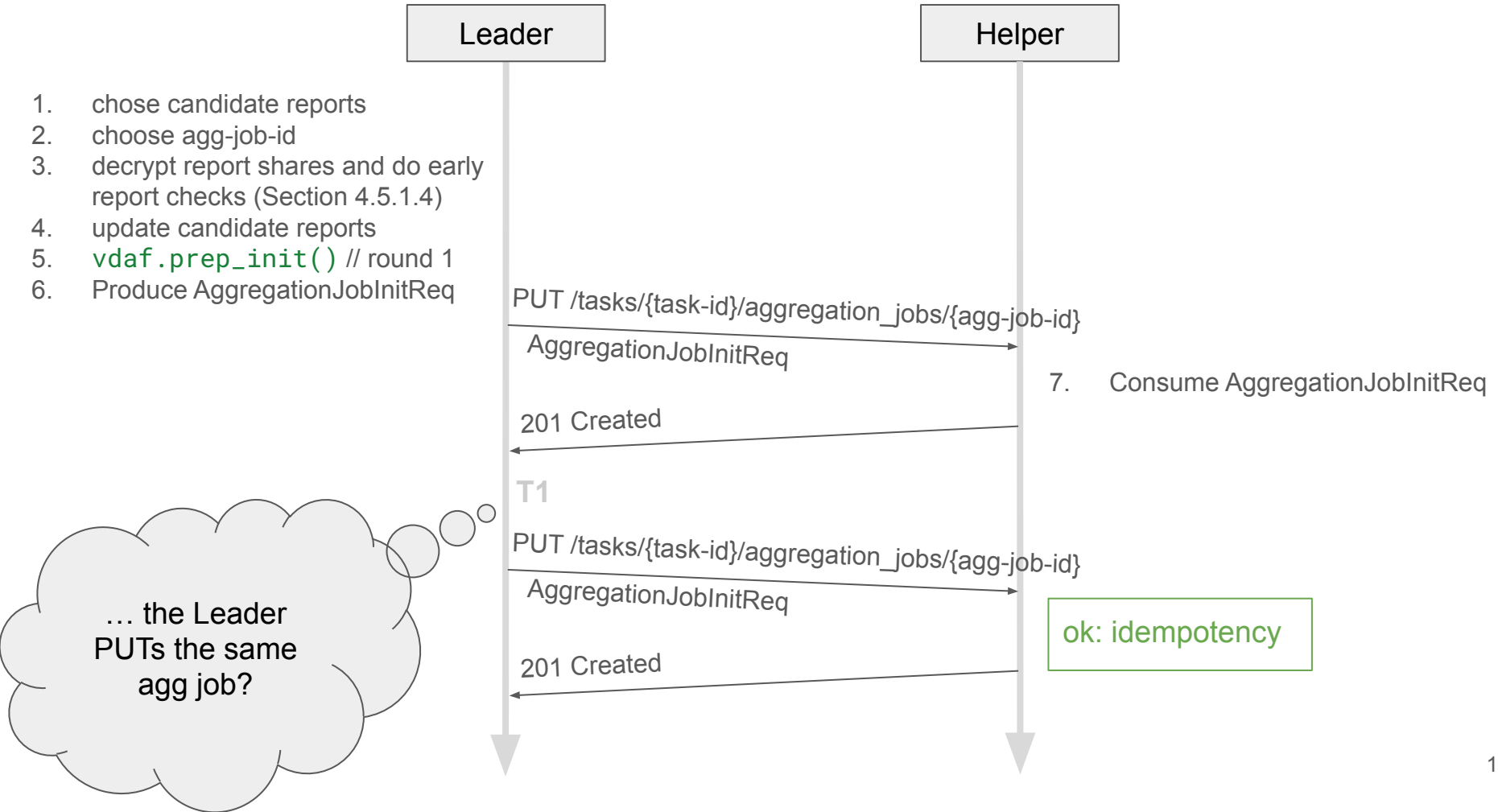
PR #569: Allow aggregation job skew recovery at any step

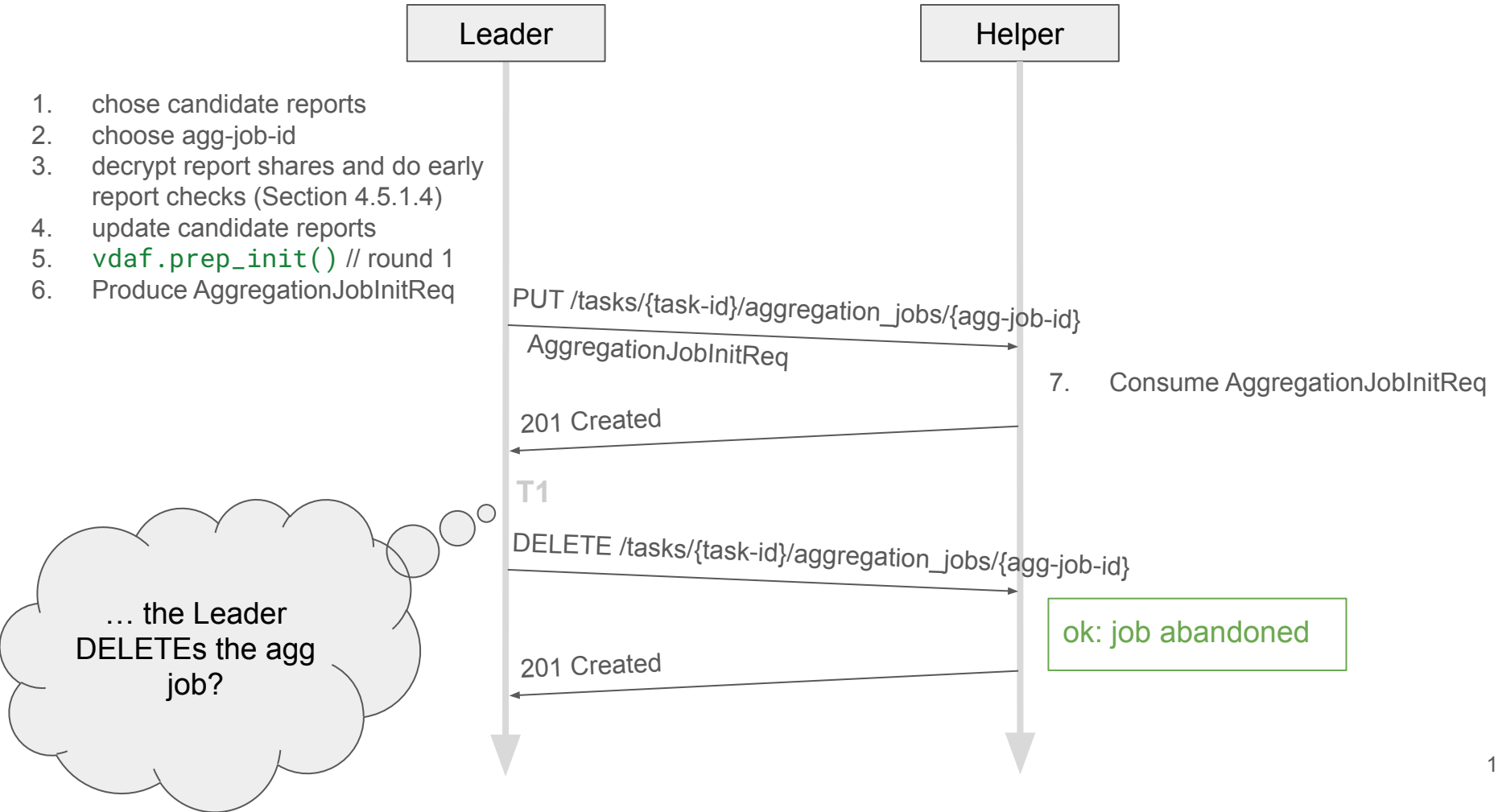
- Context:
  - Multi-round VDAFs:
    - 1-round VDAFs (Prio3) complete after initialization (the `AggregationJobInitReq` and corresponding `AggregationJobResp`)
    - 2-round VDAFs (Poplar1) require initialization and one round of continuation (`AggregationJobContinueReq`)
    - 3-round VDAFs require one continuation, 4-round VDAFs require two rounds of continuation, …
  - If the Leader sends the request for the previous continuation step (e.g., due to state rollback), then the Helper MAY reply with the corresponding `AggregationJobResp` to help the Leader recover. We call this **step skew recovery**.
- Proposal: Allow the Helper to recover from any continuation step.

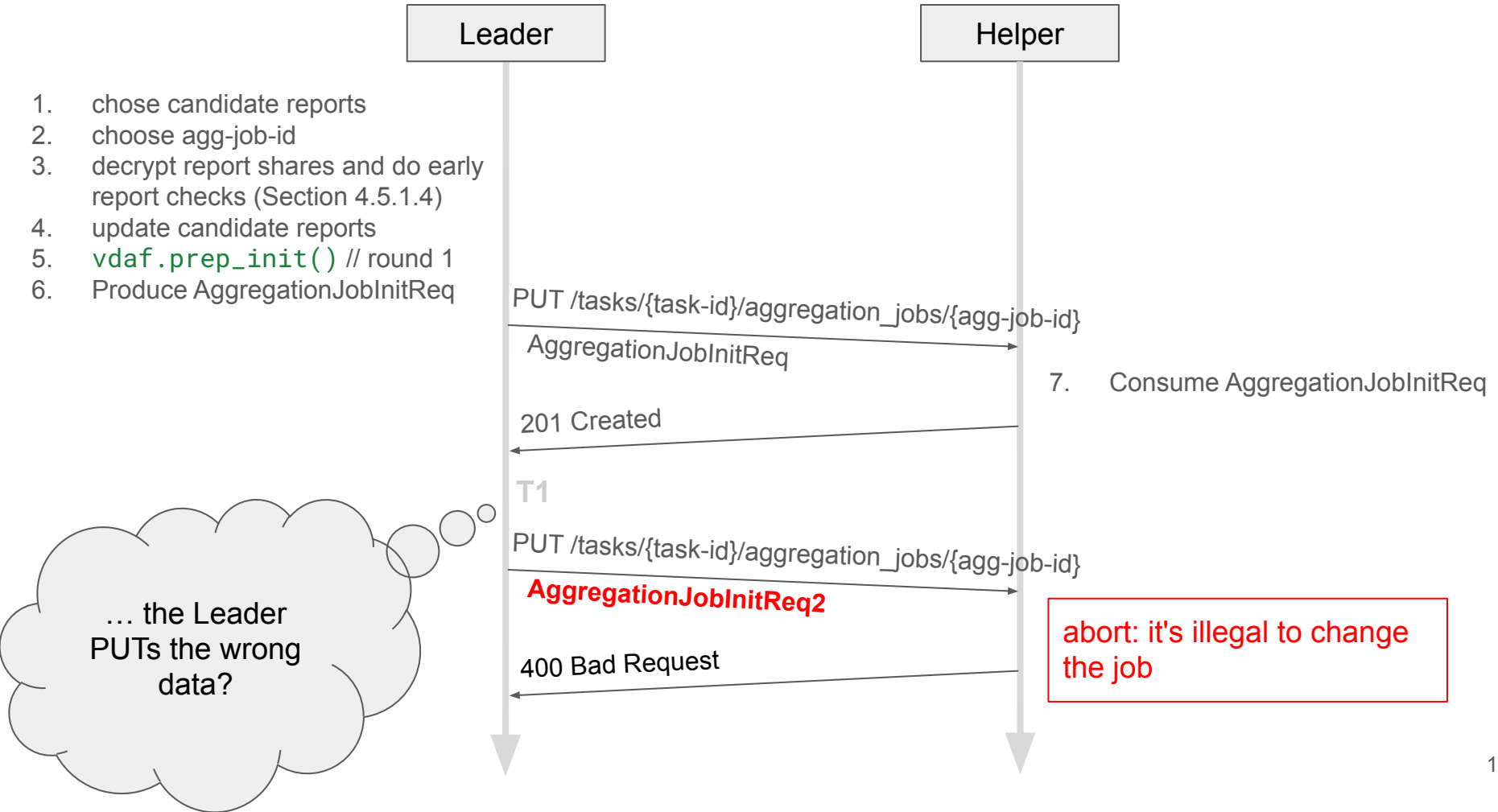
Life of an aggregation job

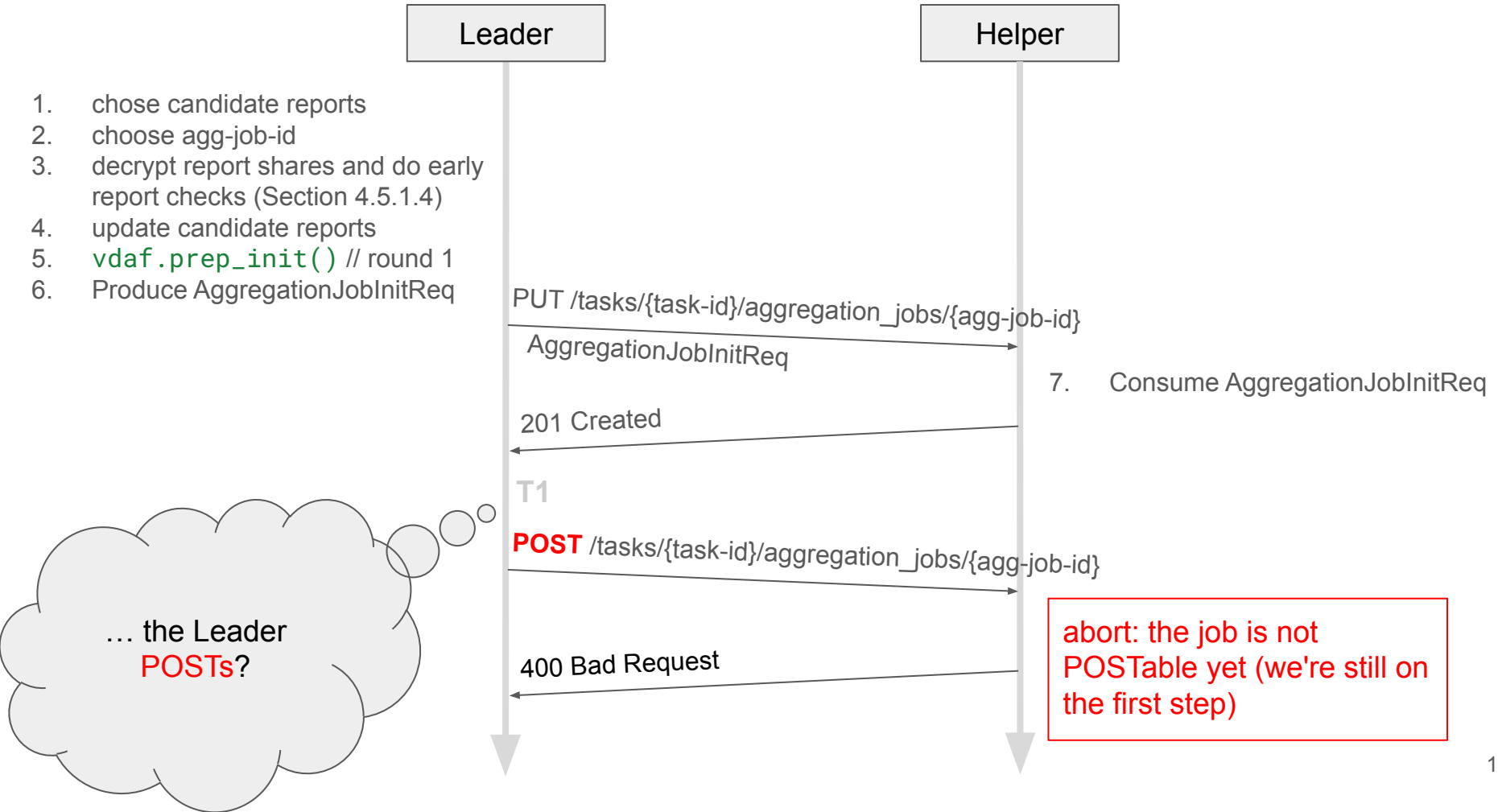
(2-round VDAF)

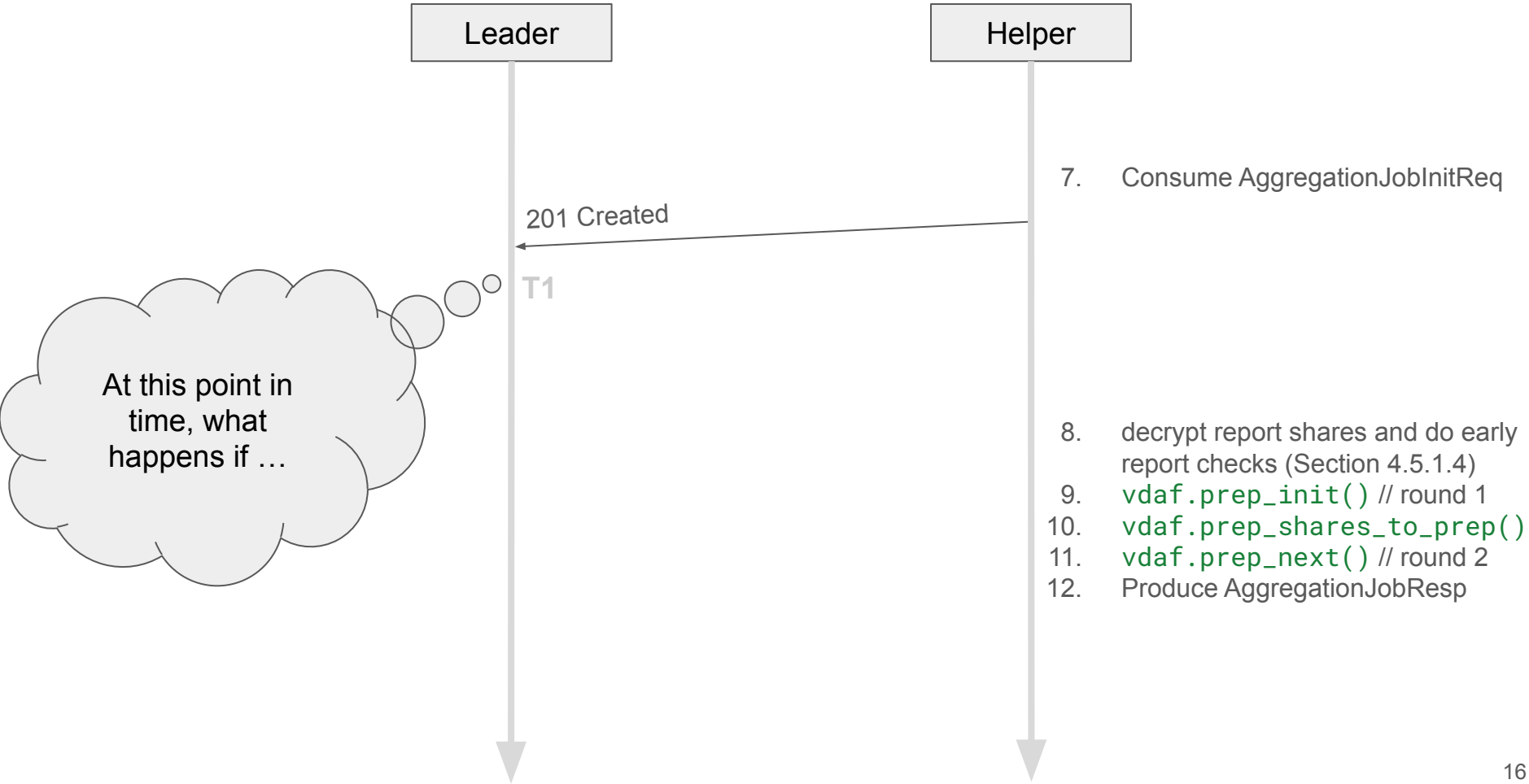

Leader
Helper
1. chose candidate reports
2. choose agg-job-id
3. decrypt report shares and do early report checks (Section 4.5.1.4)
4. update candidate reports
5. vdaf.prep_init() // round 1
6. Produce AggregationJobInitReq
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
7. Consume AggregationJobInitReq
201 Created
T1
At this point in time, what happens if …

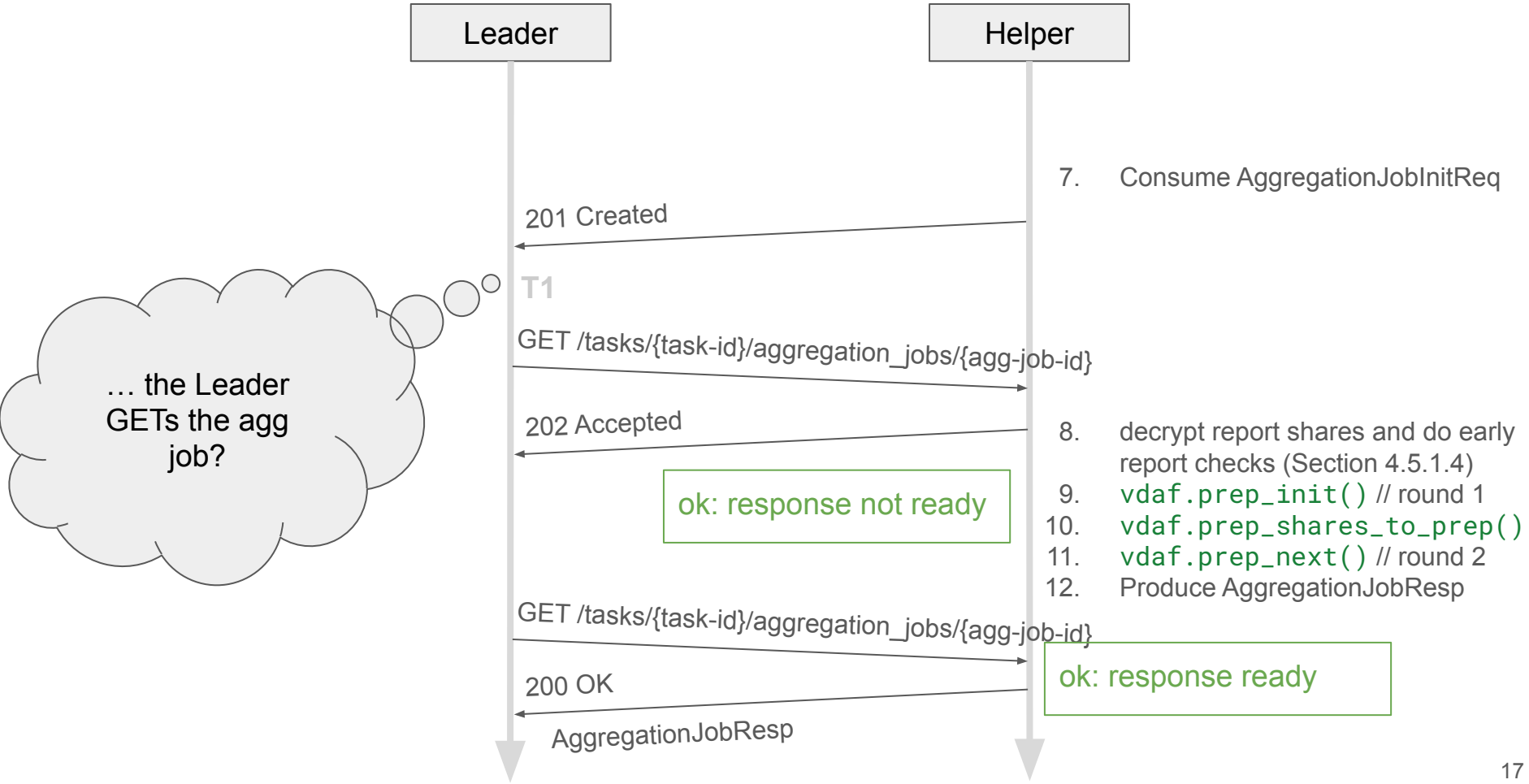

Leader
Helper
1. chose candidate reports
2. choose agg-job-id
3. decrypt report shares and do early report checks (Section 4.5.1.4)
4. update candidate reports
5. vdaf.prep_init() // round 1
6. Produce AggregationJobInitReq
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
7. Consume AggregationJobInitReq
201 Created
T1
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
ok: idempotency
201 Created
… the Leader PUTs the same agg job?

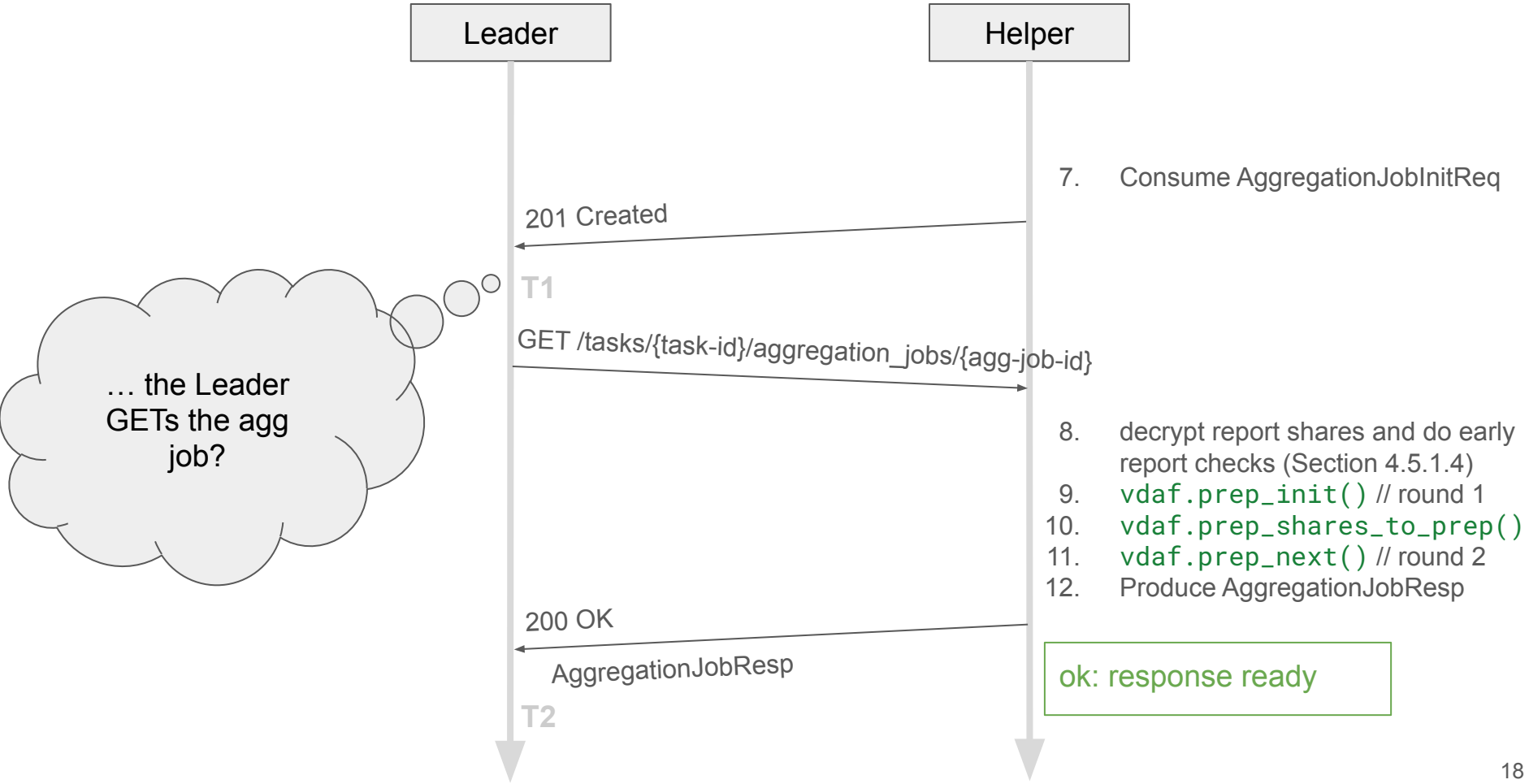

Leader
Helper
1. chose candidate reports
2. choose agg-job-id
3. decrypt report shares and do early report checks (Section 4.5.1.4)
4. update candidate reports
5. vdaf.prep_init() // round 1
6. Produce AggregationJobInitReq
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
7. Consume AggregationJobInitReq
201 Created
T1
DELETE /tasks/{task-id}/aggregation_jobs/{agg-job-id}
ok: job abandoned
201 Created
… the Leader DELETEs the agg job?

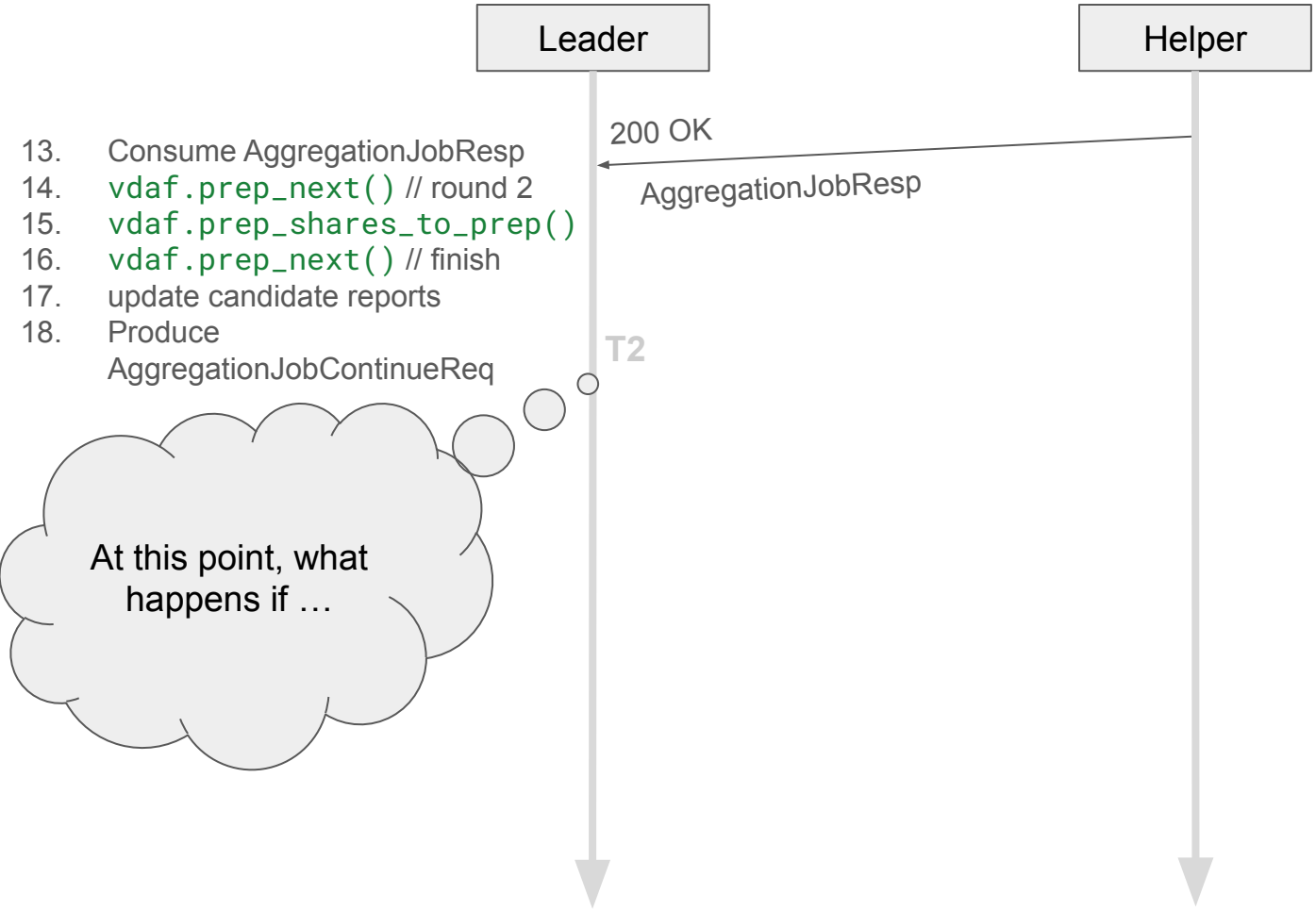

Leader
Helper
1. chose candidate reports
2. choose agg-job-id
3. decrypt report shares and do early report checks (Section 4.5.1.4)
4. update candidate reports
5. vdaf.prep_init() // round 1
6. Produce AggregationJobInitReq
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
7. Consume AggregationJobInitReq
201 Created
T1
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq2
400 Bad Request
abort: it's illegal to change the job
… the Leader PUTs the wrong data?

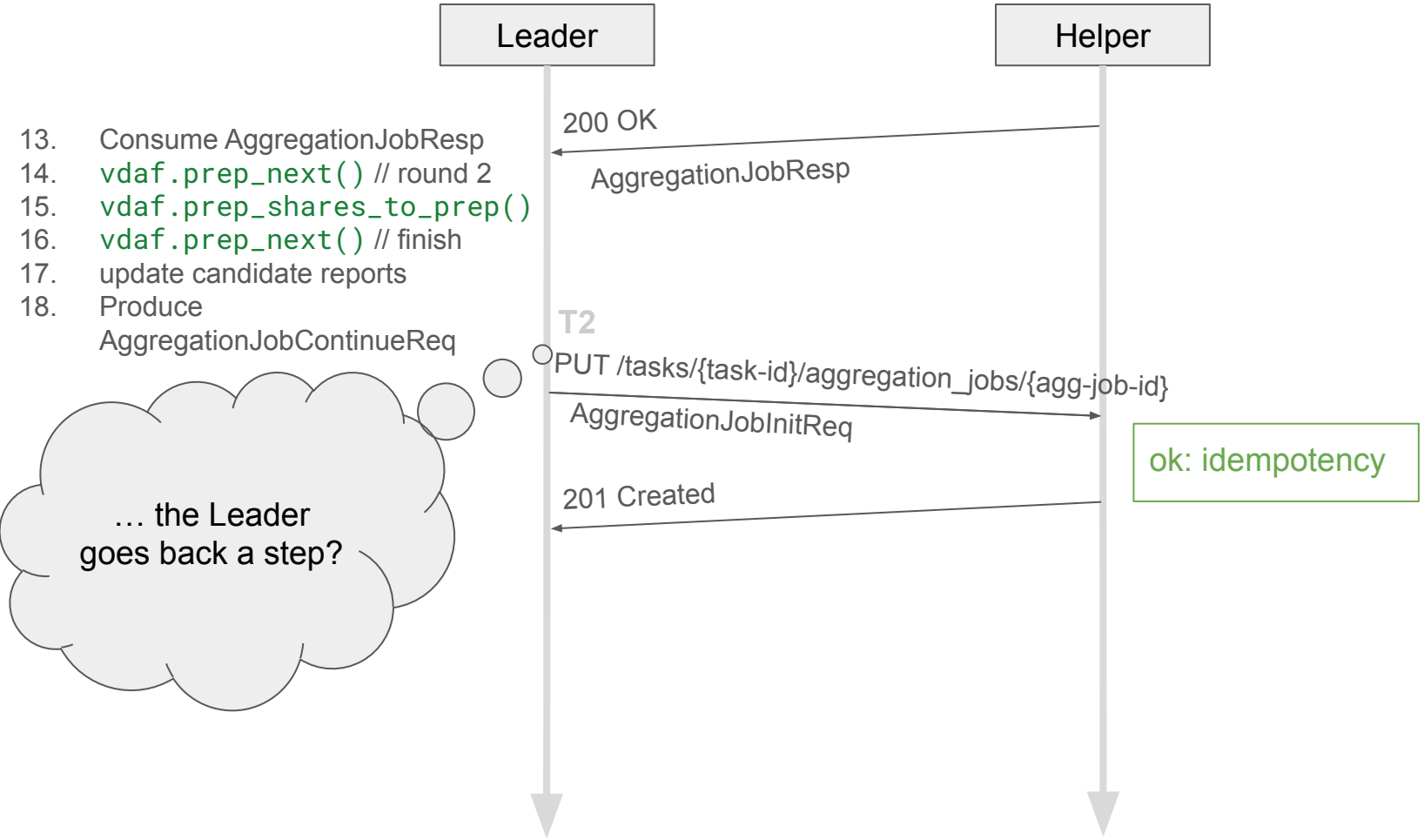

Leader
Helper
1. chose candidate reports
2. choose agg-job-id
3. decrypt report shares and do early report checks (Section 4.5.1.4)
4. update candidate reports
5. vdaf.prep_init() // round 1
6. Produce AggregationJobInitReq
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
7. Consume AggregationJobInitReq
201 Created
T1
POST /tasks/{task-id}/aggregation_jobs/{agg-job-id}
… the Leader POSTs?
400 Bad Request
abort: the job is not POSTable yet (we're still on the first step)

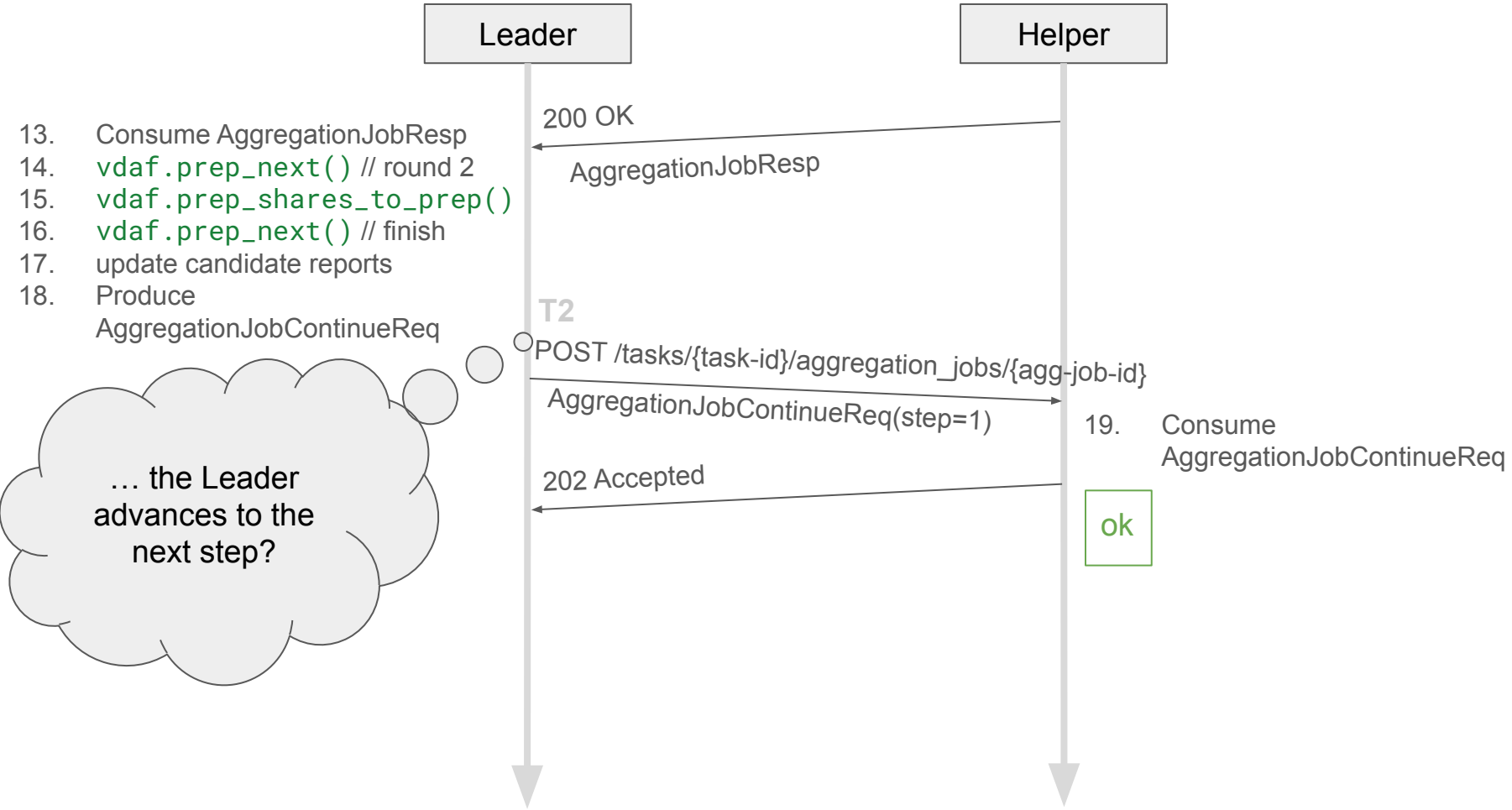

Leader
Helper
7. Consume AggregationJobInitReq
201 Created
T1
At this point in time, what happens if …
8. decrypt report shares and do early report checks (Section 4.5.1.4)
9. vdaf.prep_init() // round 1
10. vdaf.prep_shares_to_prep()
11. vdaf.prep_next() // round 2
12. Produce AggregationJobResp

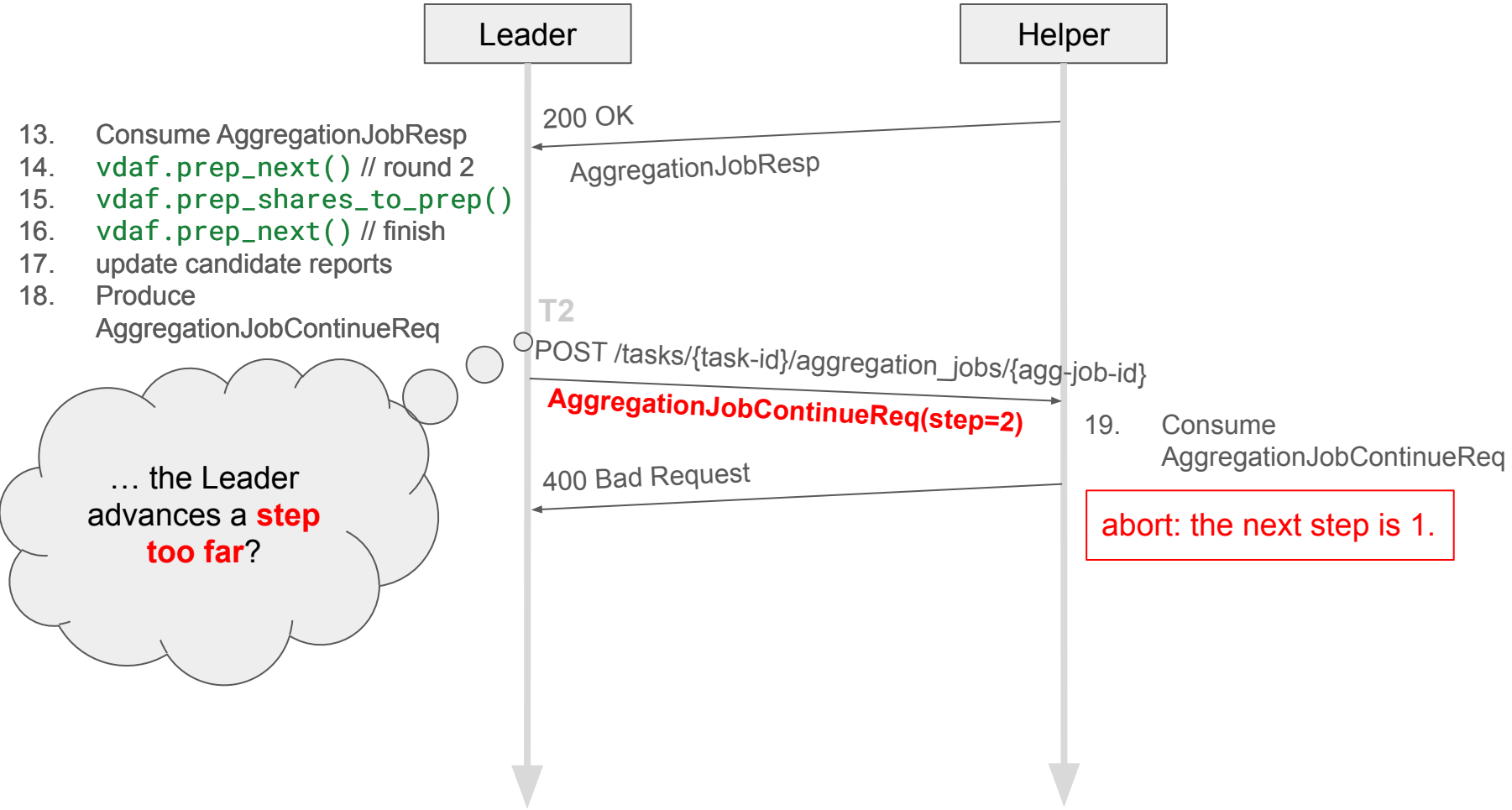

Leader
Helper
7. Consume AggregationJobInitReq
201 Created
T1
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
… the Leader GETs the agg job?
202 Accepted
8. decrypt report shares and do early report checks (Section 4.5.1.4)
ok: response not ready
9. vdaf.prep_init() // round 1
10. vdaf.prep_shares_to_prep()
11. vdaf.prep_next() // round 2
12. Produce AggregationJobResp
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
ok: response ready
200 OK
AggregationJobResp

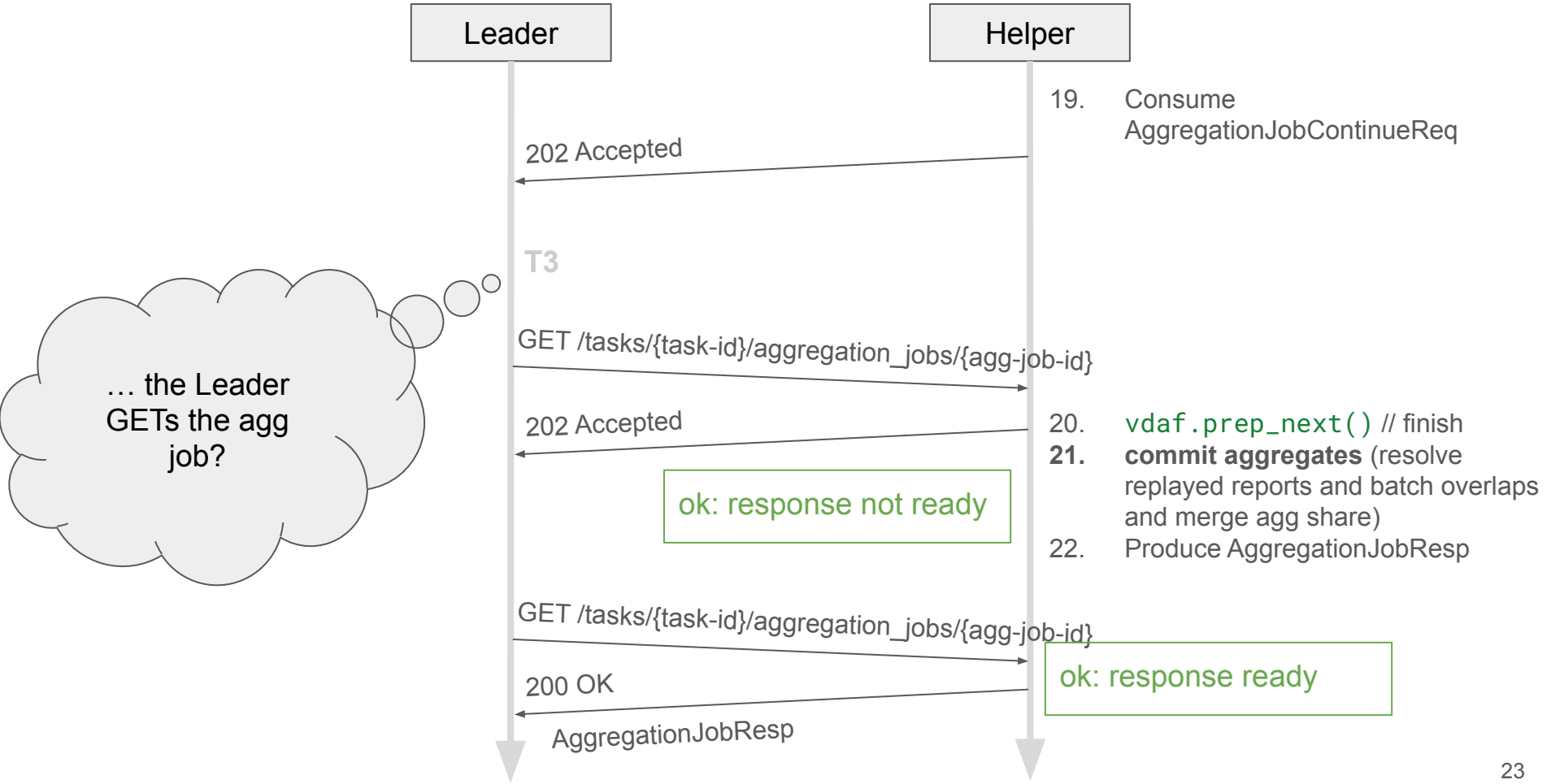

Leader
Helper
7. Consume AggregationJobInitReq
201 Created
T1
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
… the Leader GETs the agg job?
8. decrypt report shares and do early report checks (Section 4.5.1.4)
9. vdaf.prep_init() // round 1
10. vdaf.prep_shares_to_prep()
11. vdaf.prep_next() // round 2
12. Produce AggregationJobResp
200 OK
AggregationJobResp
T2
ok: response ready

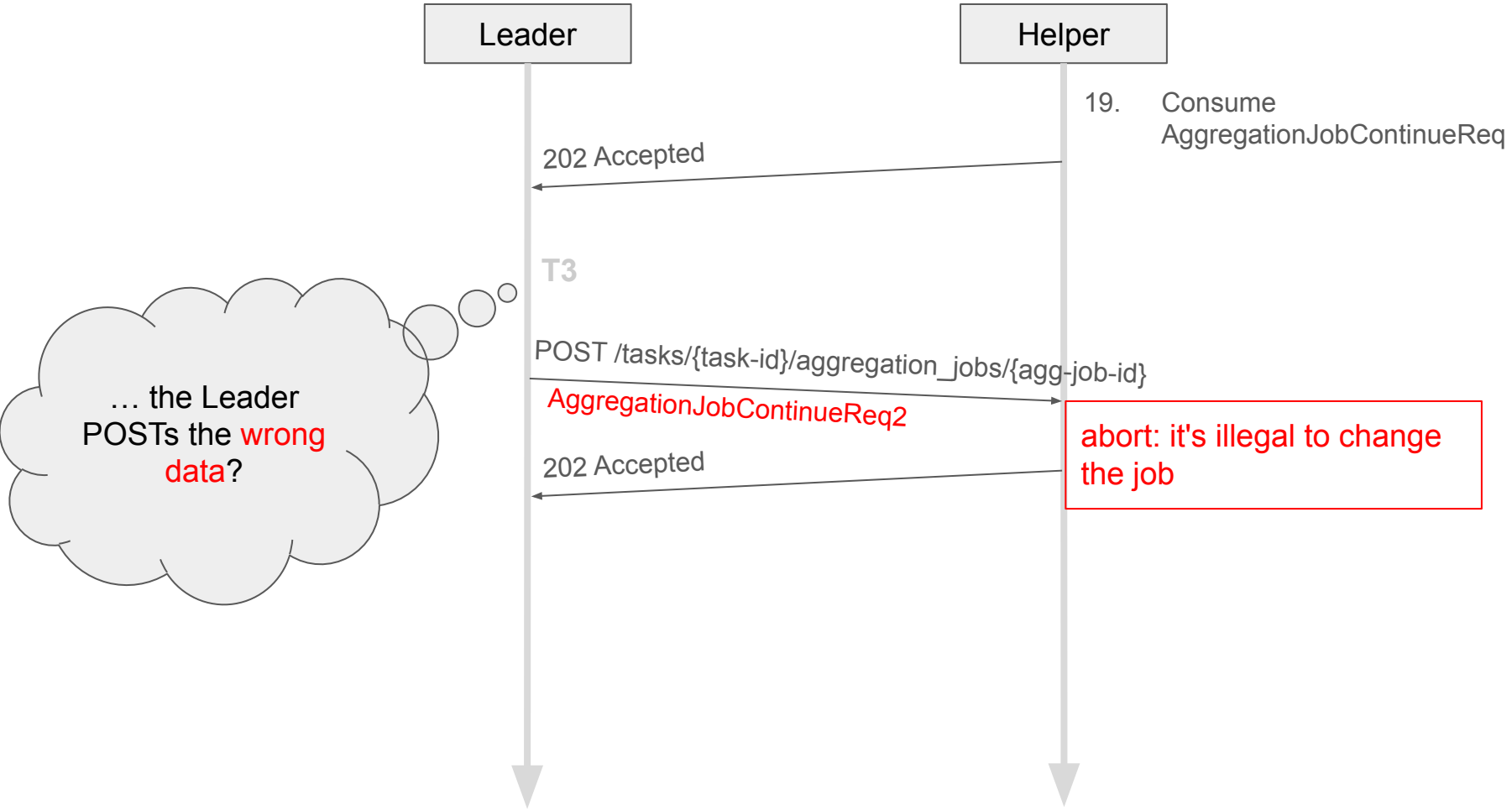

Leader
Helper
200 OK
AggregationJobResp
13. Consume AggregationJobResp
14. vdaf.prep_next() // round 2
15. vdaf.prep_shares_to_prep()
16. vdaf.prep_next() // finish
17. update candidate reports
18. Produce AggregationJobContinueReq
T2
At this point, what happens if …

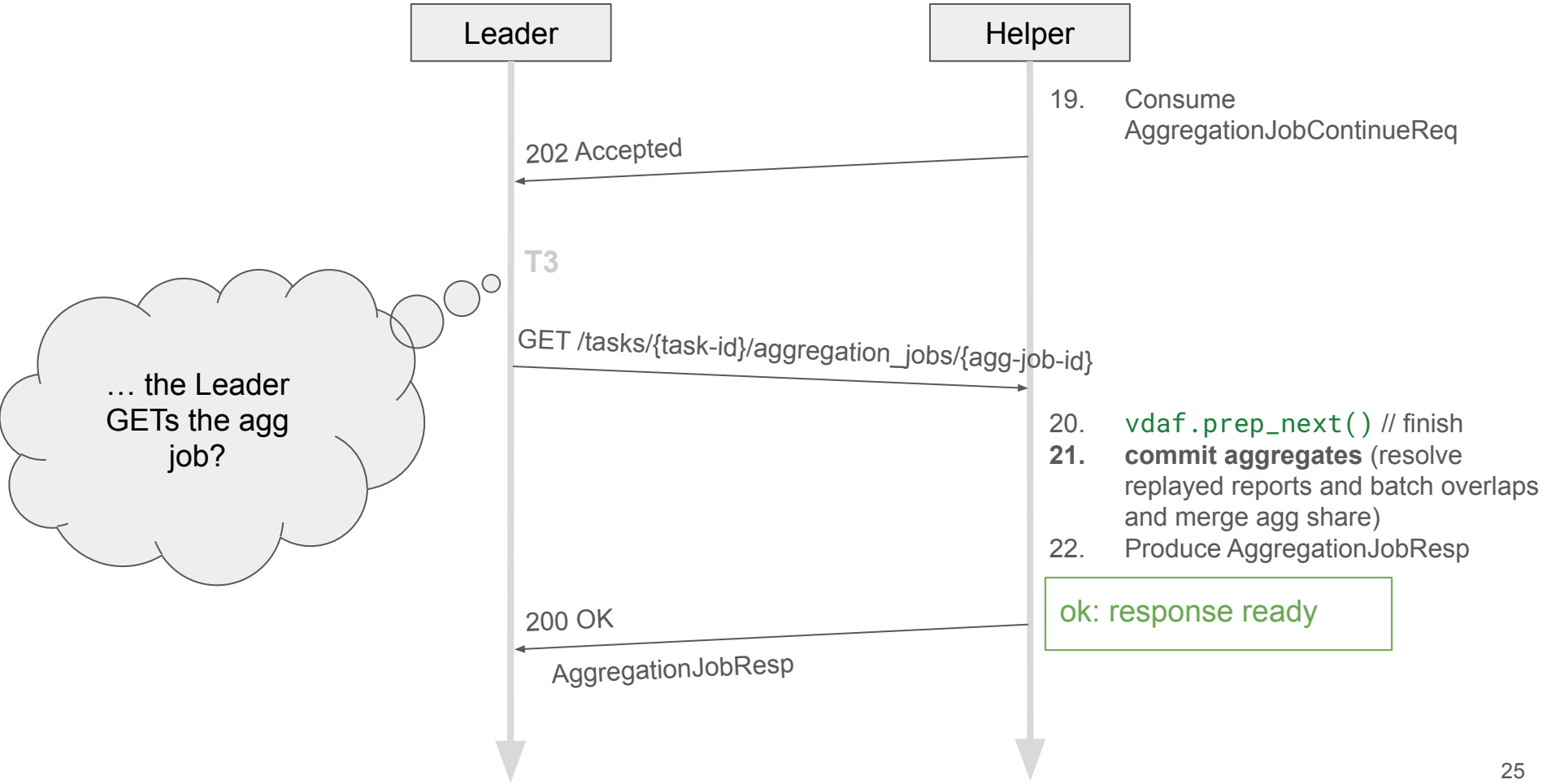

Leader
Helper
200 OK
AggregationJobResp
13. Consume AggregationJobResp
14. vdaf.prep_next() // round 2
15. vdaf.prep_shares_to_prep()
16. vdaf.prep_next() // finish
17. update candidate reports
18. Produce AggregationJobContinueReq
T2
PUT /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobInitReq
ok: idempotency
201 Created
… the Leader goes back a step?

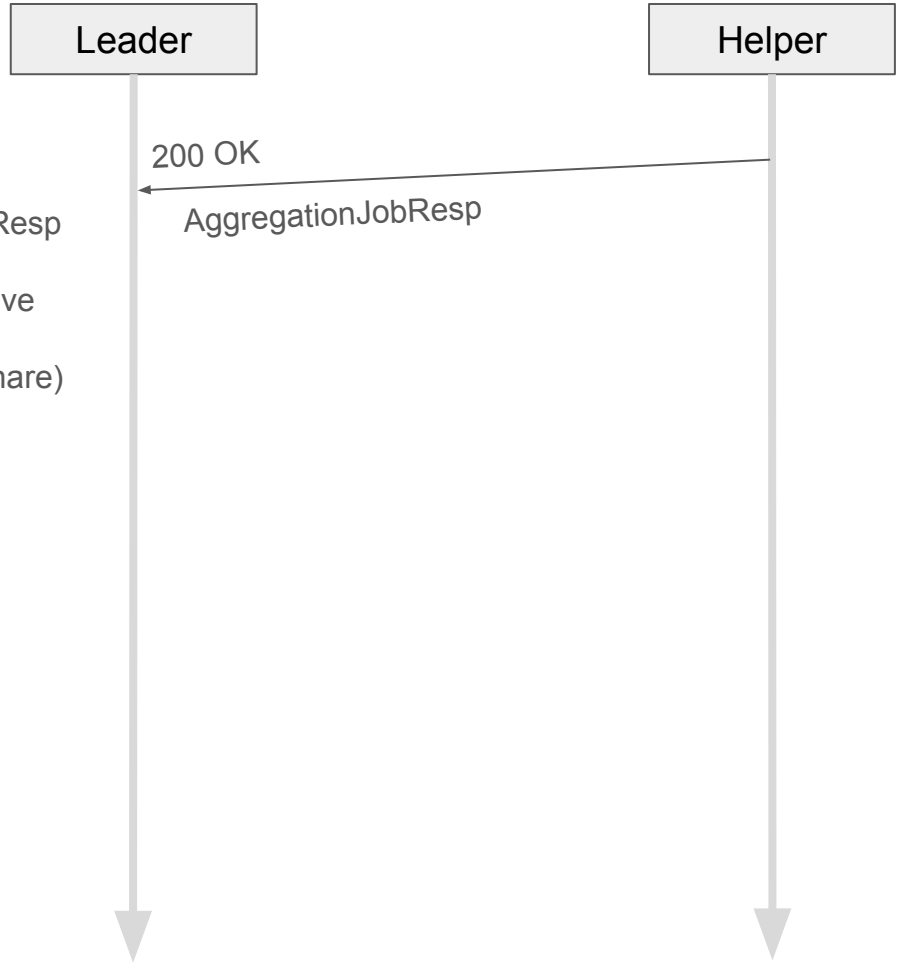

Leader
Helper
200 OK
AggregationJobResp
13. Consume AggregationJobResp
14. vdaf.prep_next() // round 2
15. vdaf.prep_shares_to_prep()
16. vdaf.prep_next() // finish
17. update candidate reports
18. Produce AggregationJobContinueReq
T2
POST /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobContinueReq(step=1)
19. Consume AggregationJobContinueReq
202 Accepted
ok
… the Leader advances to the next step?


Leader
Helper
200 OK
AggregationJobResp
13. Consume AggregationJobResp
14. vdaf.prep_next() // round 2
15. vdaf.prep_shares_to_prep()
16. vdaf.prep_next() // finish
17. update candidate reports
18. Produce AggregationJobContinueReq
T2
POST /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobContinueReq(step=2)
19. Consume AggregationJobContinueReq
400 Bad Request
abort: the next step is 1.
… the Leader advances a step too far?


Leader
Helper
19. Consume AggregationJobContinueReq
202 Accepted
T3
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
… the Leader GETs the agg job?
202 Accepted
20. vdaf.prep_next() // finish
21. commit aggregates (resolve replayed reports and batch overlaps and merge agg share)
22. Produce AggregationJobResp
ok: response not ready
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
ok: response ready
200 OK
AggregationJobResp


Leader
Helper
19. Consume AggregationJobContinueReq
202 Accepted
T3
POST /tasks/{task-id}/aggregation_jobs/{agg-job-id}
AggregationJobContinueReq2
202 Accepted
abort: it's illegal to change the job
… the Leader POSTs the wrong data?


Leader
Helper
19. Consume AggregationJobContinueReq
202 Accepted
T3
GET /tasks/{task-id}/aggregation_jobs/{agg-job-id}
… the Leader GETs the agg job?
20. vdaf.prep_next() // finish
21. commit aggregates (resolve replayed reports and batch overlaps and merge agg share)
22. Produce AggregationJobResp
ok: response ready
200 OK
AggregationJobResp



23. Consume AggregationJobResp
24. update candidate reports
25. **commit aggregates** (resolve replayed reports and batch overlaps and merge agg share)

PR #564: Make aggregation asynchronous

Ambiguous GET requests - which step is the Leader requesting?