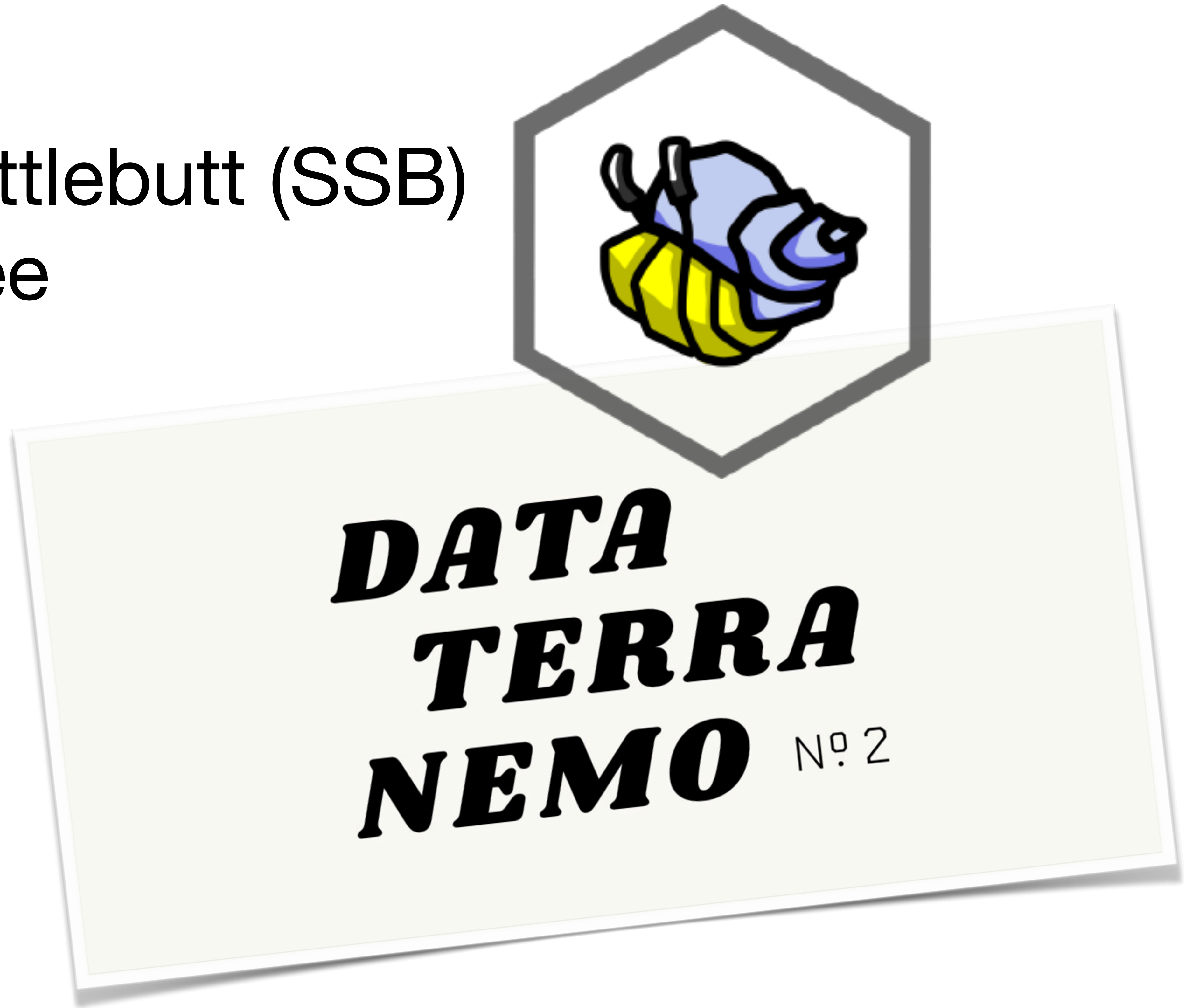
“Push it” – update 1

Christian Tschudin, University of Basel
July 21, 2019
ICNRG interim meeting, IETF Montreal

Disclaimer

This talk is only indirectly about Secure Scuttlebutt (SSB) and other decent(ralized) projects per se, see e.g. the Berlin DTN meeting in May 2019.



Instead, this talk is about push-based communication (which surprised SSB proponents because they *thought* they were pull-based)

History of this “Push”-Thread

- Sep 2018 - ICN2018-Panel: “Pull() vs Push() is an ill posed problem”
  I introduced SSB, its use of append-only logs, and remained the only person on the panel in favor of push, for unclear reasons at that time

- Feb 2019 - *submission of a CCR editorial note “A Broadcast-only communication model”, regarding the relation between PUSH and append-only logs*

- Mar 2019 - *presentation at ICNRG meeting in Prague (with a lot of interrupts)*
  “PUSH is for gods, PULL is for mortals”

- May 2019 - *CCR-online editorial note goes online (1)*

- **July 2019 - *recap and new insights***

(1) **https://ccronline.sigcomm.org/2019/a-broadcast-only-communication-model-based-on-replicated-append-only-logs/**

Overview

1. Mindset: cumulative immutable data and the “freshness frontier”

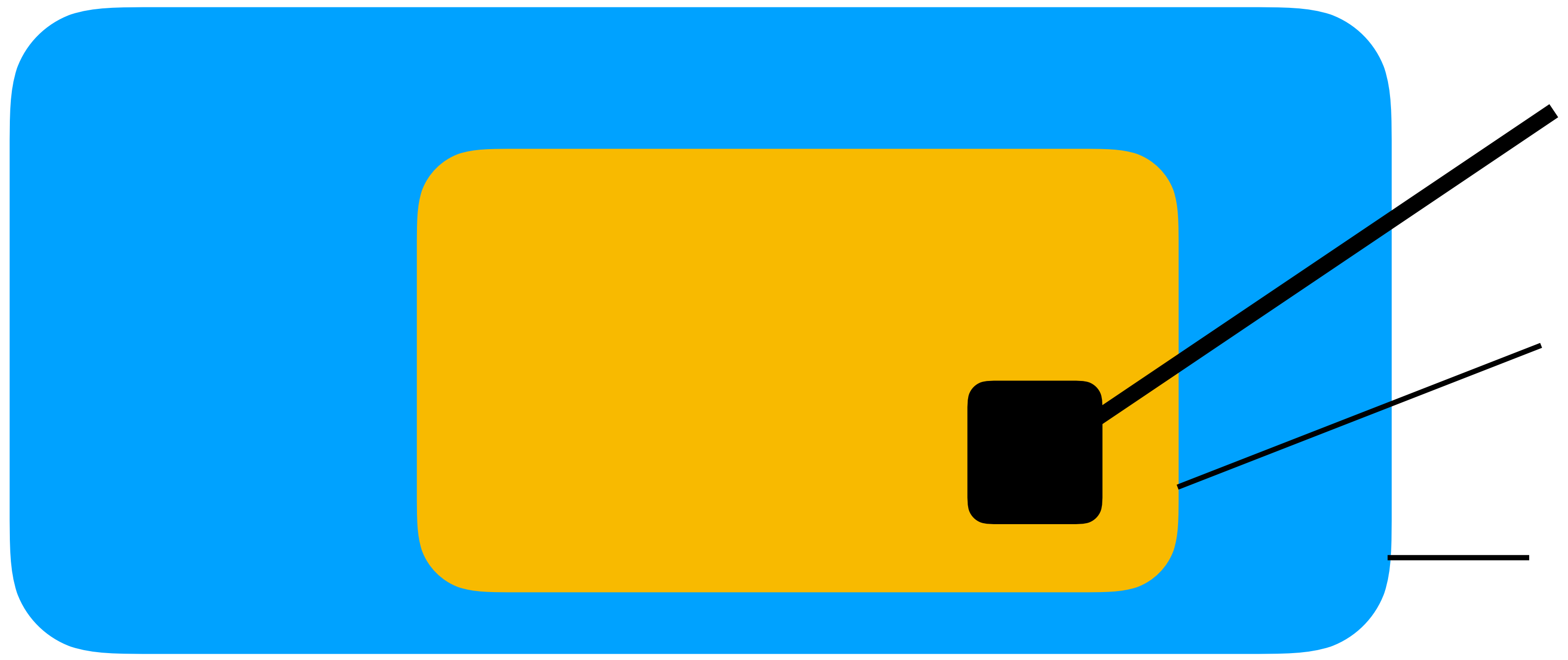
2. Recap of CCR-online note: broadcast-only through append-only logs

3. The need for push (not contested if it’s at app level)

4. Two problems of emulating “app-level push()” using “net-level pull()”

5. Extrapolations:
   - push and Shannon entropy
   - in-network memory is *not* optional

1) Cumulative Immutable Content

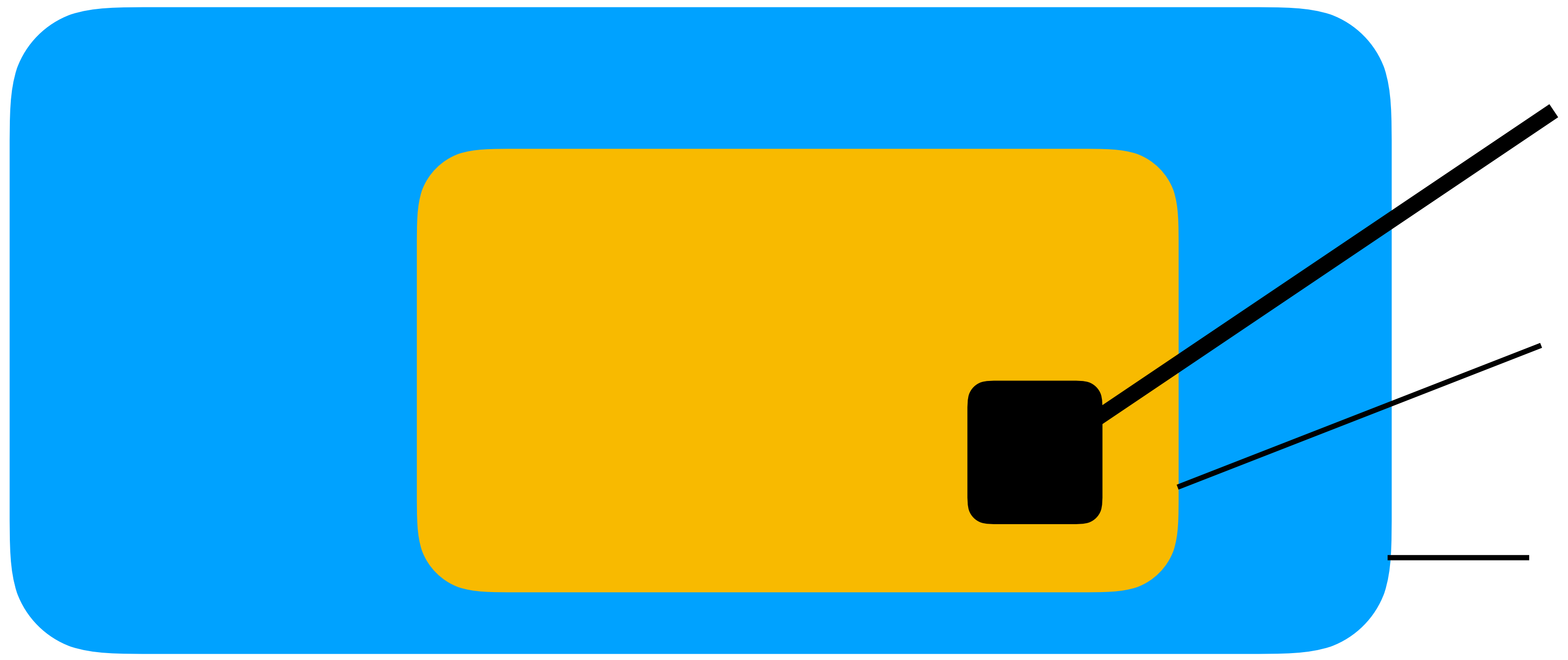


Humanity’s generated content so far, **named via some hash fct**

Image of the hash function (e.g. 2^{256} distinct values)

Universe of all content items (to be folded onto the hash’s image)

1) Cumulative Immutable Content



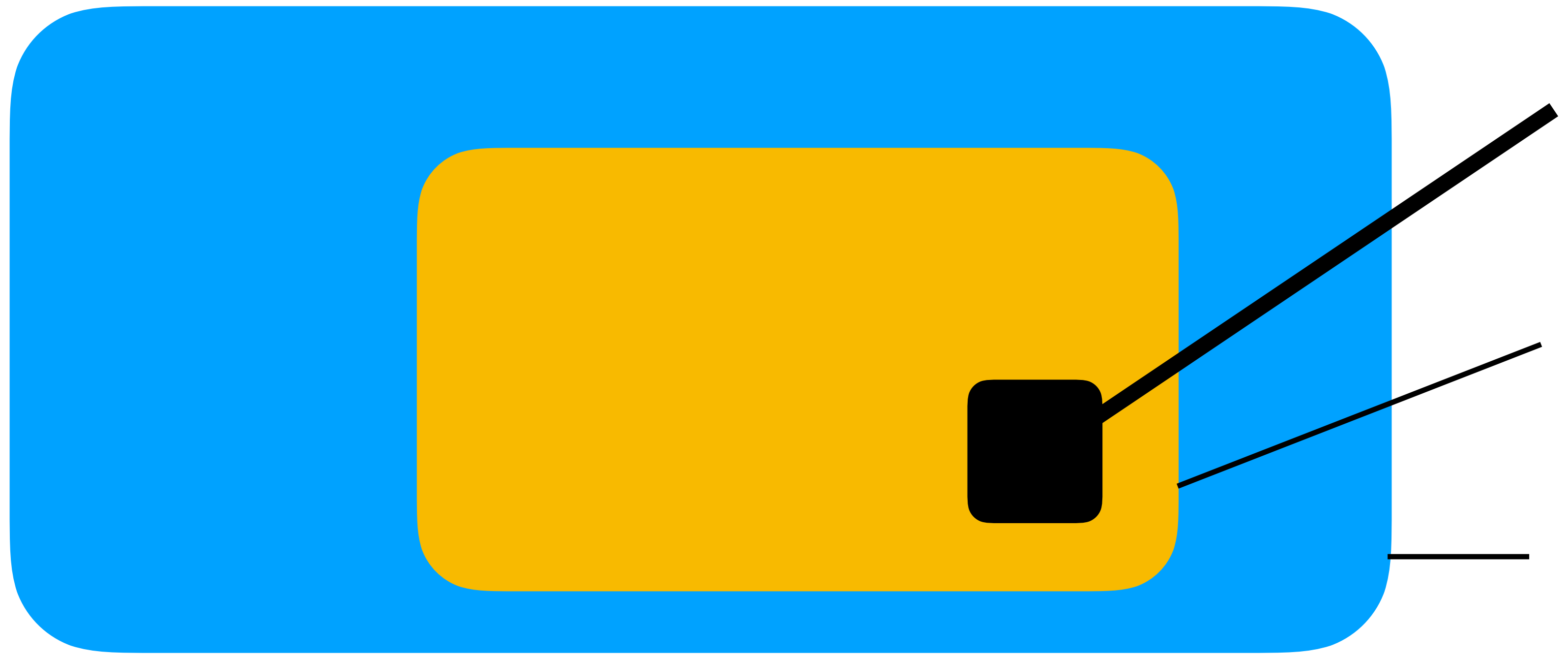
Humanity’s generated content so far, **named via some hash fct**

Image of the hash function (e.g. 2^{256} distinct values)

Universe of all content items (to be folded onto the hash’s image)

Black set (hash-named content) grows over time, has to spread in space:

1) Cumulative Immutable Content

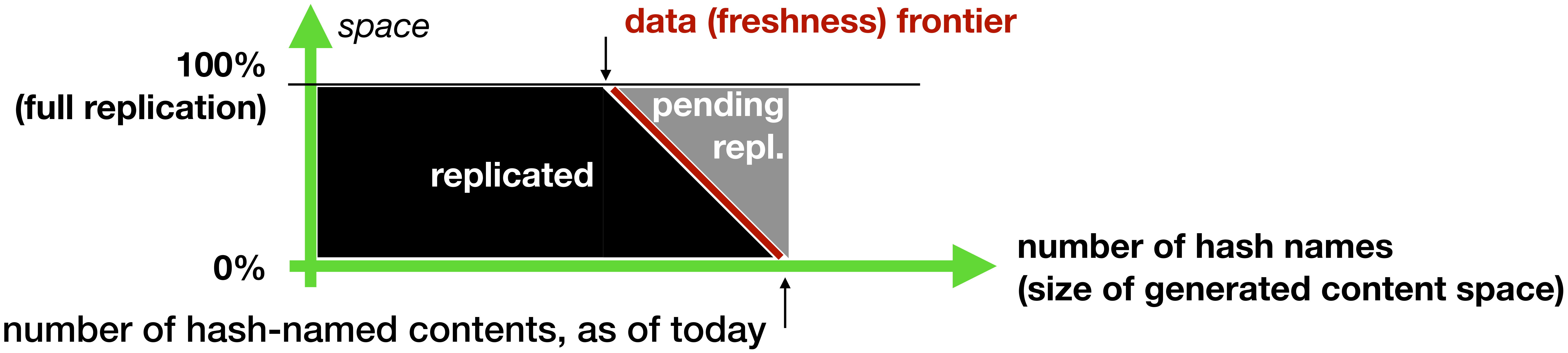


Humanity’s generated content so far, **named via some hash fct**

Image of the hash function (e.g. 2^{256} distinct values)

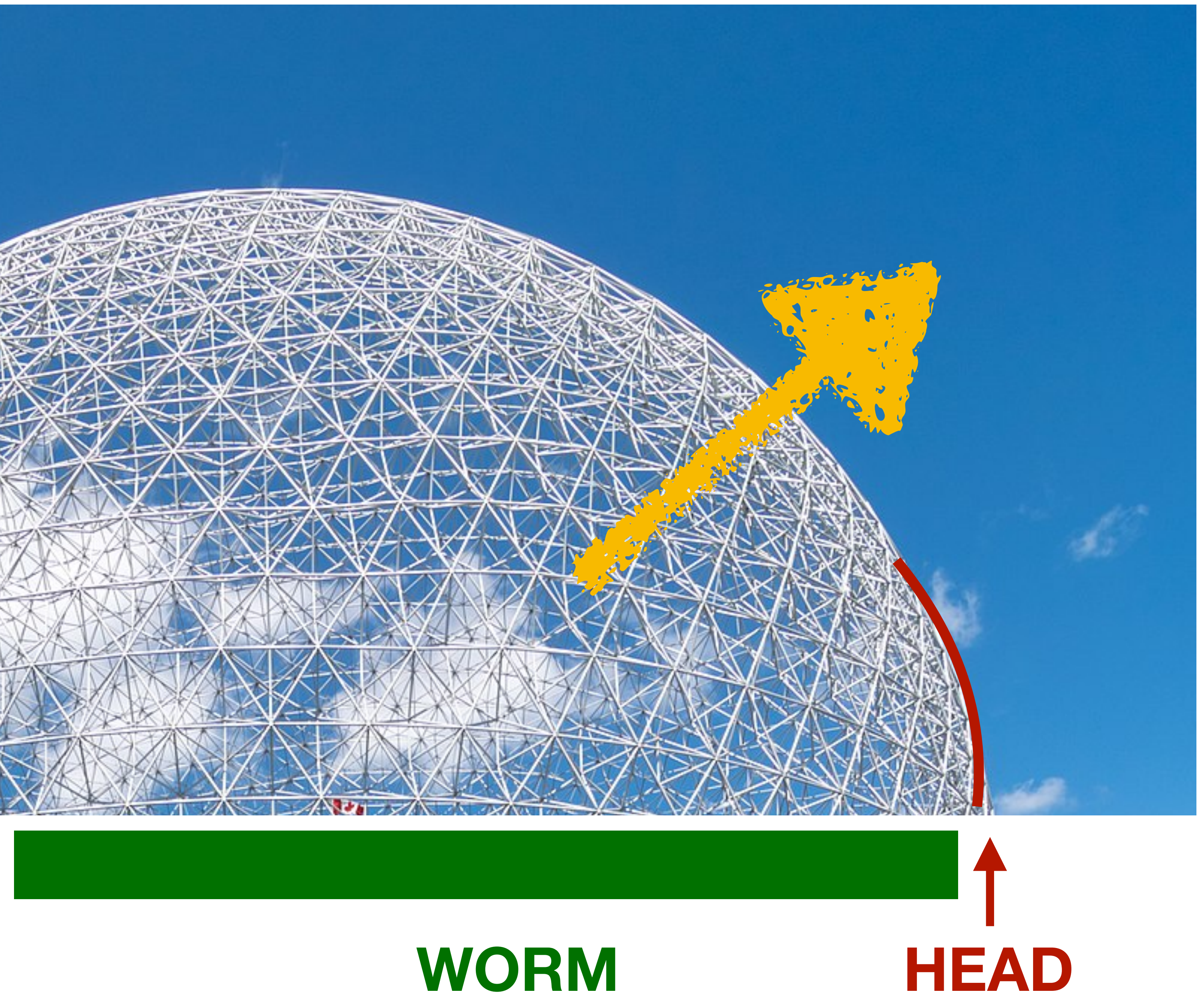
Universe of all content items (to be folded onto the hash’s image)

Black set (hash-named content) grows over time, has to spread in space:



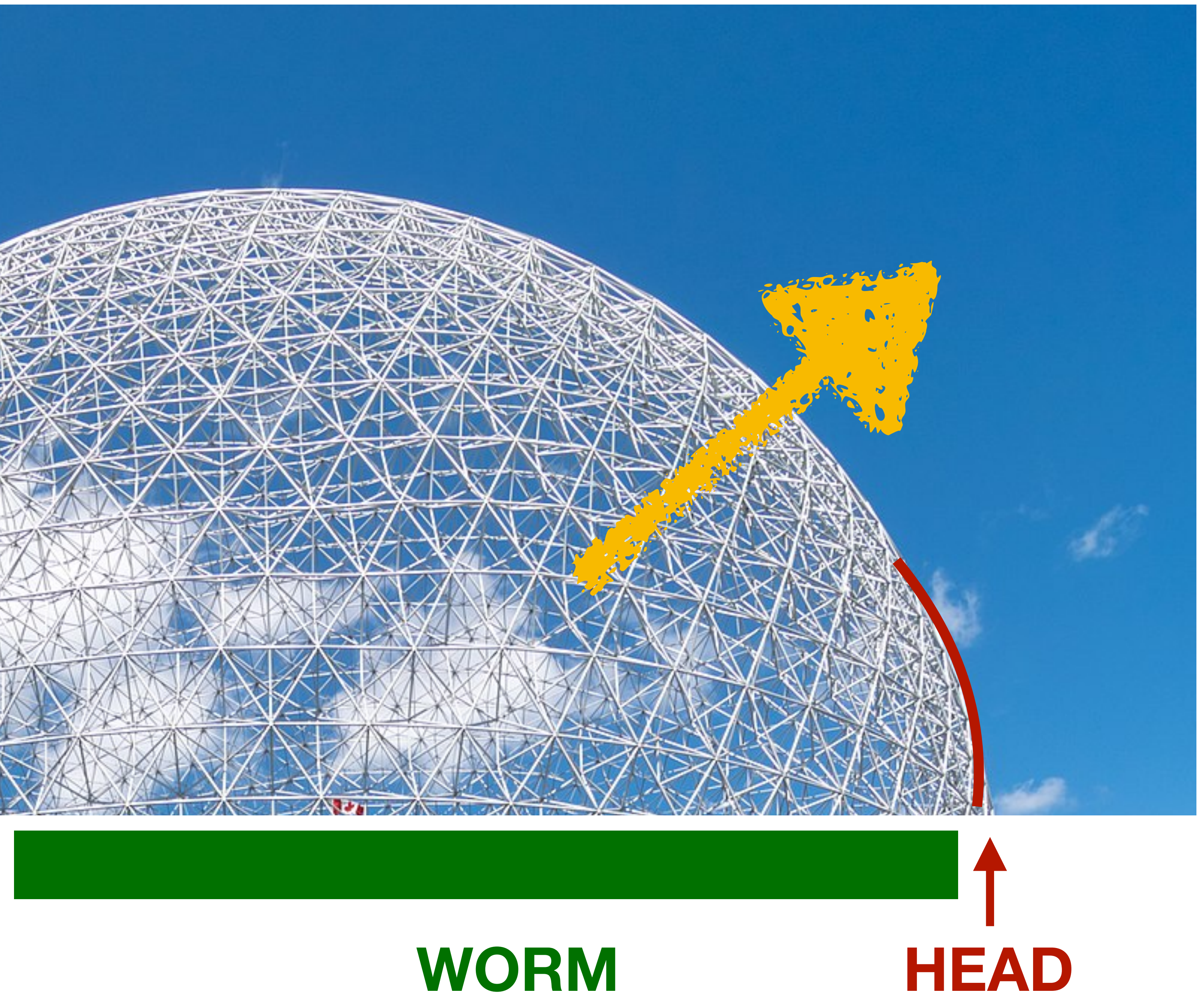
1’) Cumulative Immutable Content

- Growing set of hash-named content
- Accumulation so far: **WORM** (write-one-read-many)
- Frontier: **HEAD** (à la Git, also called “tips” in IOTA)



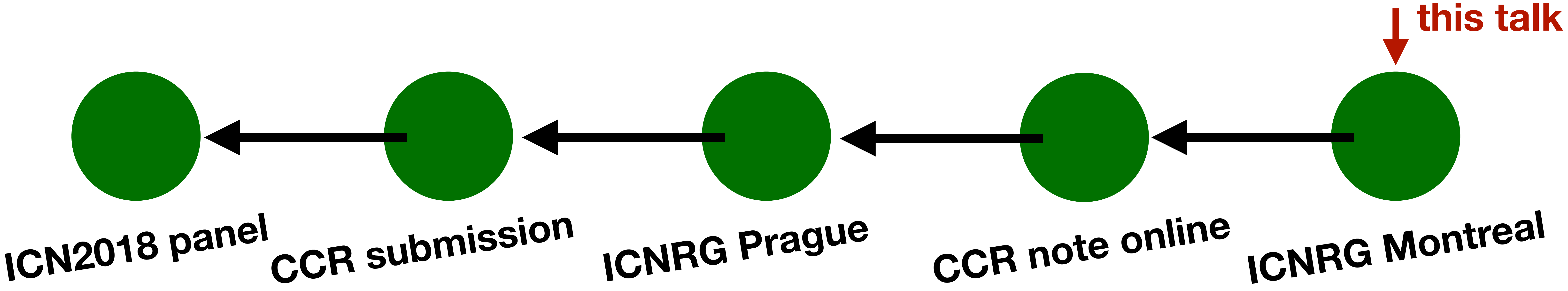
1') Cumulative Immutable Content

- Growing set of hash-named content
- Accumulation so far: **WORM** (write-one-read-many)
- Frontier: **HEAD** (à la Git, also called "tips" in IOTA)
- Prototypical WORM: hash-chain (append-only log)
  prototypical HEAD: hash (or seq#) of newest entry



1’) Cumulative Immutable Content

- Growing set of hash-named content
- Accumulation so far: **WORM** (write-one-read-many)
- Frontier: **HEAD** (à la Git, also called “tips” in IOTA)
- Prototypical WORM: hash-chain (append-only log) prototypical HEAD: hash (or seq#) of newest entry

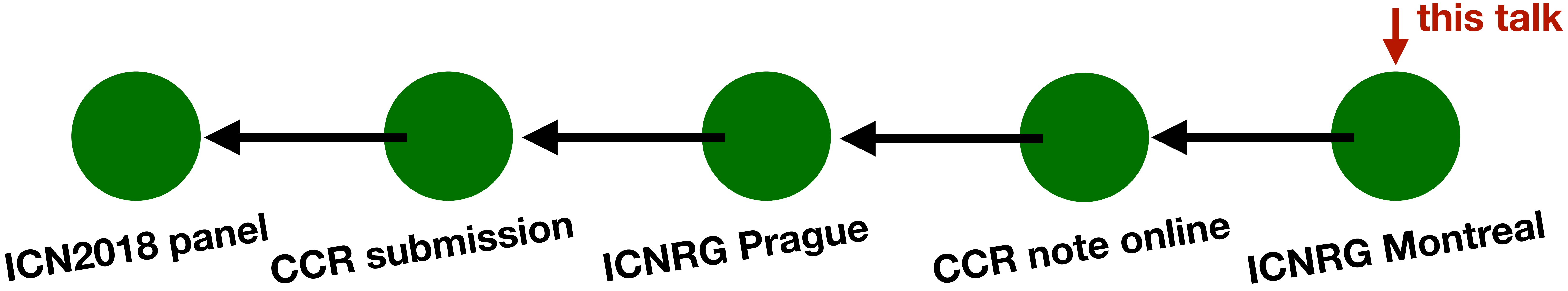


1’) Cumulative Immutable Content

- Growing set of hash-named content
- Accumulation so far: **WORM** (write-one-read-many)
- Frontier: **HEAD** (à la Git, also called “tips” in IOTA)
- Prototypical WORM: hash-chain (append-only log)
  prototypical HEAD: hash (or seq#) of newest entry

WORM

HEAD



this talk

ICN2018 panel

CCR submission

ICNRG Prague

CCR note online

ICNRG Montreal

- "Single-author append-only event log”: SSB’s basis

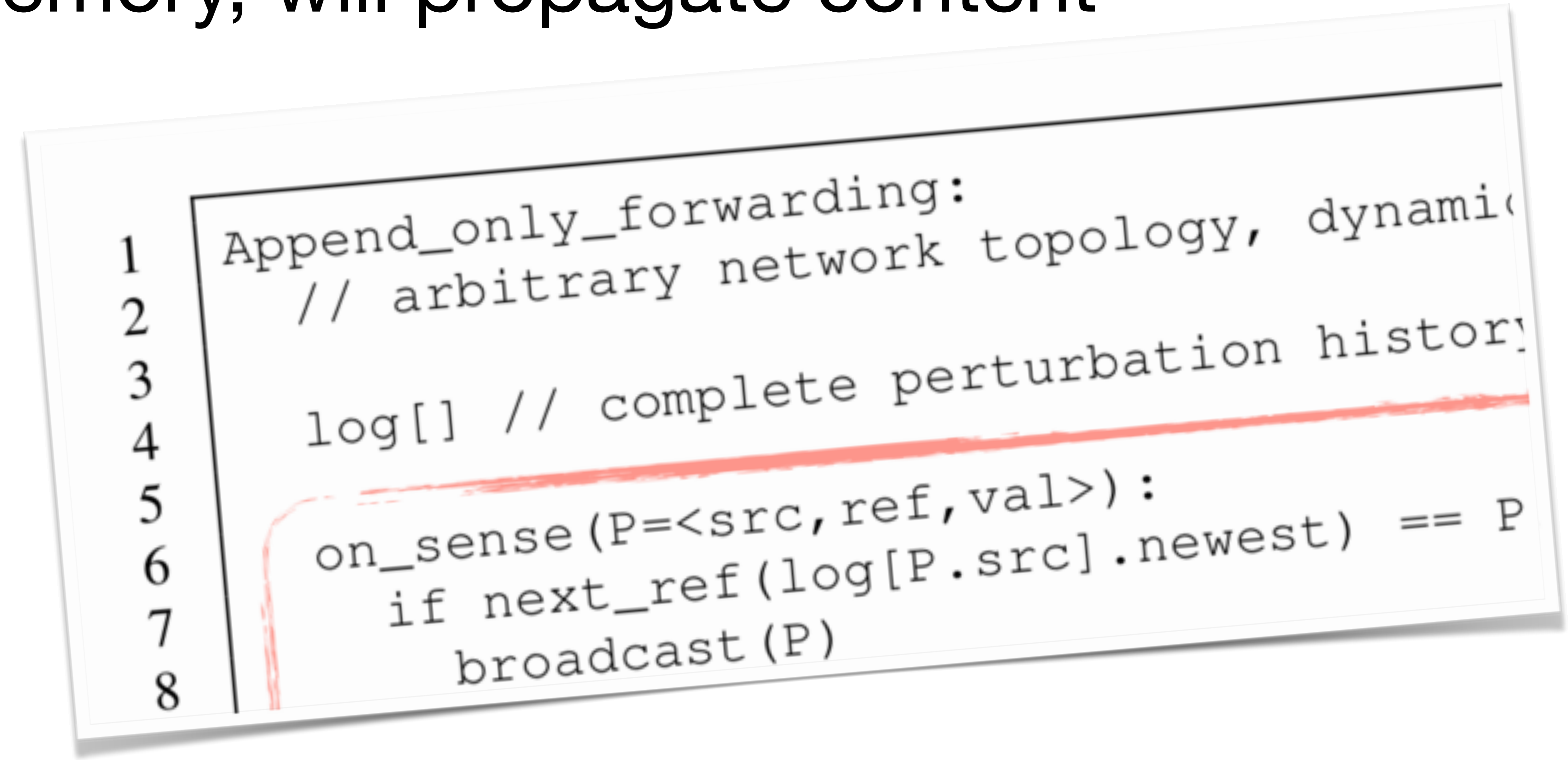
2) Broadcast-only communication

- **Reliable global broadcast**: desirable networking service, used in: secure scuttlebutt, cert transparency (CT), Google Pub/Sub, Amazon SNS

- Global broadcast must be built from local broadcast range (due to limited reach, but also to handle offline situations)

- Global broadcast needs relays with memory, will propagate content as soon as possible, and only once

- Append-only log "induced" by global broadcast networking task

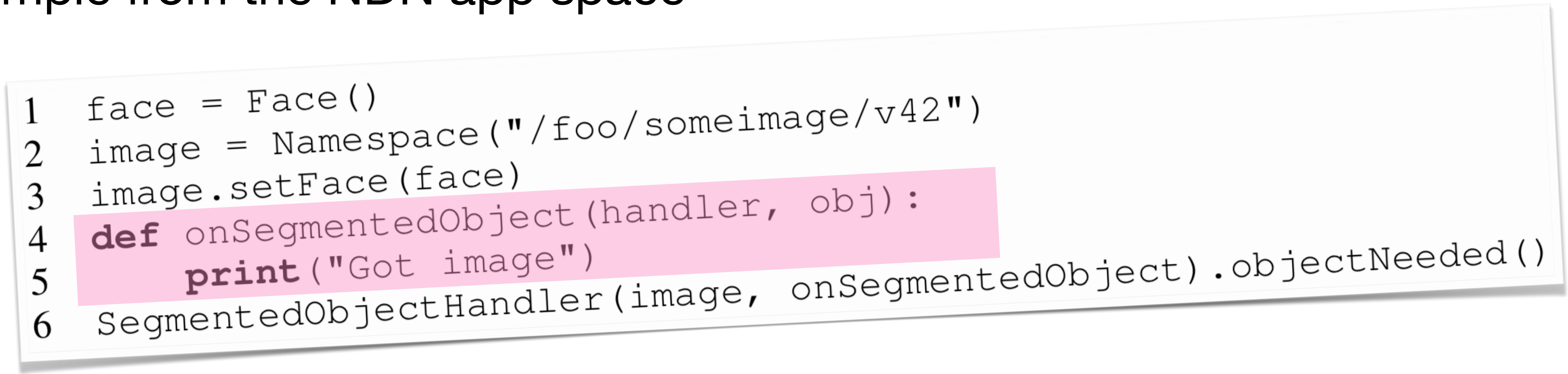


```
Append_only_forwarding:
  // arbitrary network topology, dynami

  log[] // complete perturbation histor

  on_sense(P=<src,ref,val>):
    if next_ref(log[P.src].newest) == P
      broadcast(P)
```

3) The need for global push()

- Known under many names: SYNC, NOTIFY, PUB/SUB…

- Recent example from the NDN app space



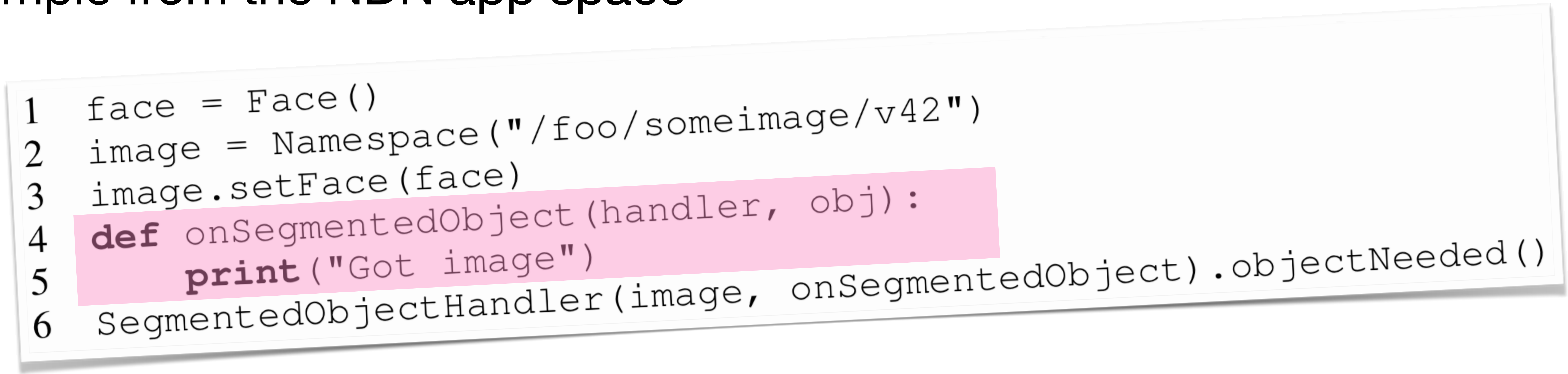
```
face = Face()
image = Namespace("/foo/someimage/v42")
image.setFace(face)
def onSegmentedObject(handler, obj):
    print("Got image")
SegmentedObjectHandler(image, onSegmentedObject).objectNeeded()
```

- **Callback handlers** as a prevailing coding style, triggered by some notification.

3) The need for global push()

- Known under many names: SYNC, NOTIFY, PUB/SUB…

- Recent example from the NDN app space



```
face = Face()
image = Namespace("/foo/someimage/v42")
image.setFace(face)
def onSegmentedObject(handler, obj):
    print("Got image")
SegmentedObjectHandler(image, onSegmentedObject).objectNeeded()
```

- **Callback handlers** as a prevailing coding style, triggered by some notification.

- Q: How to implement the notification? —> Pub/Sub library over pull-based ICN, long-lived interest… But once satisfied, things become "interesting”

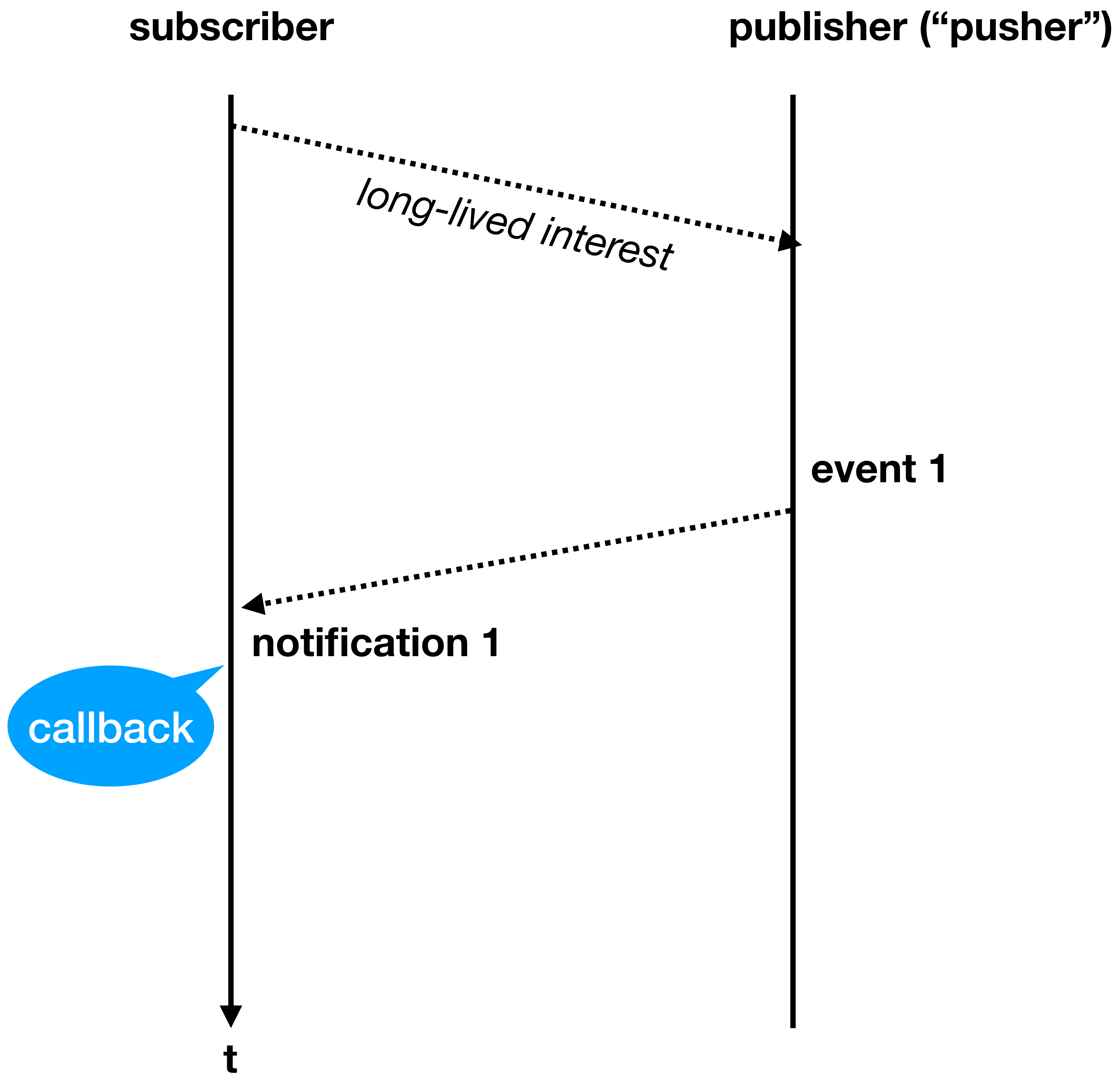
4) Two problems of emulated push()

- Problem 0:
  routing to multiple replicas, getting the freshest content fastest
  … see ICNRG Prague talk, case still needs to be written up,
  basically can only be solved by (global) interest flooding …

- Problem 1: pull leads to “inter-notification gap >= RTT”

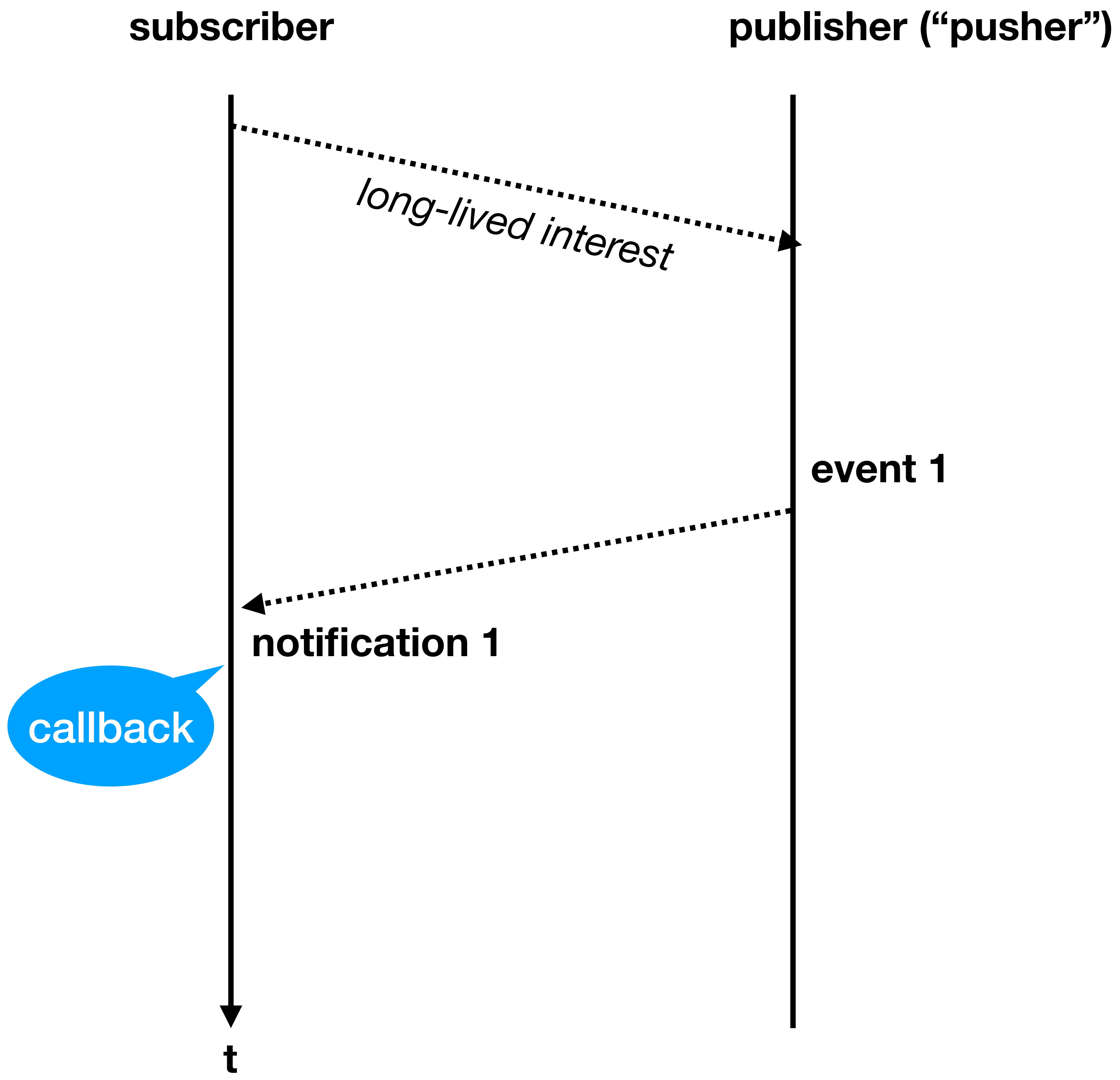
- Problem 2: pull leads to “recursion corridors"

4a) Inter-notification gap >= RTT



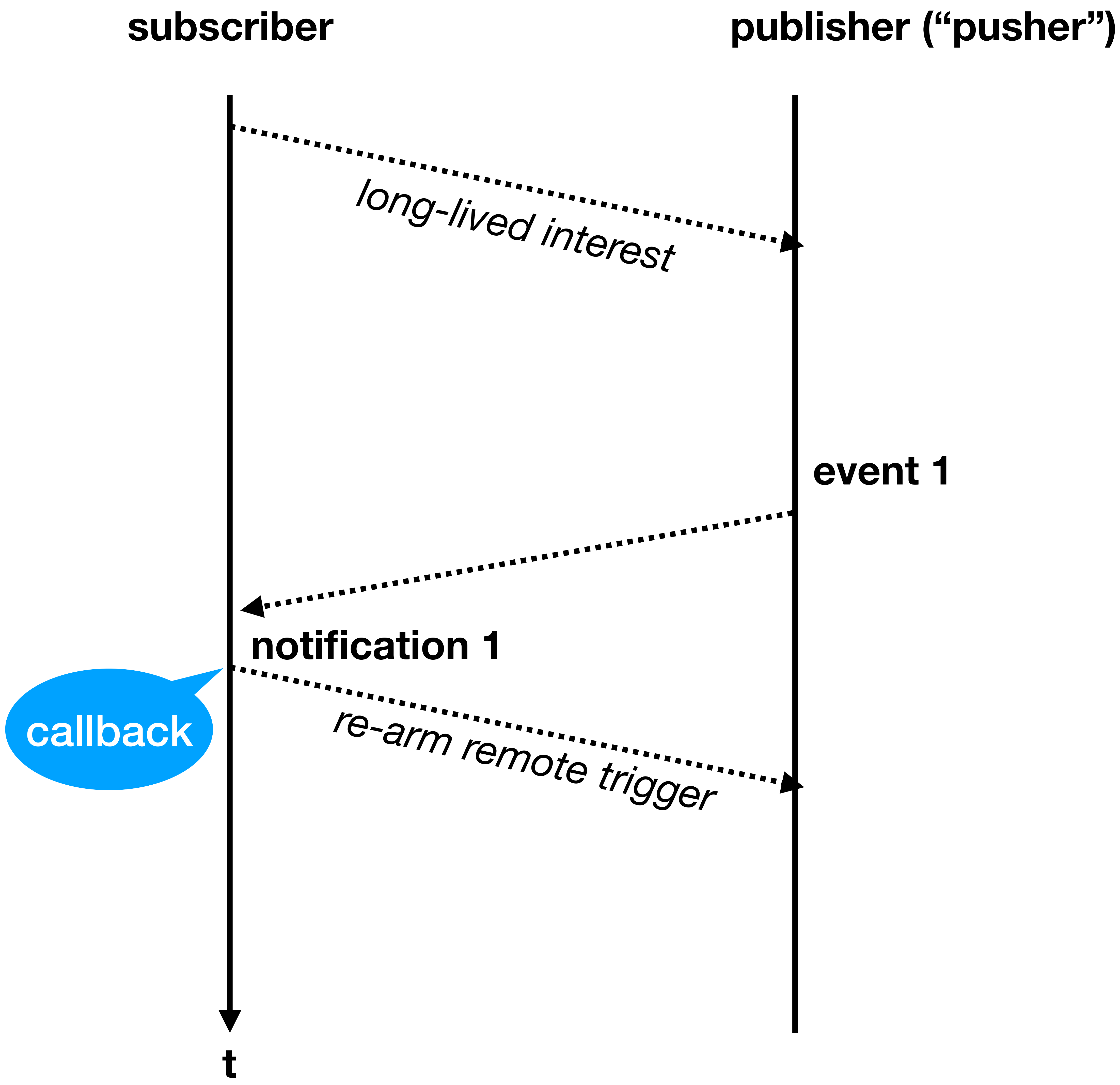
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler

4a) Inter-notification gap >= RTT



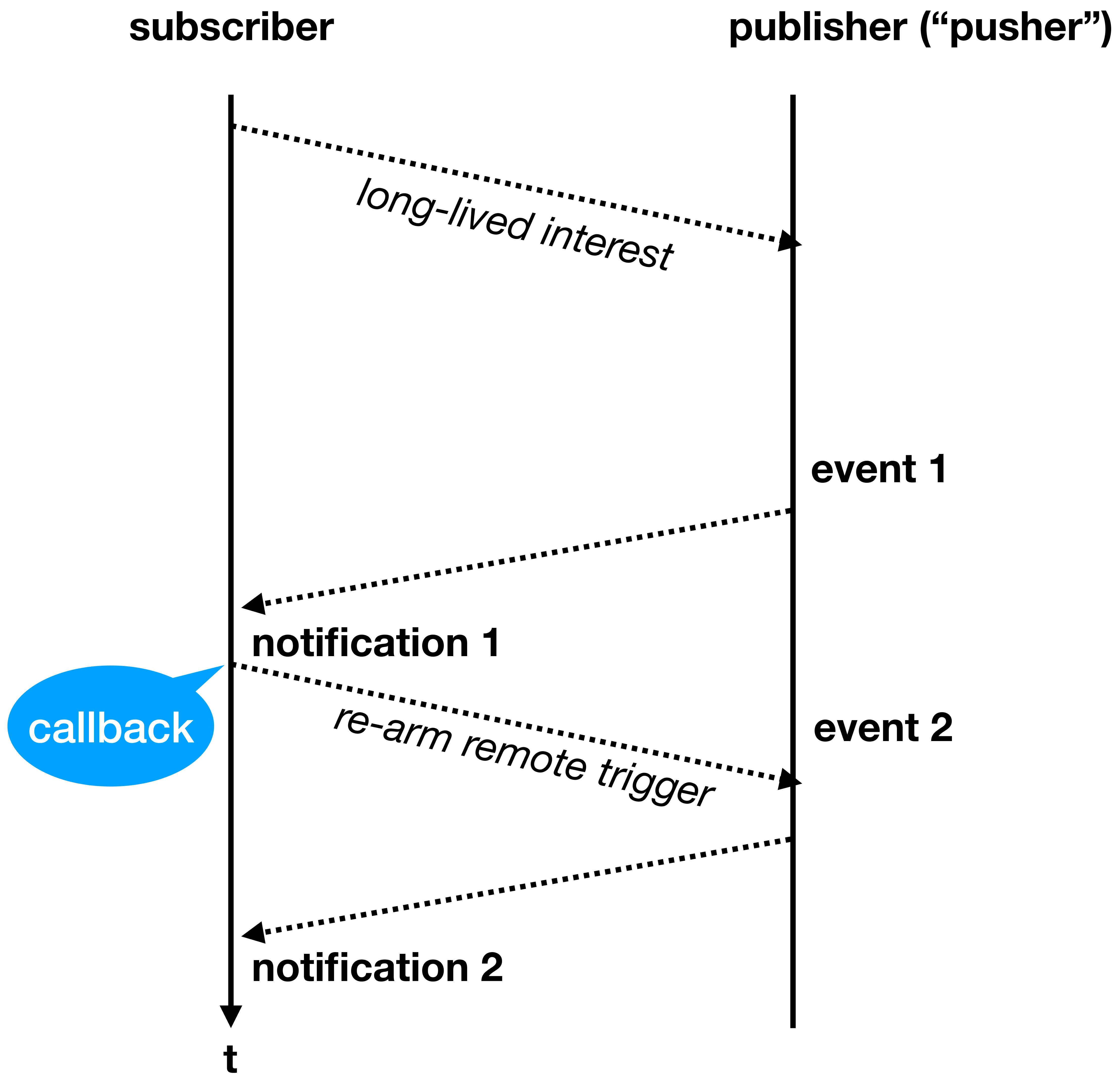
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm

4a) Inter-notification gap >= RTT



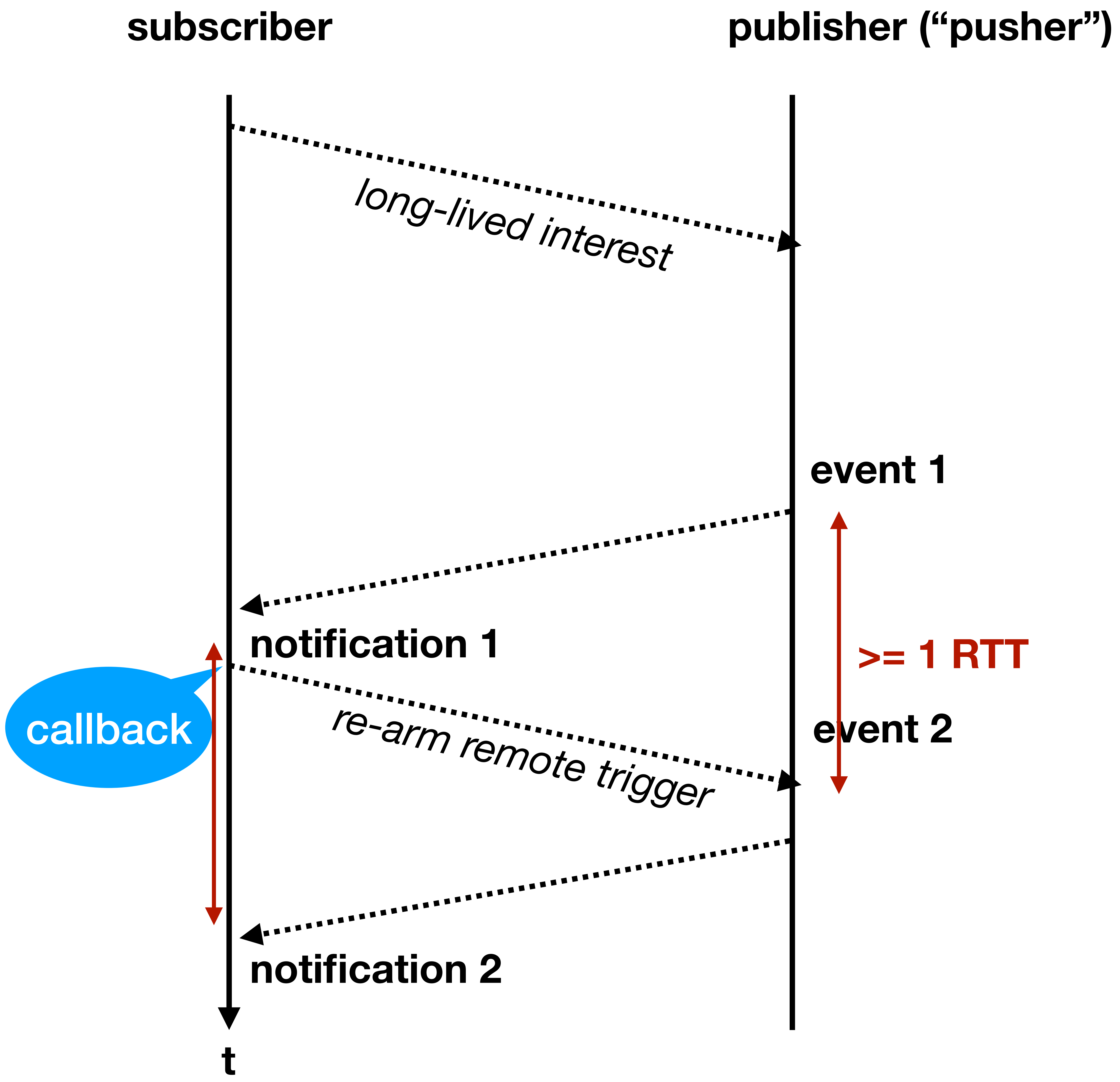
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm

4a) Inter-notification gap >= RTT



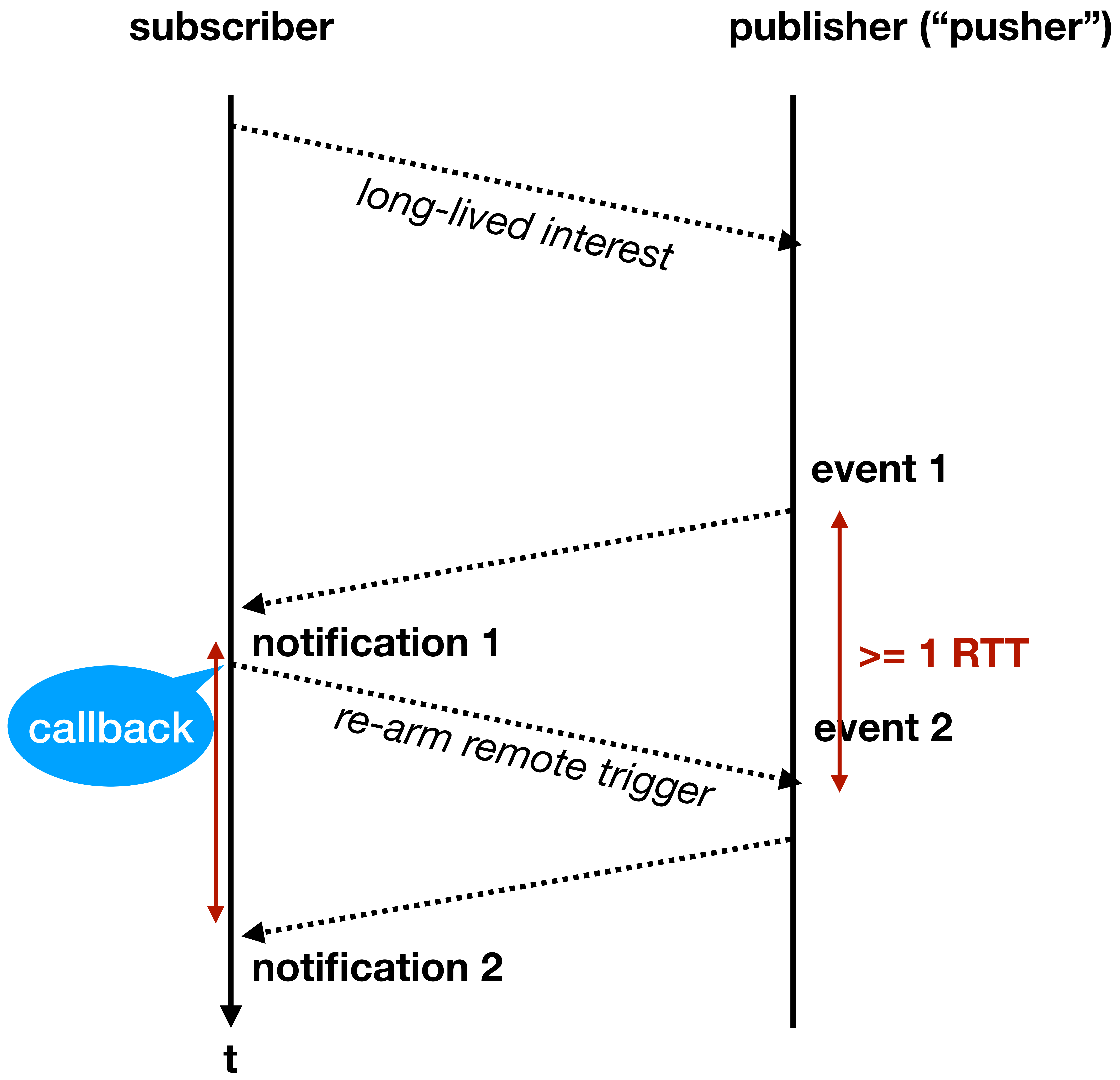
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm

4a) Inter-notification gap >= RTT



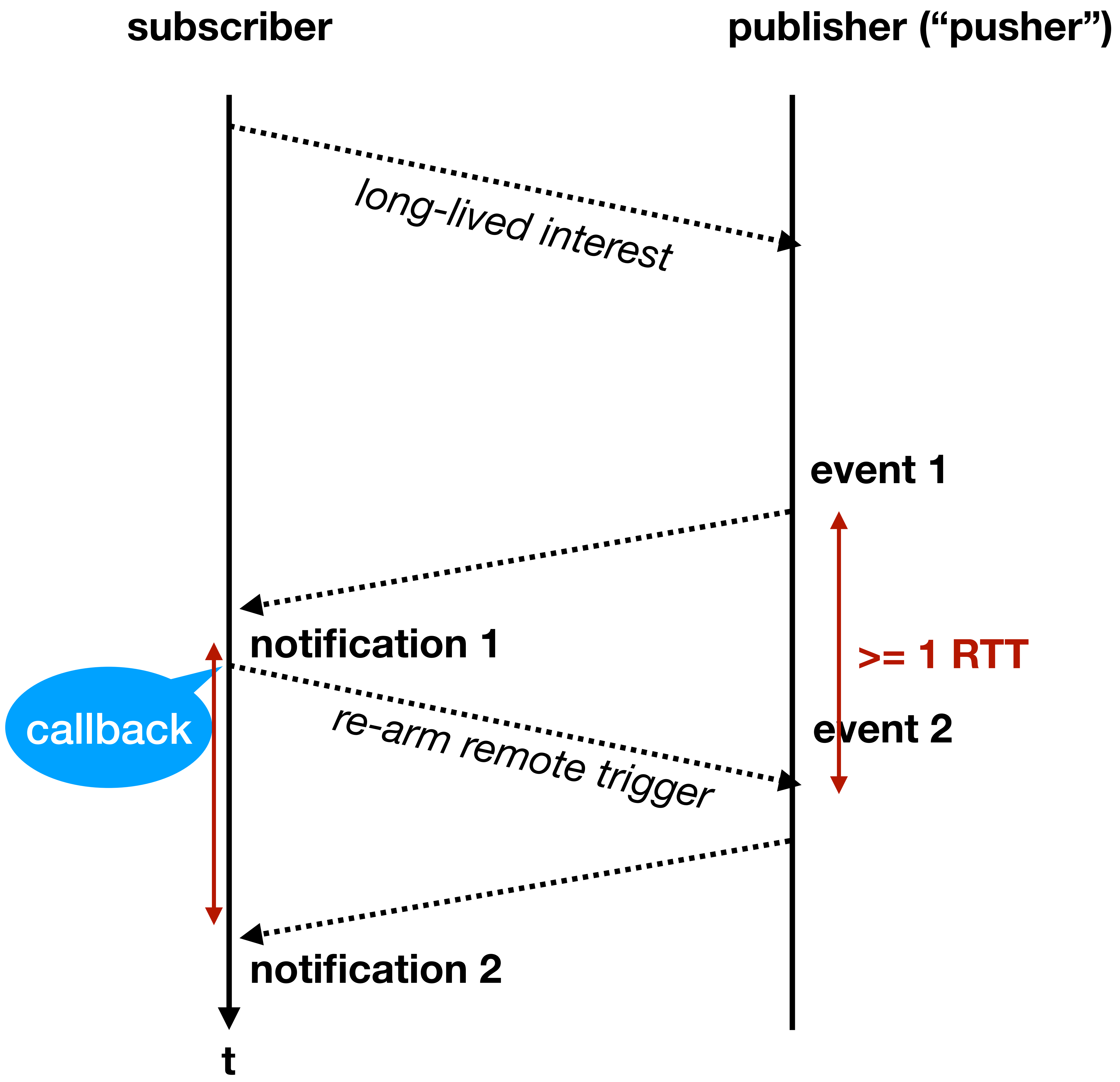
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm

4a) Inter-notification gap >= RTT



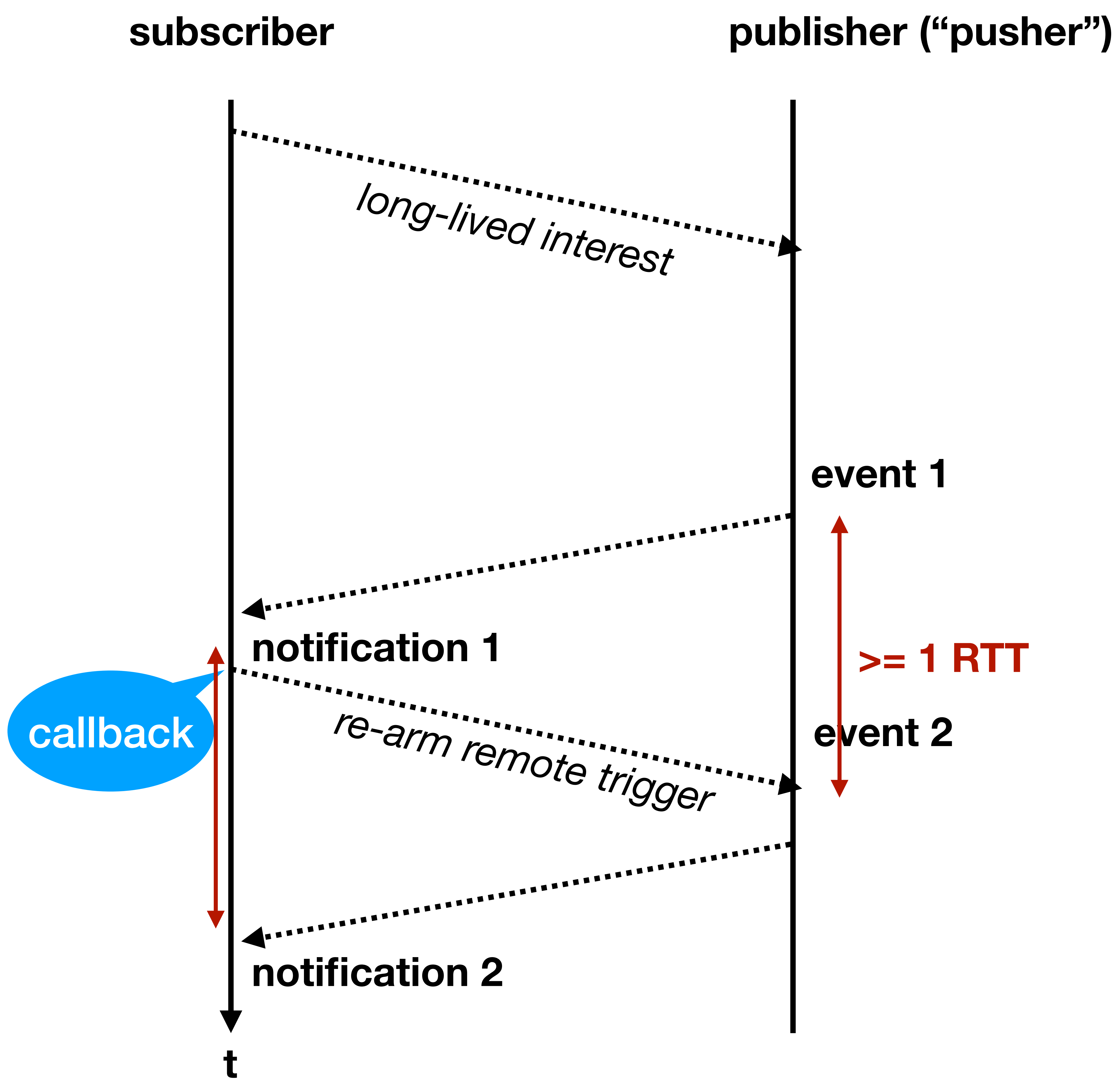
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm
- leads to >= 1 RTT inter-notification gap —> this is a rate limiter

4a) Inter-notification gap >= RTT



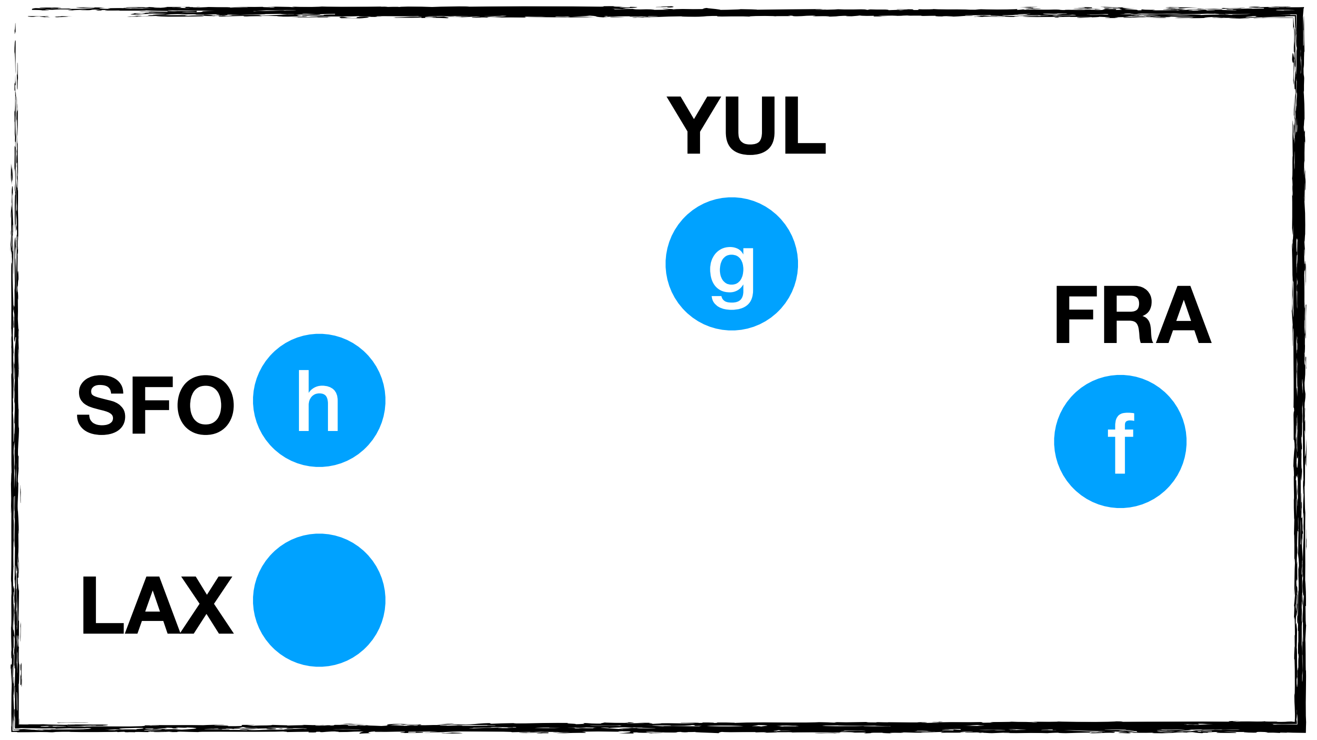
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm
- leads to >= 1 RTT inter-notification gap —> this is a rate limiter
- Moreover, could loose events:
  - events *during unarmed interval*
  - due to unreliable PULL

4a) Inter-notification gap >= RTT



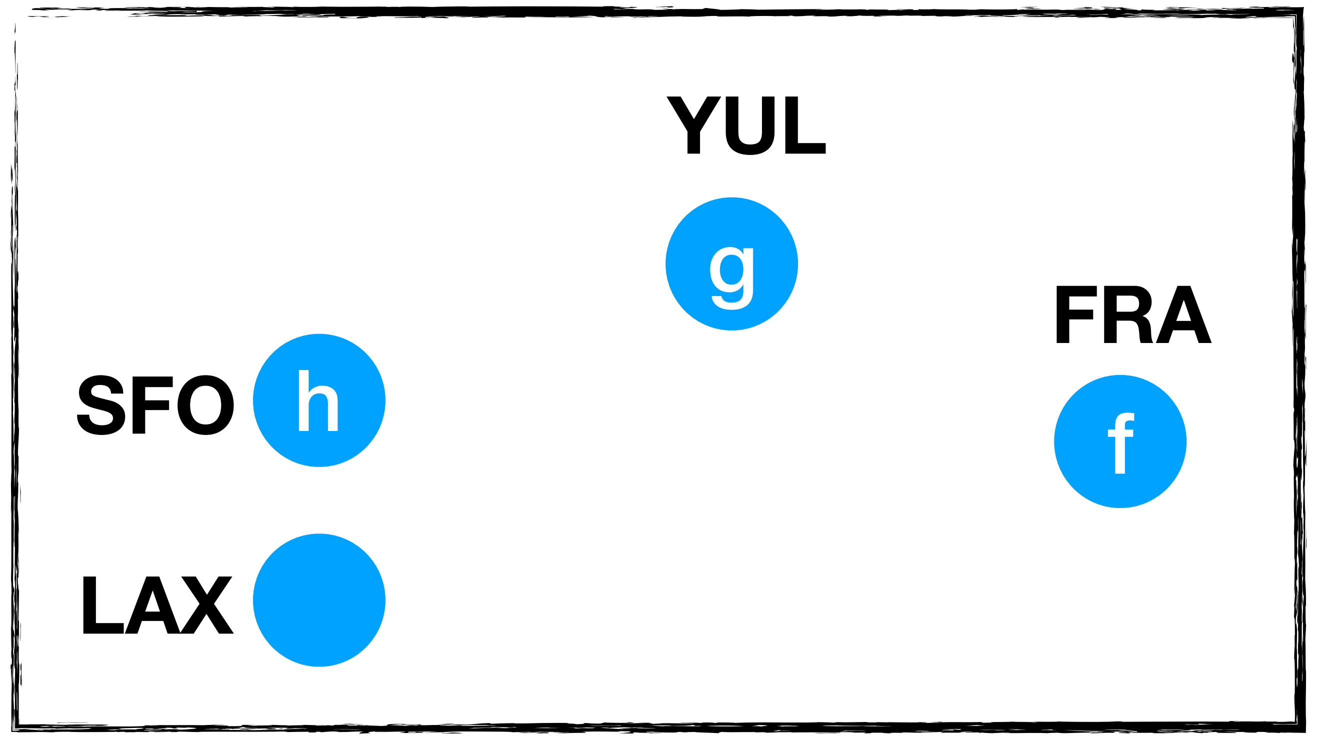
- Emulating app-level PUSH with network-level PULL
- Long-lived interest used to "arm" a notification handler
- After the event: must re-arm
- leads to >= 1 RTT inter-notification gap —> this is a rate limiter
- Moreover, could loose events:
  - events *during unarmed interval*
  - due to unreliable PULL
- Protection via publisher-side queue (**log…**)

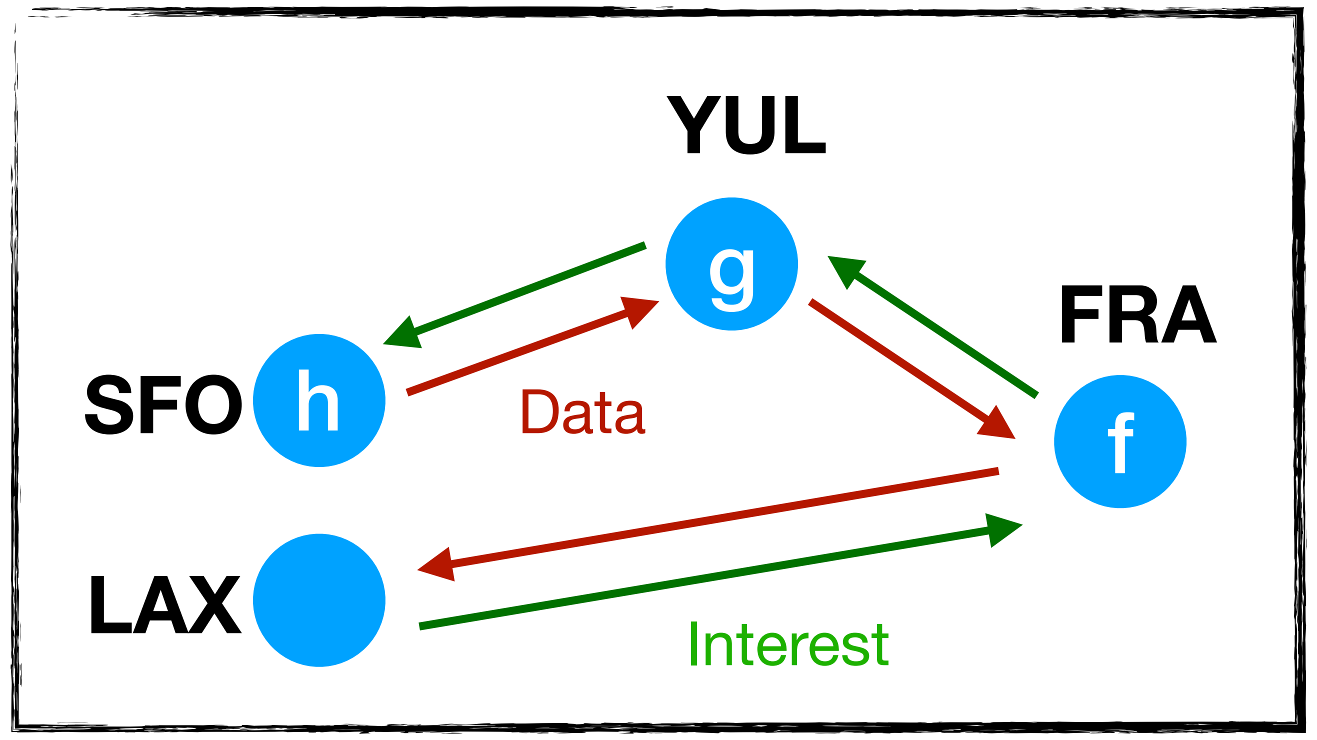
4b) Recursion corridors



- Context: COIN (compute-in-the-net)
- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX

4b) Recursion corridors

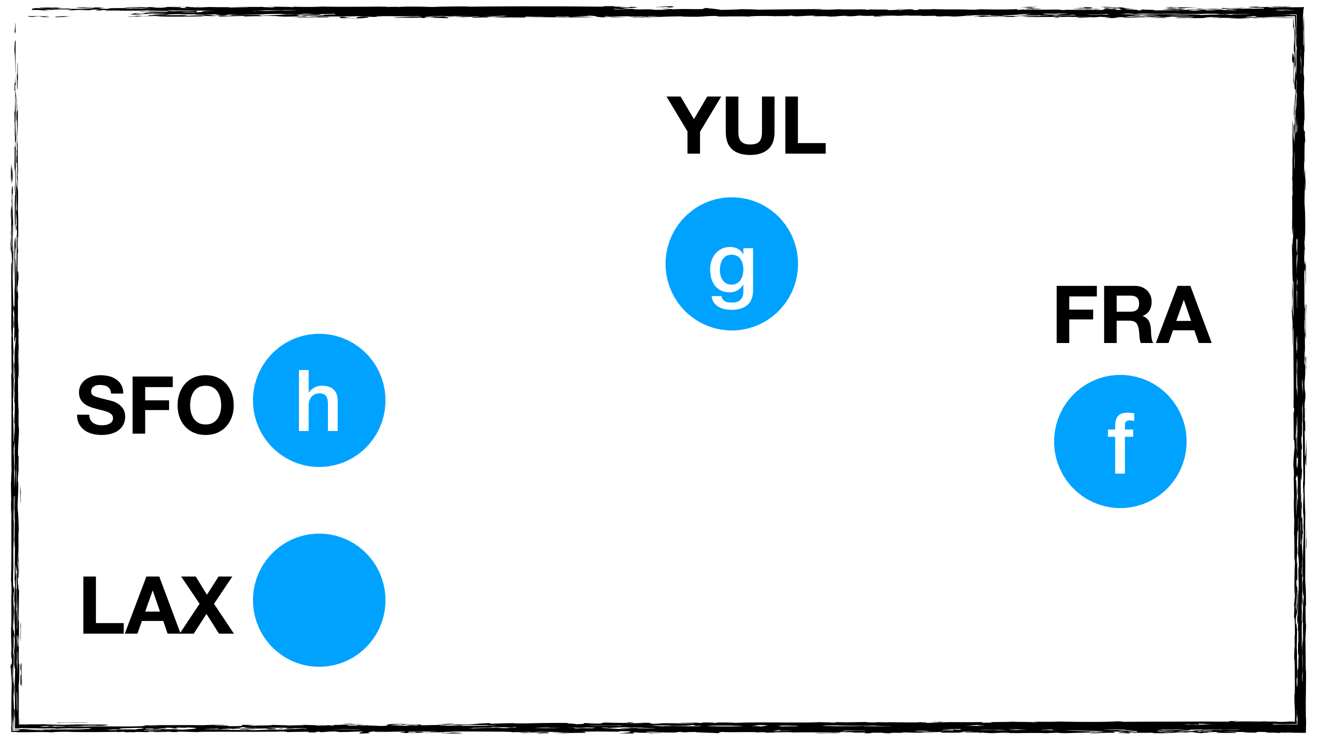


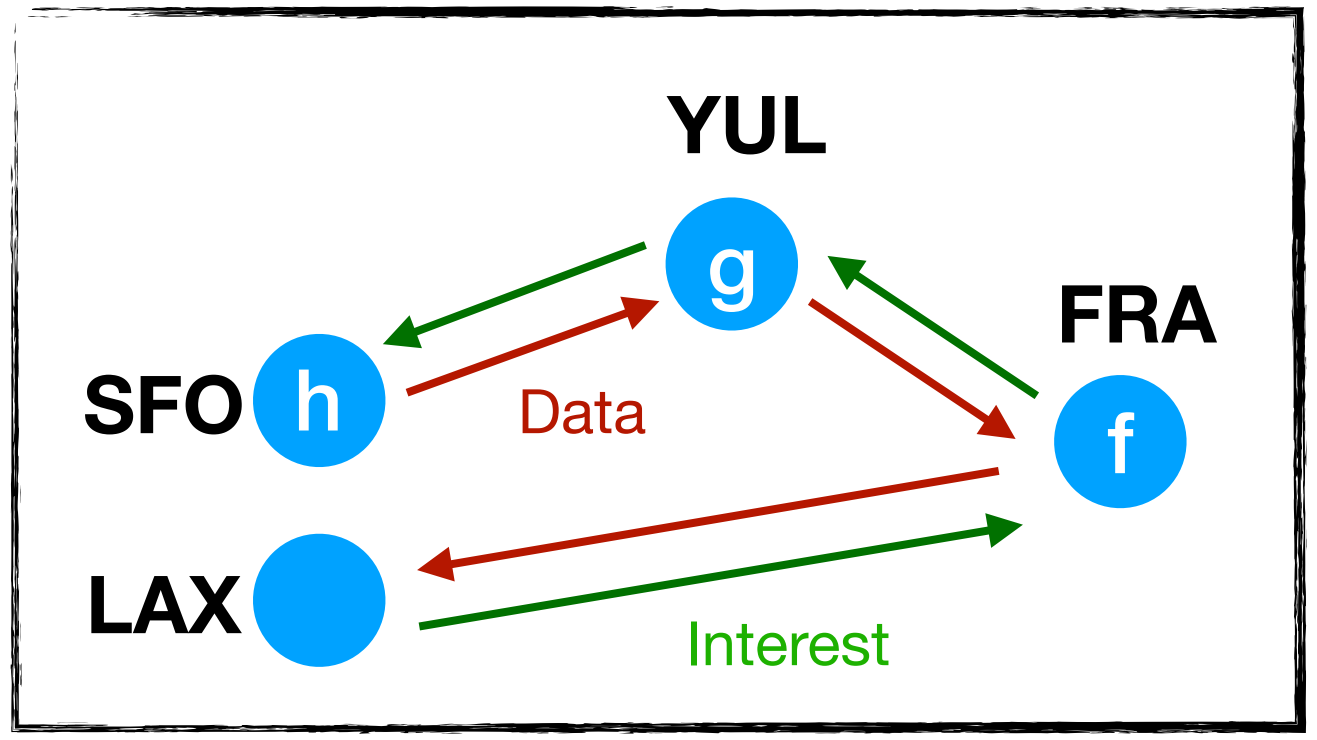


PULL-world

- Context: COIN (compute-in-the-net)
- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX

4b) Recursion corridors

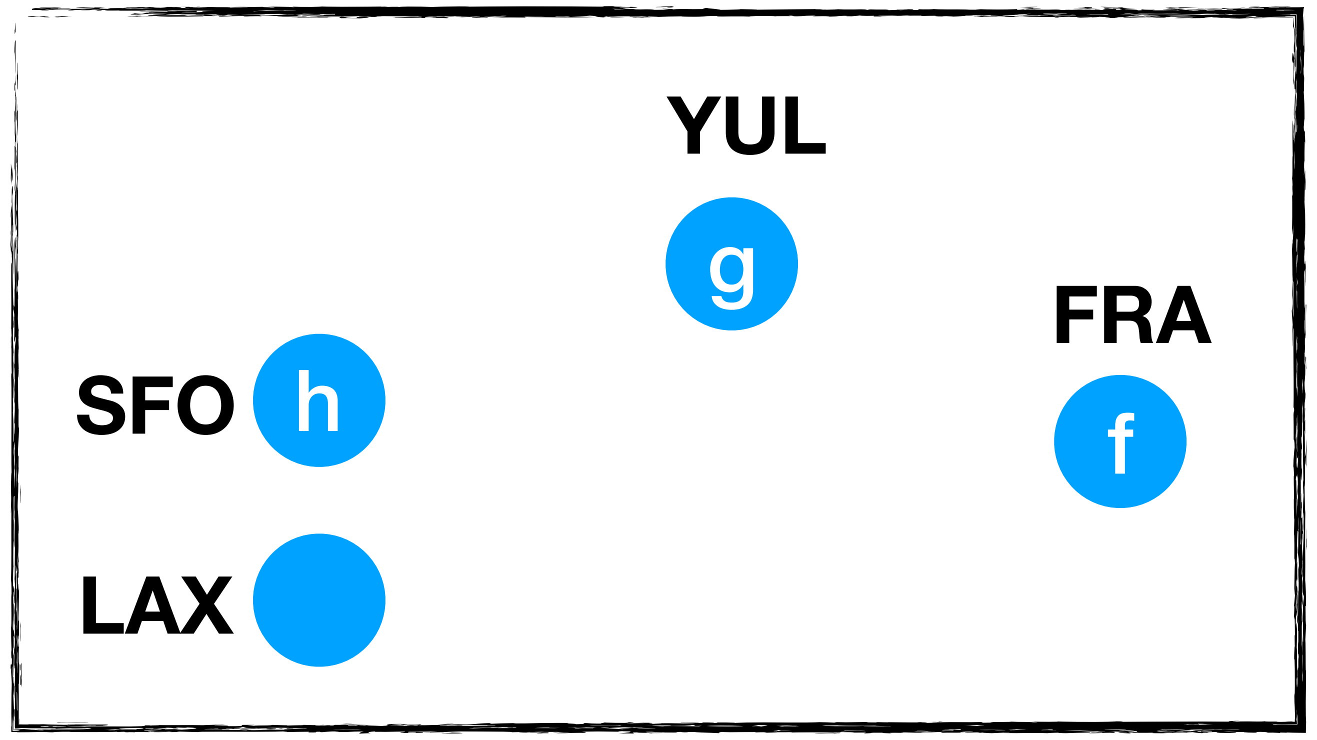


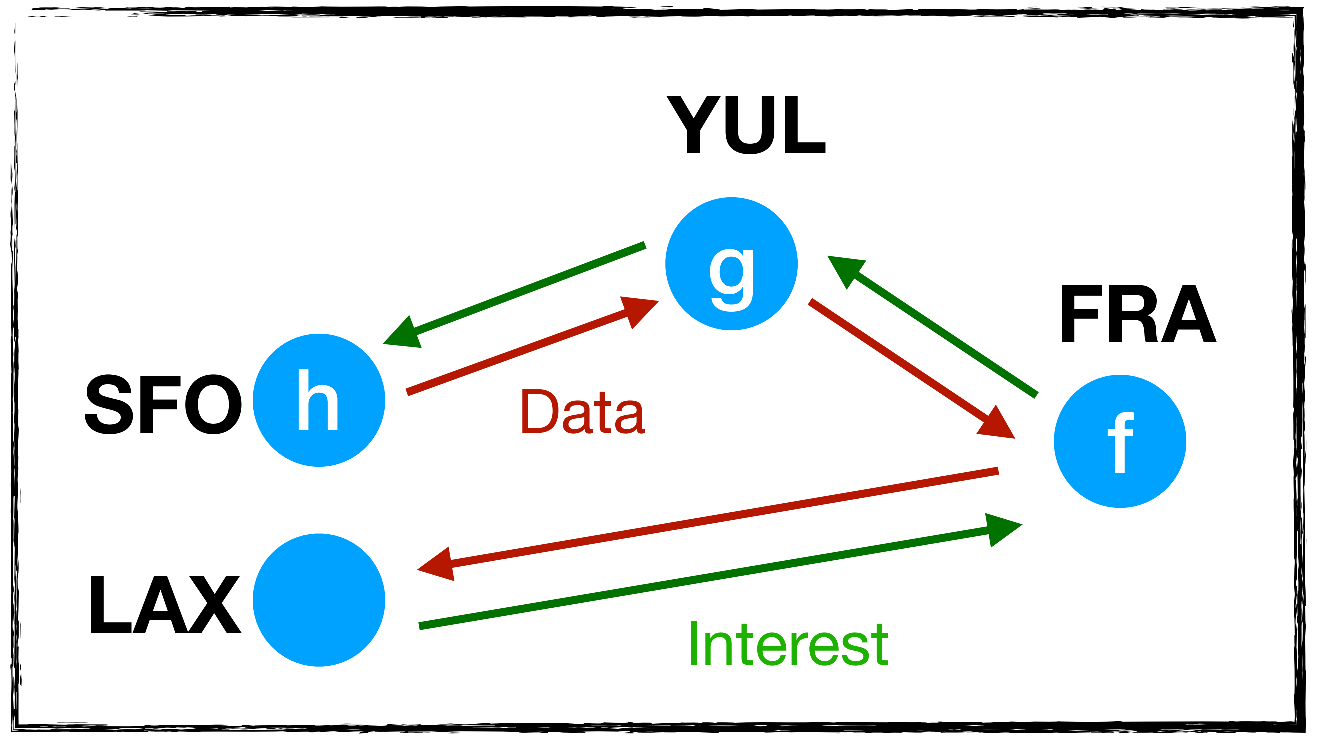


PULL-world

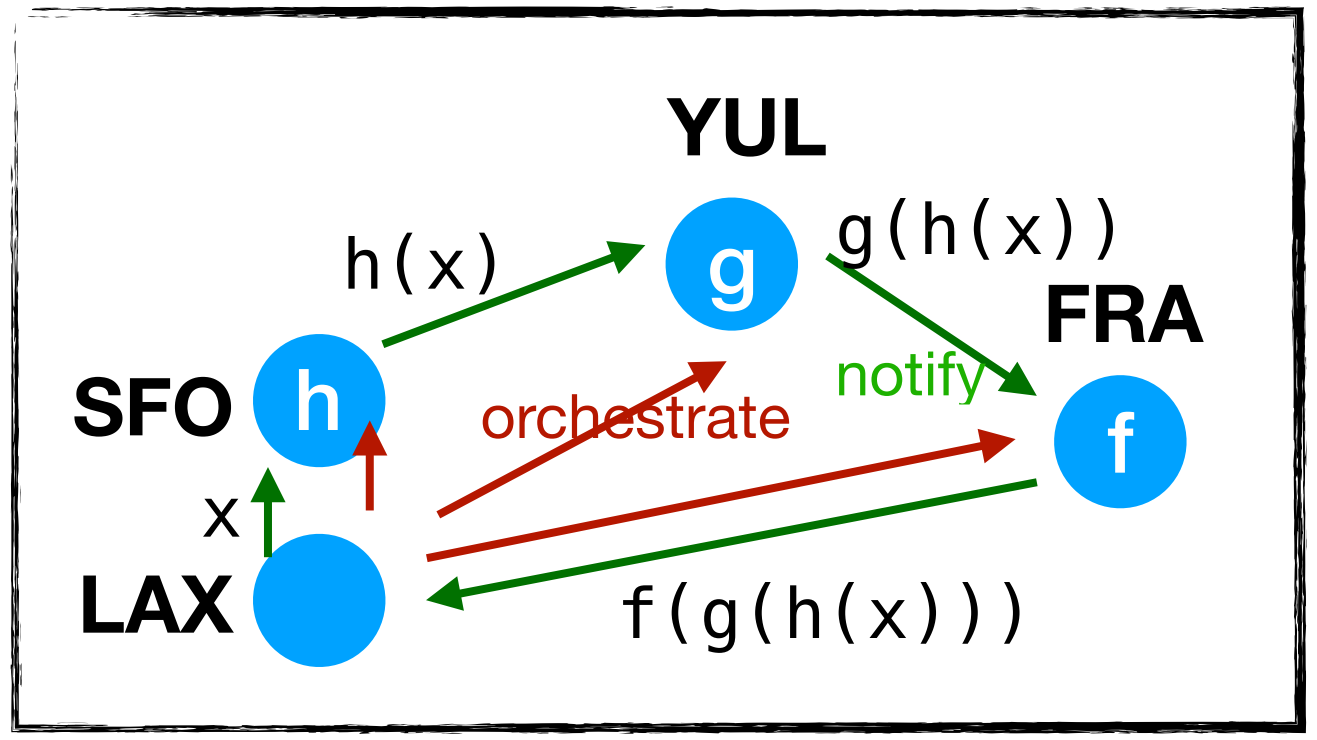
- Context: COIN (compute-in-the-net)
- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX
- PULL creates a "recursion corridor" (flashback: “mobile IP" and triangular routing…)

4b) Recursion corridors





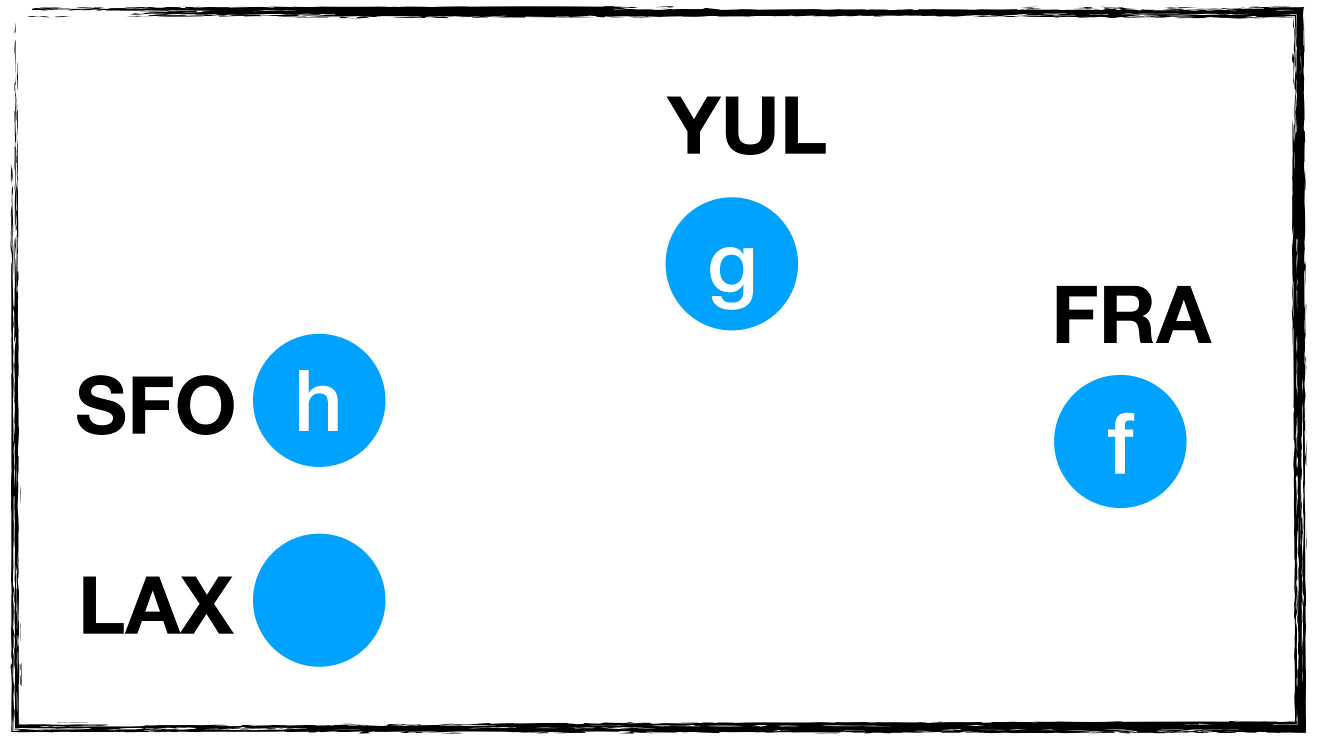
PULL-world

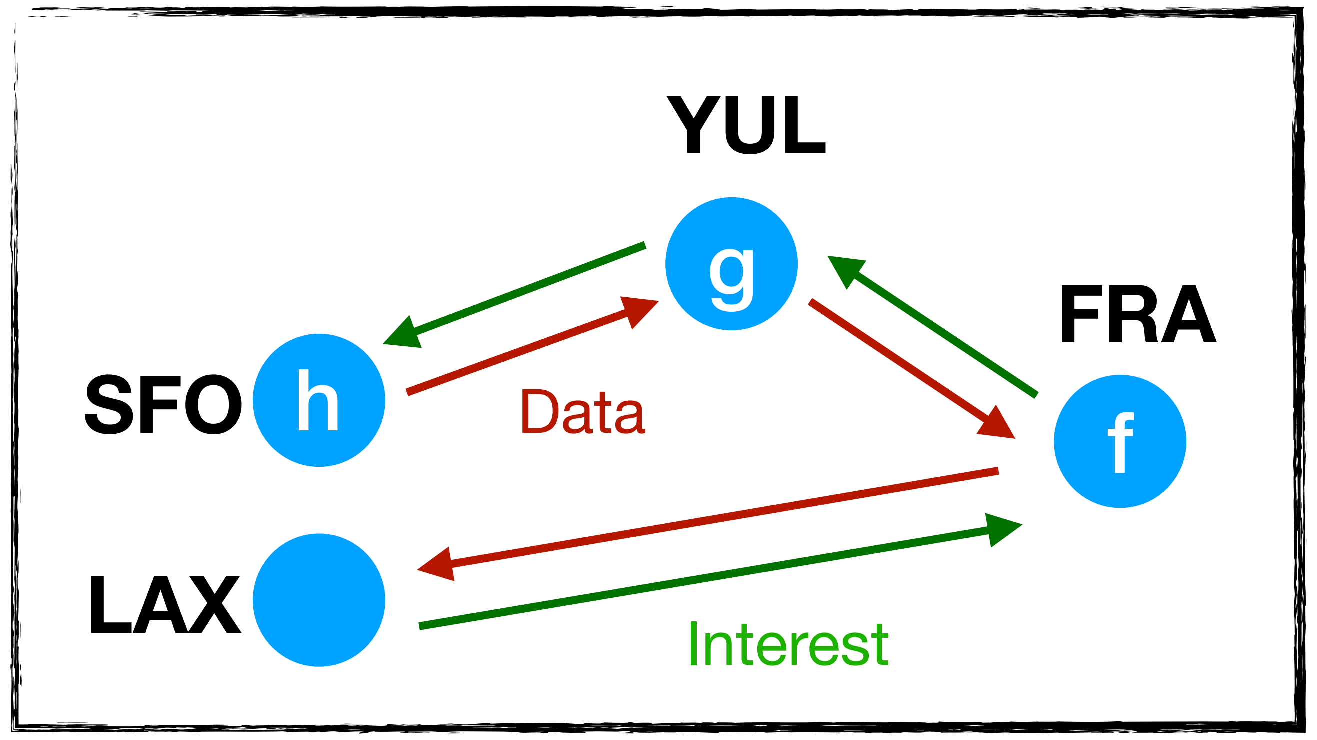


PUSH world

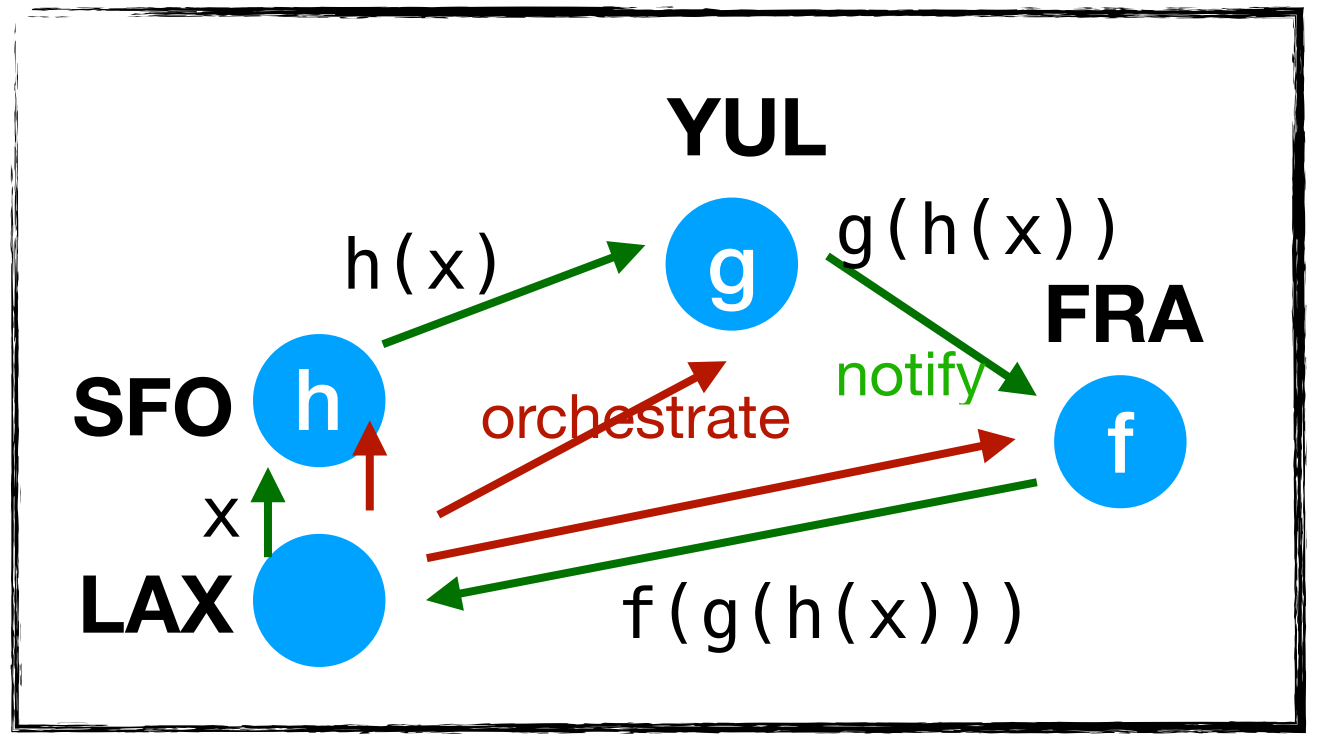
- Context: COIN (compute-in-the-net)
- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX
- PULL creates a "recursion corridor" (flashback: “mobile IP" and triangular routing…)

4b) Recursion corridors





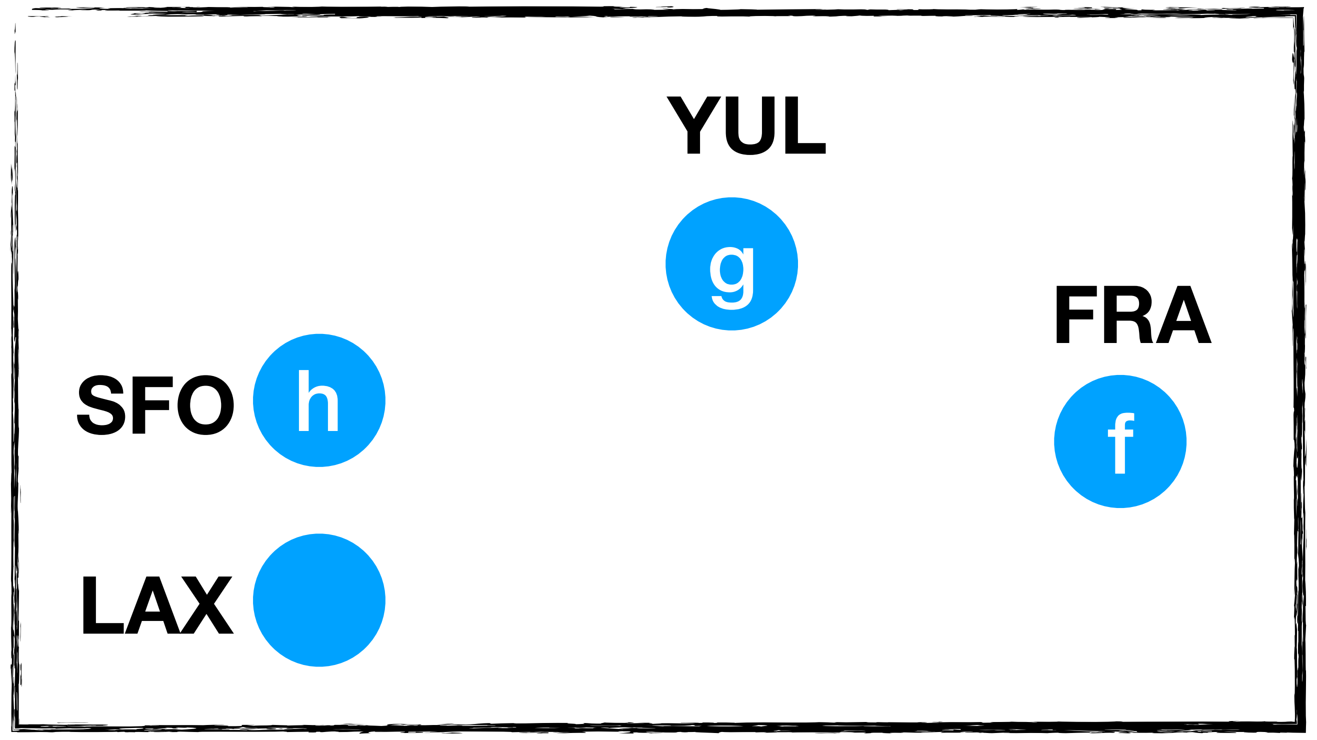
PULL-world

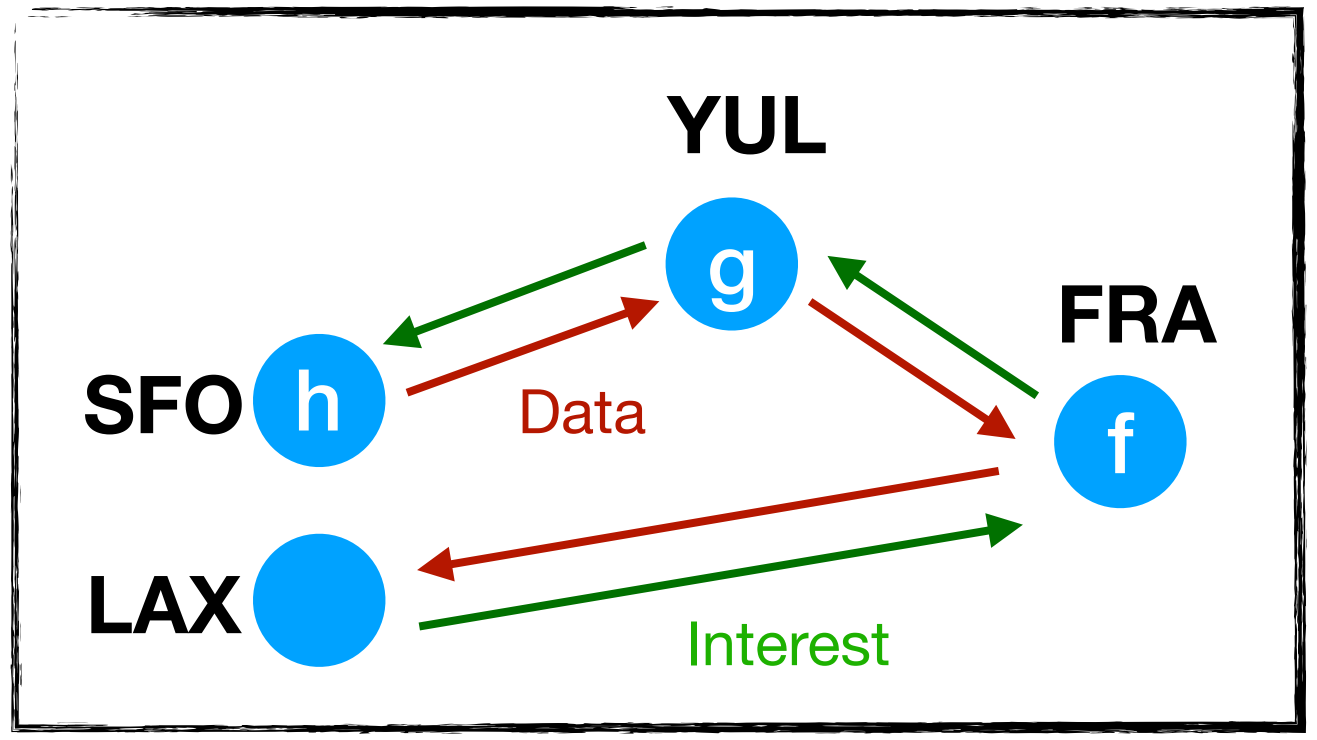


PUSH world

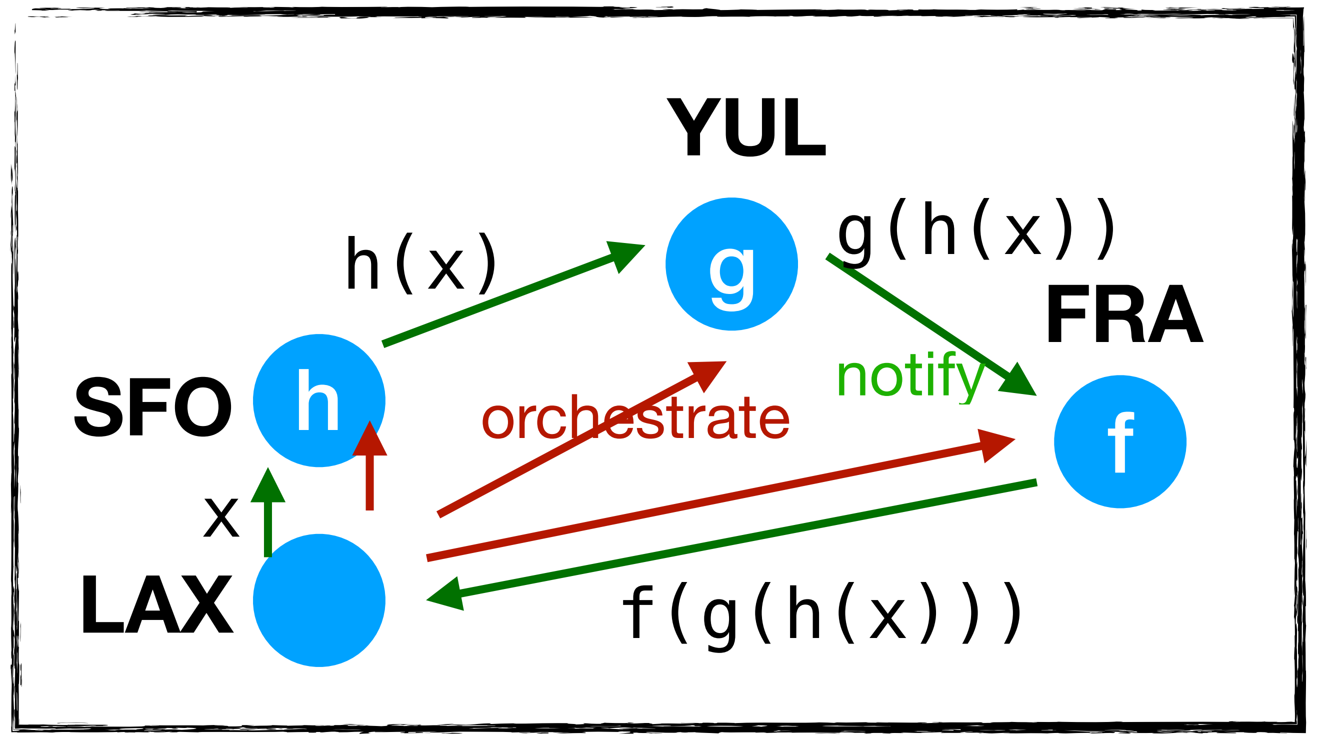
- Context: COIN (compute-in-the-net)
- Consider chain of calls **`f(g(h(x)))`** executed at FRA, YUL and SFO, requested from LAX
- PULL creates a "recursion corridor” (flashback: “mobile IP" and triangular routing…)
- In a PUSH world: only two Atlantic-crossings in the critical path (compared to four when using PULL)

4b) Recursion corridors





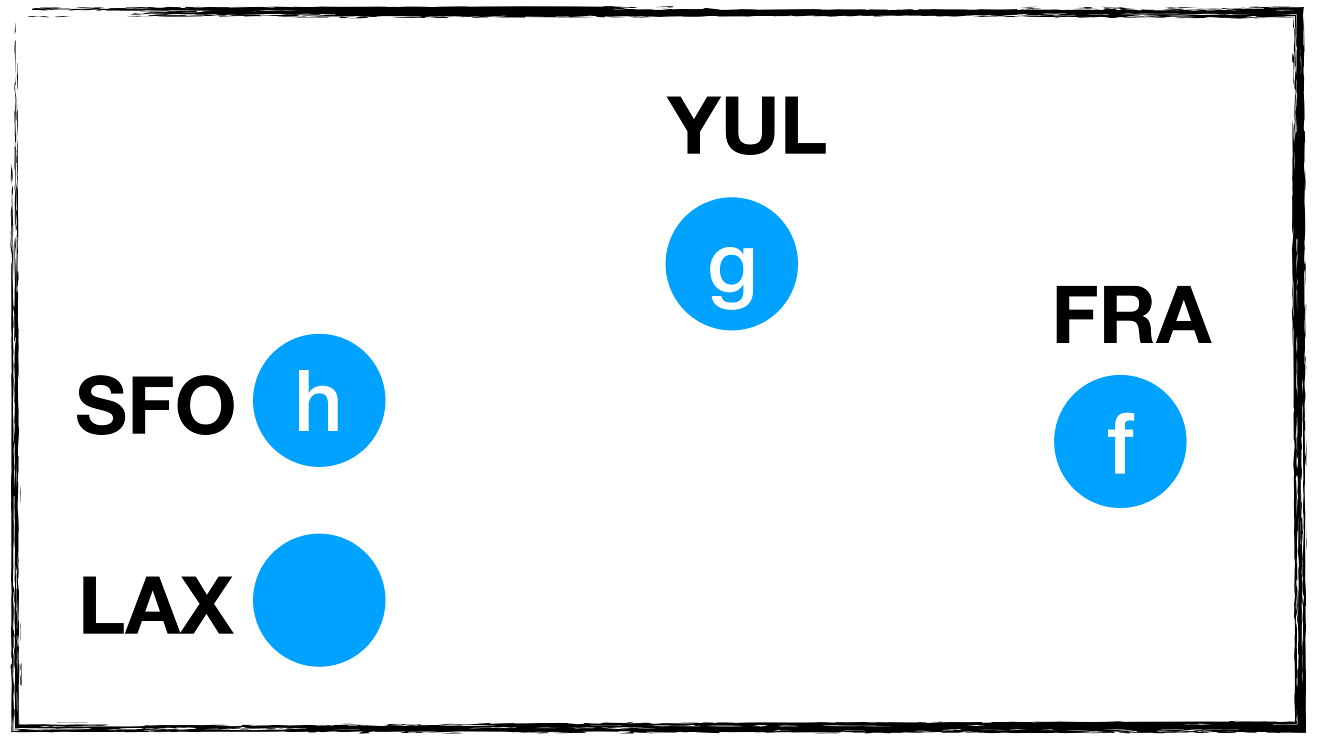
PULL-world

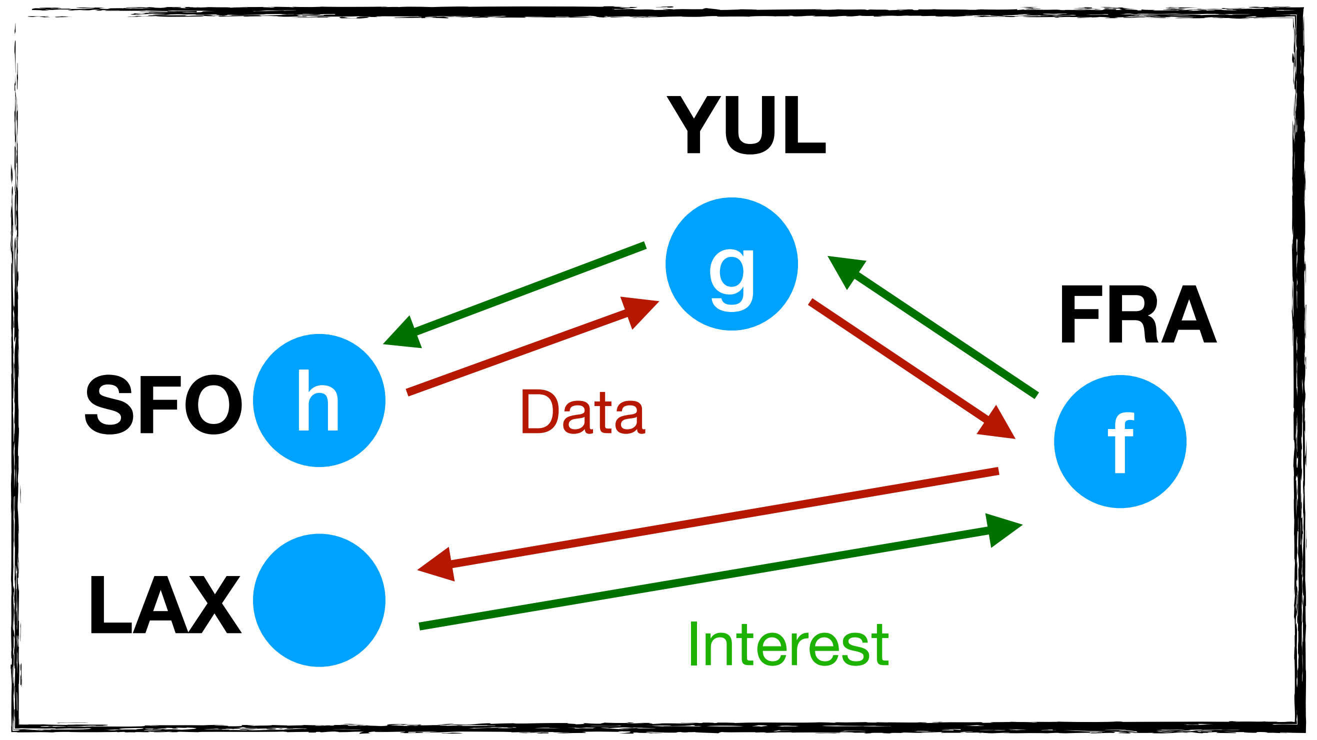


PUSH world

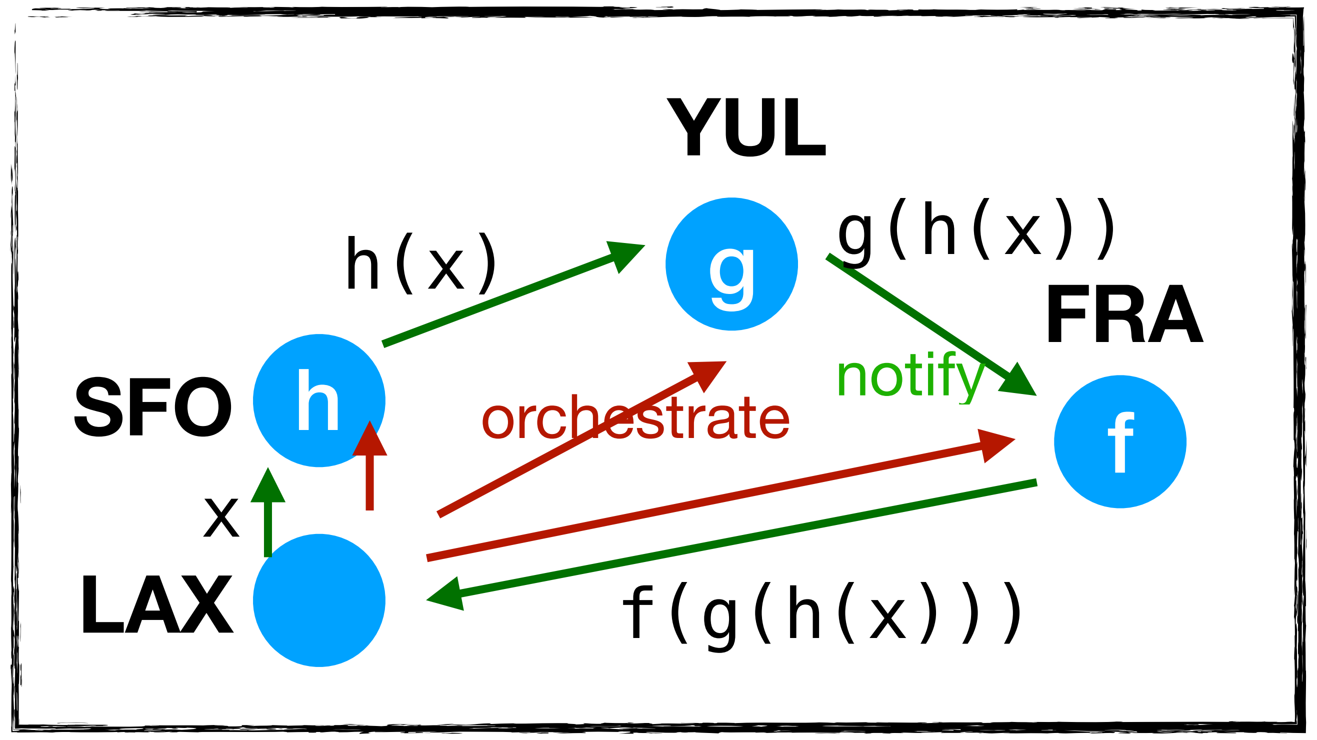
- Context: COIN (compute-in-the-net)
- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX
- PULL creates a "recursion corridor" (flashback: “mobile IP" and triangular routing…)
- In a PUSH world: only two Atlantic-crossings in the critical path (compared to four when using PULL)
- PUSH world: pipelining (no rate-limiting)

4b) Recursion corridors





PULL-world



PUSH world

- Context: COIN (compute-in-the-net)

- Consider chain of calls `f(g(h(x)))` executed at FRA, YUL and SFO, requested from LAX

- PULL creates a "recursion corridor" (flashback: “mobile IP" and triangular routing…)

- In a PUSH world: only two Atlantic-crossings in the critical path (compared to four when using PULL)

- PUSH world: pipelining (no rate-limiting)

- Corridors can be fixed in the PULL model (—> new emulation library, special name prediction tricks) but rate limitation will remain

5) Summary and Extrapolations

- I argue in favor of PUSH-of-append-only-logs
  doesn’t require infinite bandwidth: content frontier can be slowed down (backpressure)

5) Summary and Extrapolations

- I argue in favor of PUSH-of-append-only-logs
  doesn’t require infinite bandwidth: content frontier can be slowed down (backpressure)

- **Replicating append-only logs is about Shannon entropy:**
  - Heat entropy = “Verwandlungsgehalt” (transformational content)
  - Information entropy = “delta”
  Once new content is replicated (the world has reached the same temp)

5) Summary and Extrapolations

- I argue in favor of PUSH-of-append-only-logs
  doesn’t require infinite bandwidth: content frontier can be slowed down (backpressure)

- **Replicating append-only logs is about Shannon entropy:**
  - Heat entropy = “Verwandlungsgehalt” (transformational content)
  - Information entropy = “delta”
  Once new content is replicated (the world has reached the same temp)

- Unlike the NDN mantra that cache is an optional optimization:
  PUSH and storage go together: in-net storage is a MUST
  —> towards massive memory nets