A Broadcast-Only Communication Model

Christian Tschudin, University of Basel
March 24, 2019
ICNRG interim meeting, IETF Prague

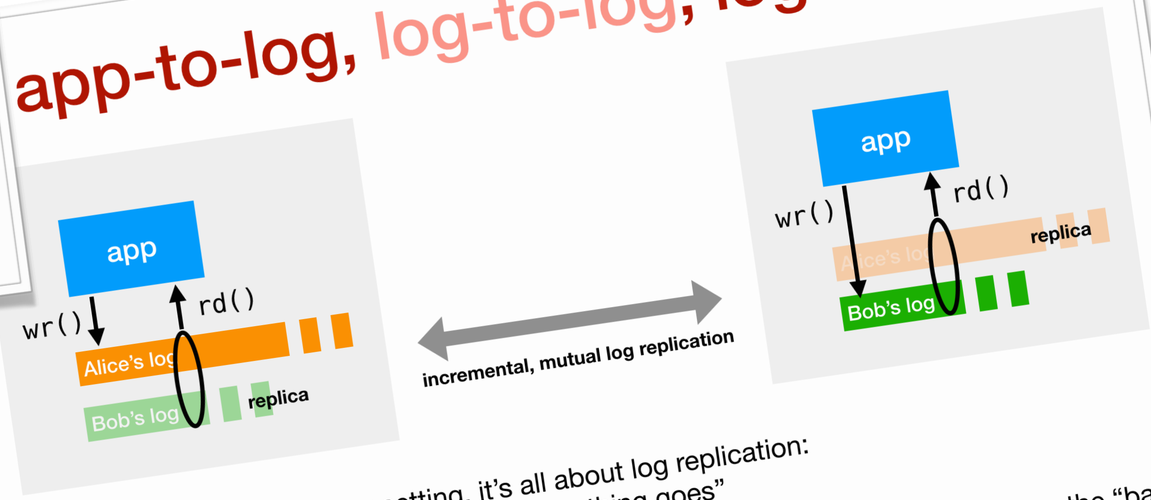
Continuity (two postcards from the past)

`pull()` vs `push()` is an ill posed problem: The "problem" disappears in another waist model

Christian Tschudin, University of Basel
Sep 23, 2018
panel slides

ACM ICN 2018 BOSTON

app-to-log, log-to-log, log-to-app



- In such a peer-to-peer setting, it's all about log replication: no `push()` or `pull()` question — "anything goes"
- The append-only logs make "set-difference" trivial — **much much** better than the "bag"

from the panel at ACM ICN 2018

Continuity (two postcards from the past)

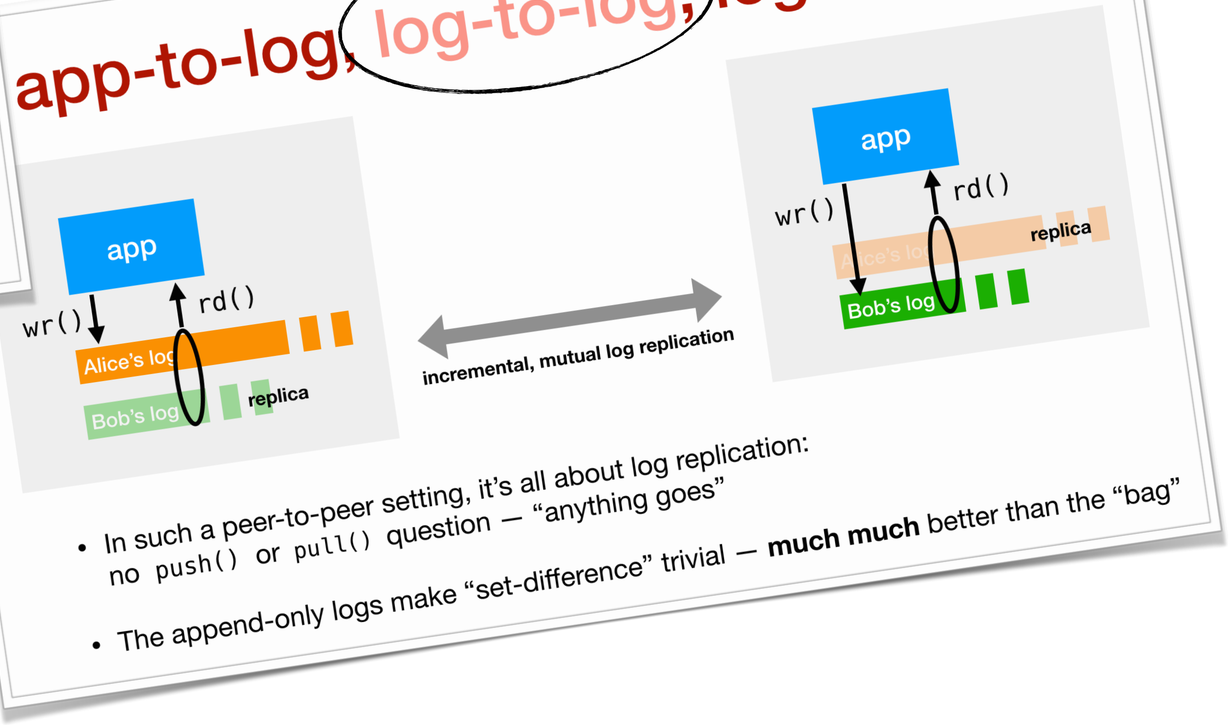
today's talk

`pull()` vs `push()` is an ill posed problem: The "problem" disappears in another waist model

Christian Tschudin, University of Basel
Sep 23, 2018
panel slides

ACM ICN 2018 BOSTON

app-to-log, log-to-log, log-to-app



- In such a peer-to-peer setting, it's all about log replication: no `push()` or `pull()` question — "anything goes"
- The append-only logs make "set-difference" trivial — **much much** better than the "bag"

from the panel at ACM ICN 2018

In a nutshell: Push is for Gods, Pull is for Mortals

- **“Global broadcast-only” induces “replicated append-only logs”**
  (the first time we see a comm model induce a data structure)

In a nutshell: Push is for Gods, Pull is for Mortals

- **“Global broadcast-only” induces “replicated append-only logs” (the first time we see a comm model induce a data structure)**

- Broadcast-only is reality, just hides in plain sight:
  Secure Scuttlebutt, PKI, Google Cloud Pub/Sub, inside Facebook

In a nutshell: Push is for Gods, Pull is for Mortals

- **“Global broadcast-only” induces “replicated append-only logs”**
  (the first time we see a comm model induce a data structure)

- Broadcast-only is reality, just hides in plain sight:
  Secure Scuttlebutt, PKI, Google Cloud Pub/Sub, inside Facebook

- Is *full* global broadcast-only possible? No, but:
  - model explains current ICN pain points well
  - guidance for “True ICN": say goodbye to the link+”arbigram” model

In a nutshell: Push is for Gods, Pull is for Mortals

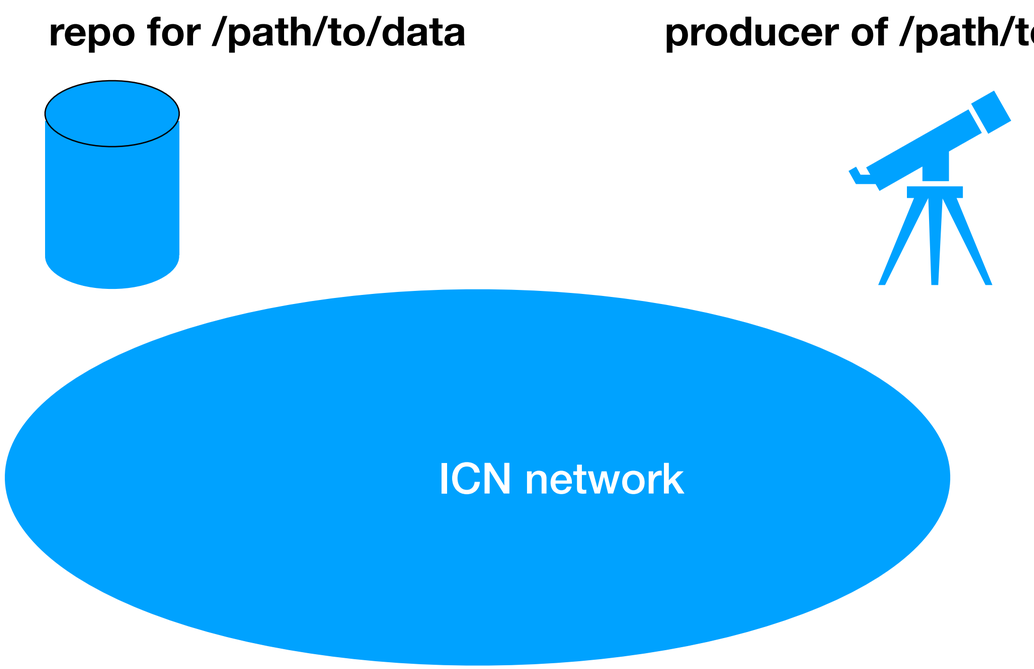
- **“Global broadcast-only” induces “replicated append-only logs”**
  (the first time we see a comm model induce a data structure)

- Broadcast-only is reality, just hides in plain sight:
  Secure Scuttlebutt, PKI, Google Cloud Pub/Sub, inside Facebook

- Is *full* global broadcast-only possible? No, but:
  - model explains current ICN pain points well
  - guidance for “True ICN": say goodbye to the link+”arbigram” model

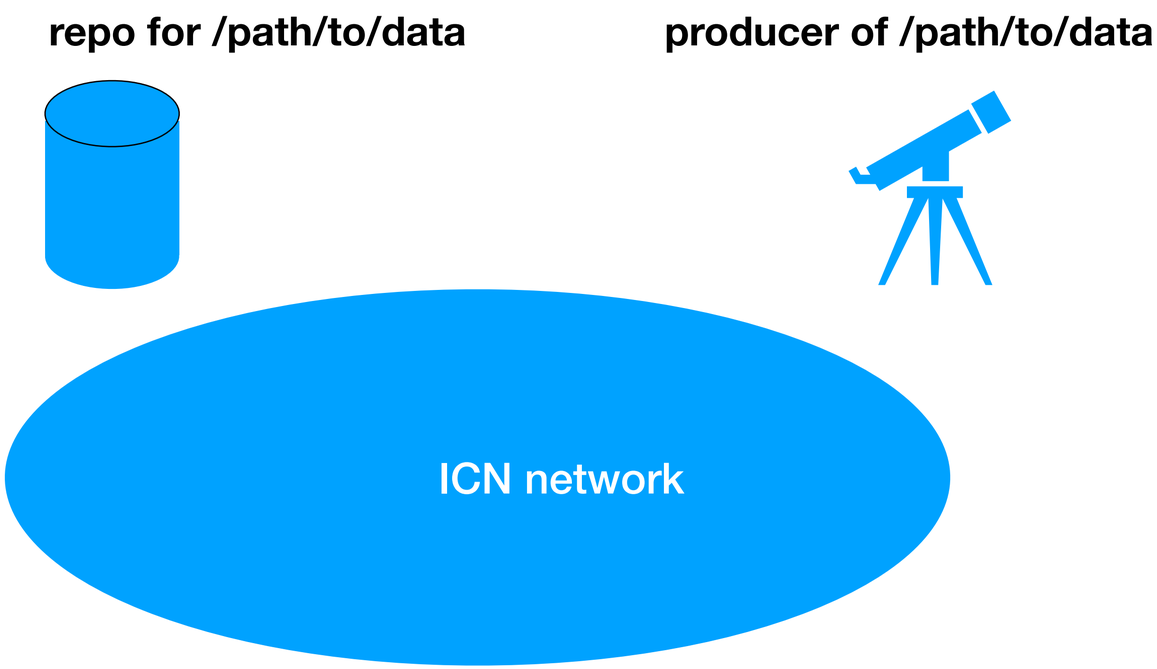
- Corollary for an ICN protocol stack waist:
  - embrace streams (solitary waves) instead of flow-balance
  - not worth fixing “faux ICN”, rather buy stocks in multicast companies

Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

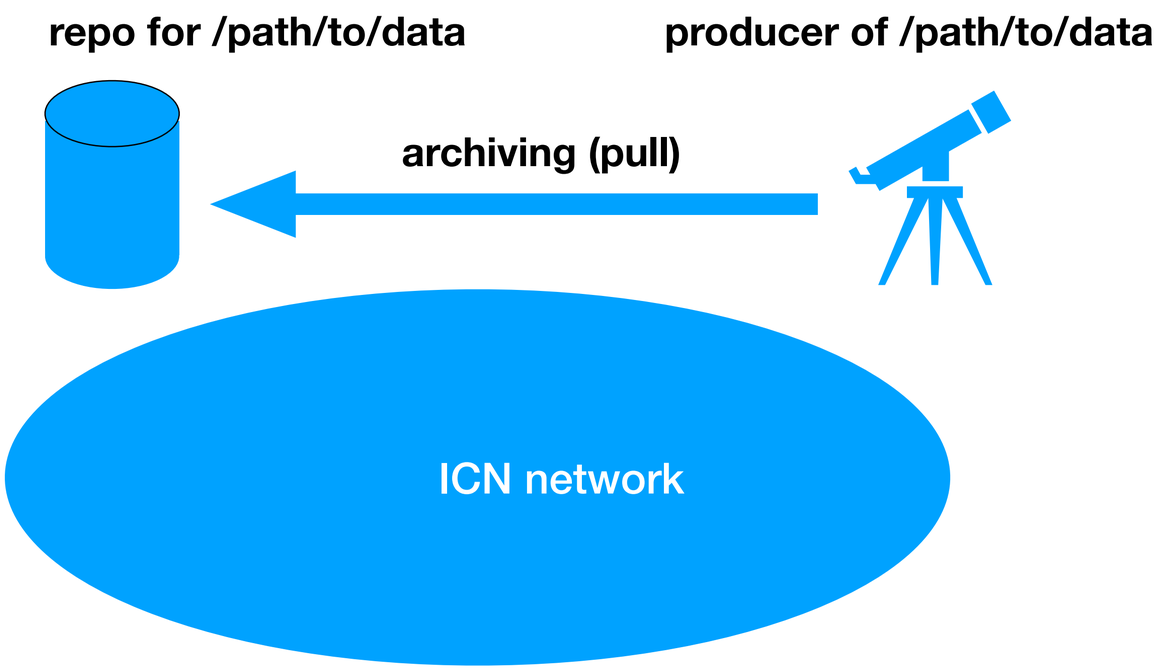
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name

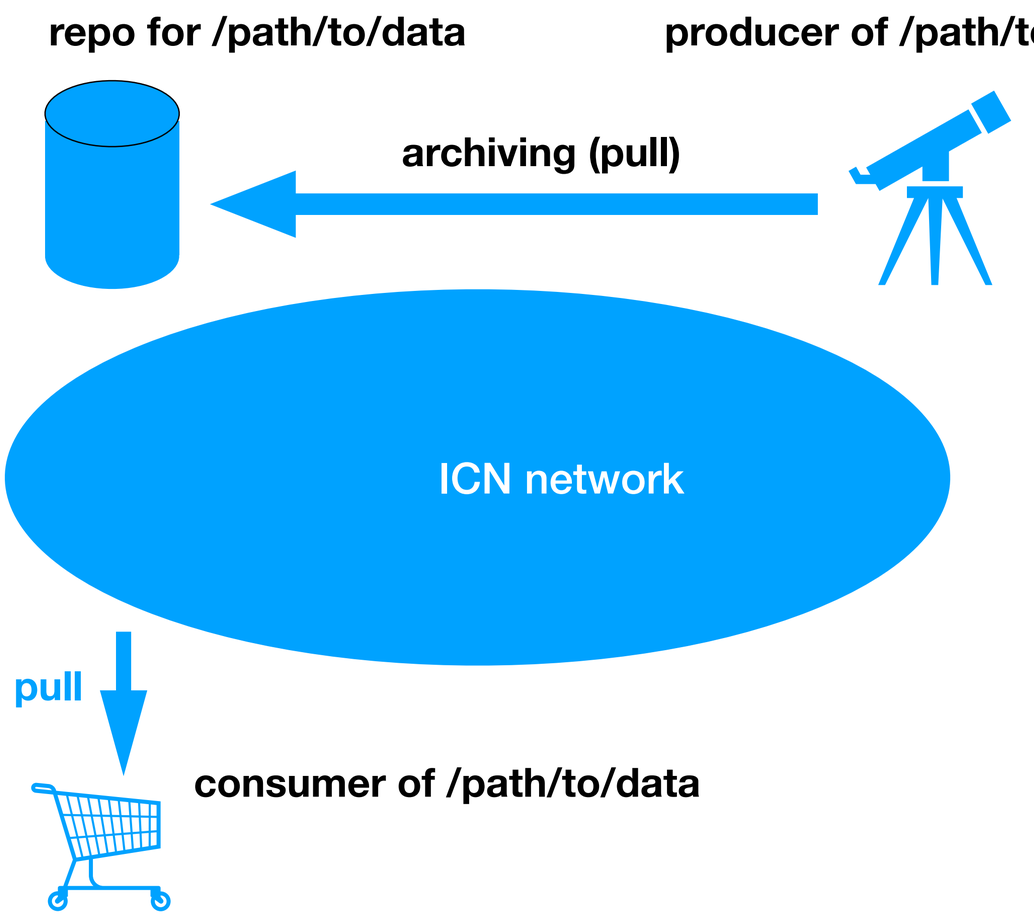
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name

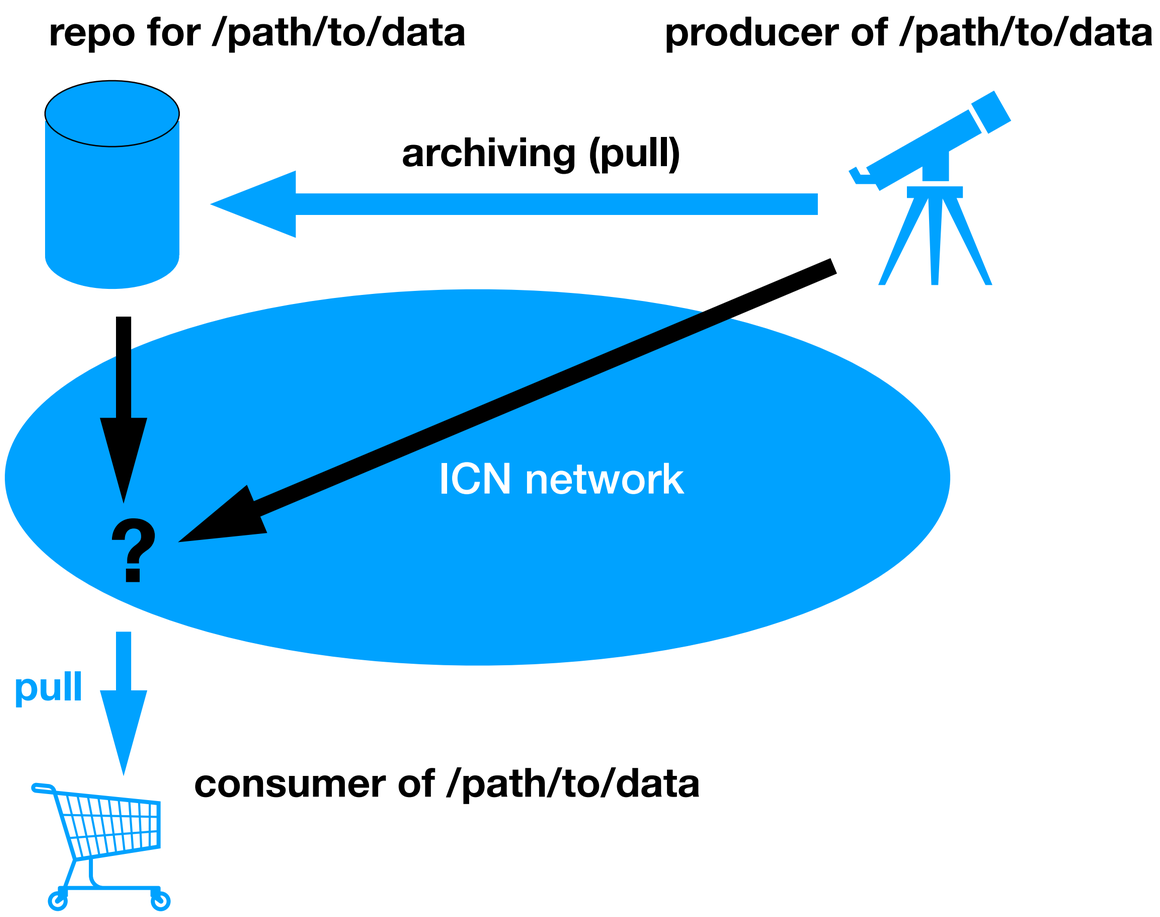
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name

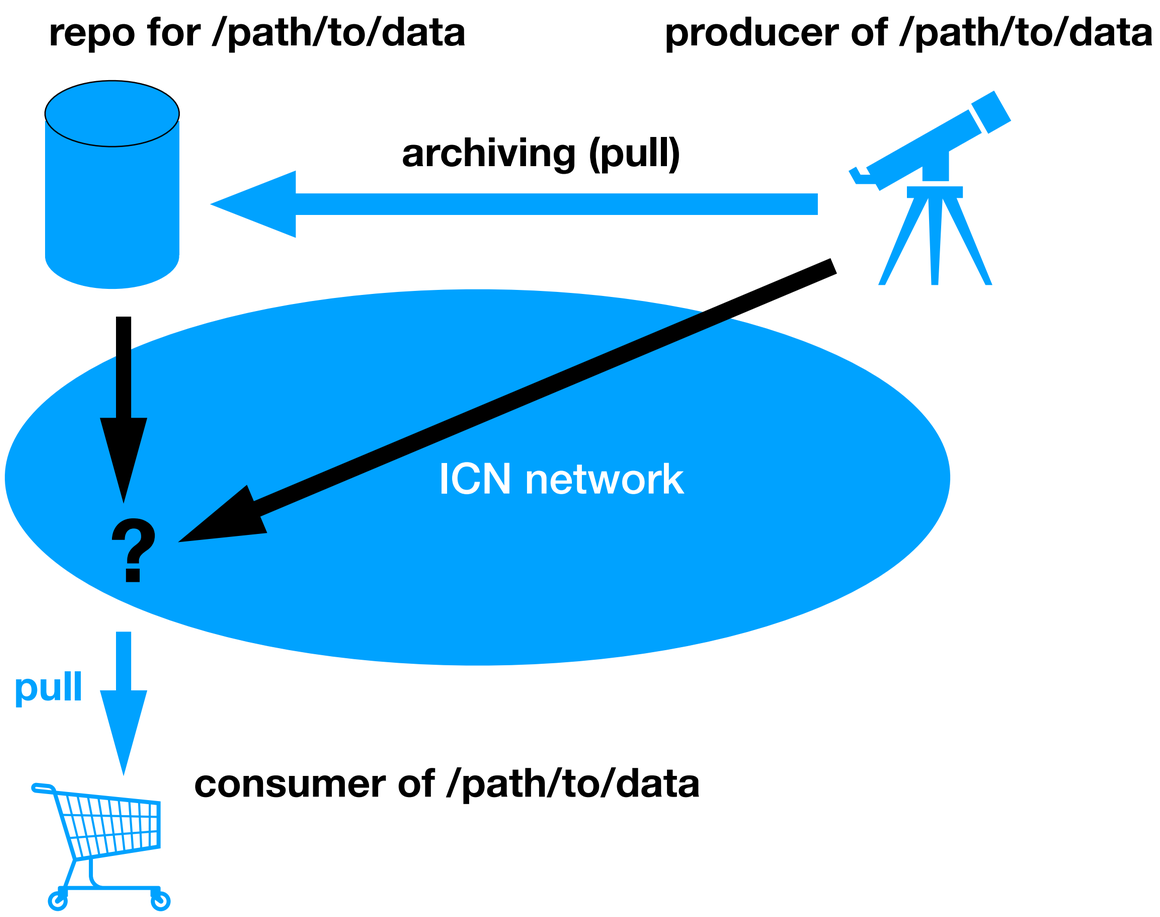
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name

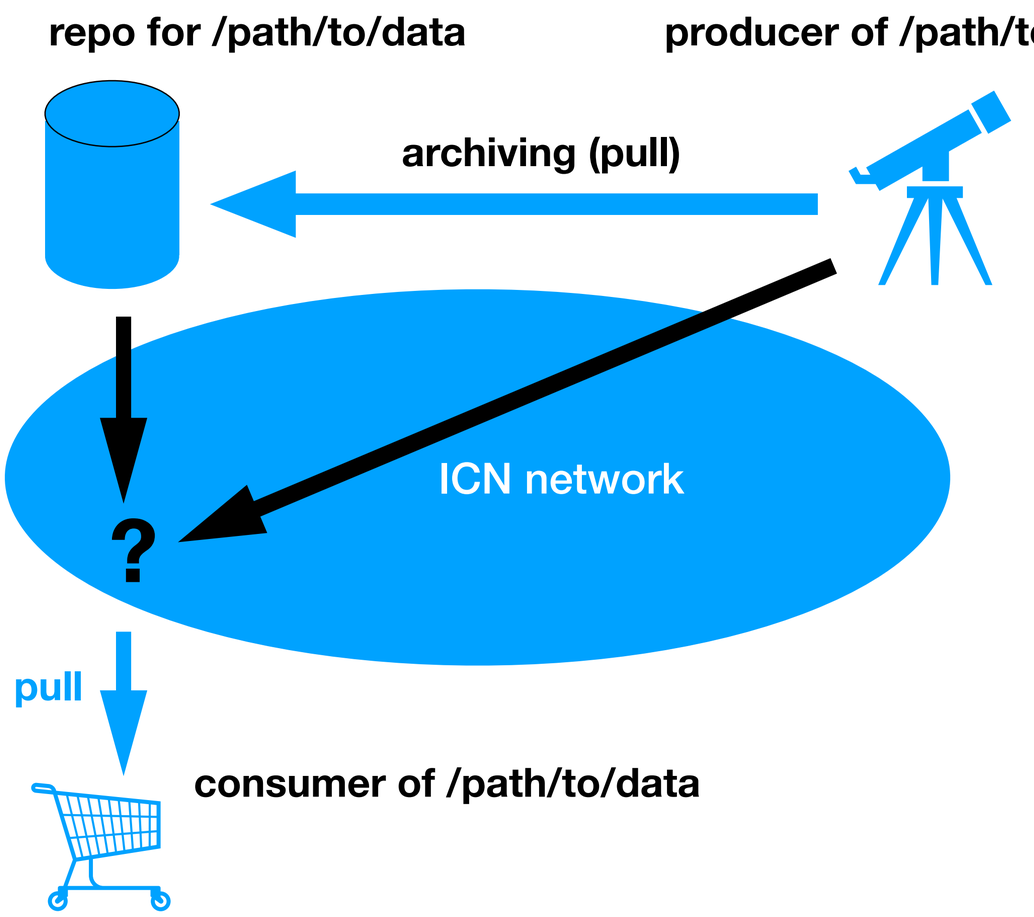
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name
- If repo delivery path has systematic lag, direct producer-consumer path is preferred

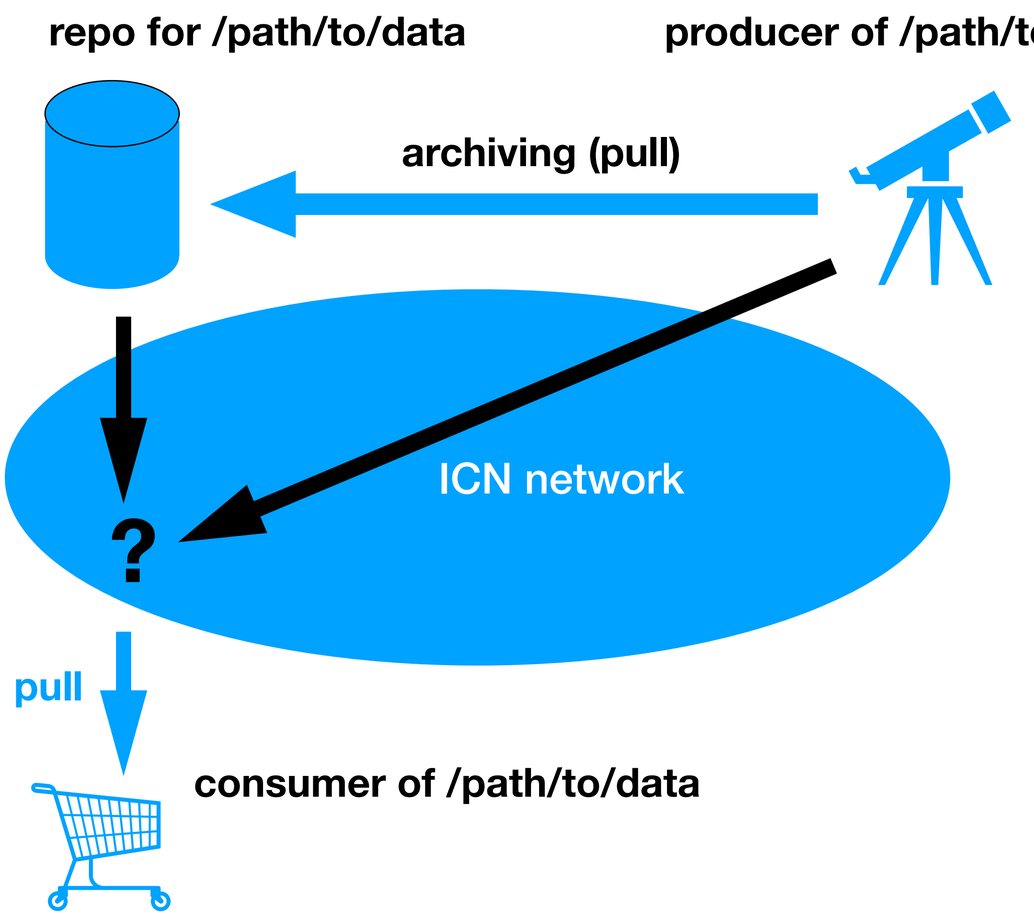
Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name
- If repo delivery path has systematic lag, direct producer-consumer path is preferred
- E.g. How can the network know when the producer stops (and only archived content is available)?

Recently, in an ICN project…



Config: streaming data is archived during recording, permits time shifting apps, also direct streaming.

- Mechanics: data producer AND repo register for the same name
- If repo delivery path has systematic lag, direct producer-consumer path is preferred
- E.g. How can the network know when the producer stops (and only archived content is available)?

Enters: Push. In a (repo AND prod) broadcast model, not an issue at all.

“With Push, not an issue, at all.”

In this talk: *PUSH = full global broadcast = full content replication*

“With Push, not an issue, at all.”

In this talk: *PUSH = full global broadcast = full content replication*

- no routing needed
- no mobile producer problem
- no requests needed
- no destination or requestor addresses
- no requestor state
- no timeouts, no polling for notifications

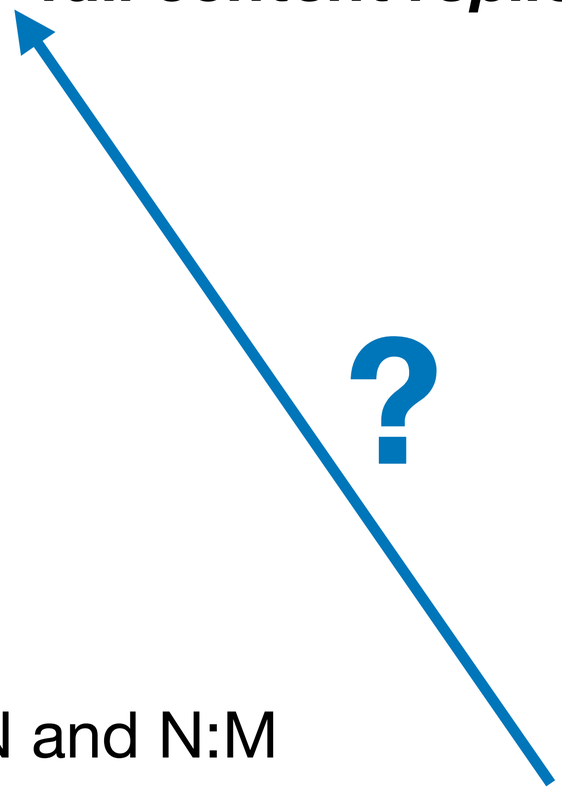
works bidirectionally and equally well for 1:N and N:M

“With Push, not an issue, at all.”

In this talk: *PUSH = full global broadcast = full content replication*

- no routing needed
- no mobile producer problem
- no requests needed
- no destination or requestor addresses
- no requestor state
- no timeouts, no polling for notifications

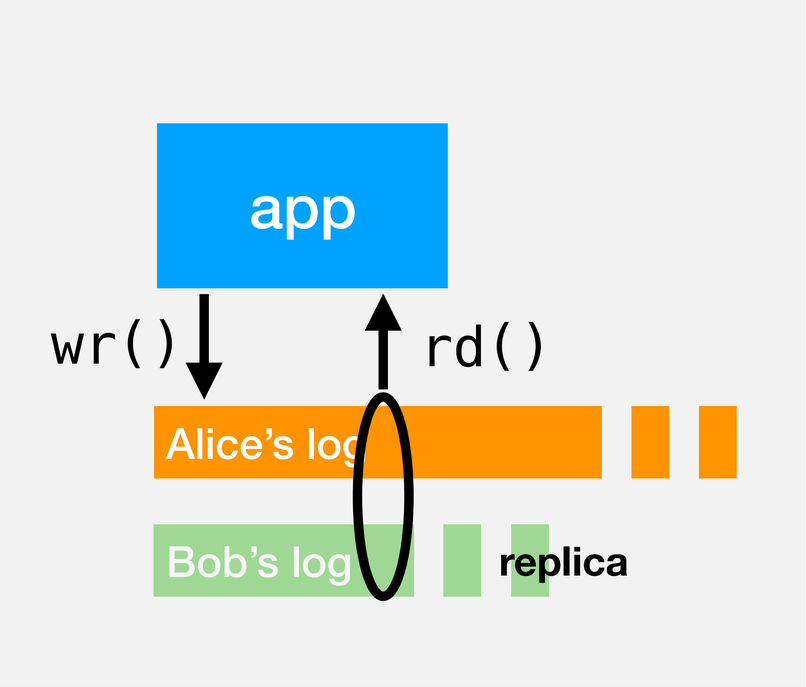
works bidirectionally and equally well for 1:N and N:M



What has “full global broadcast”
in common with “full content replication”?

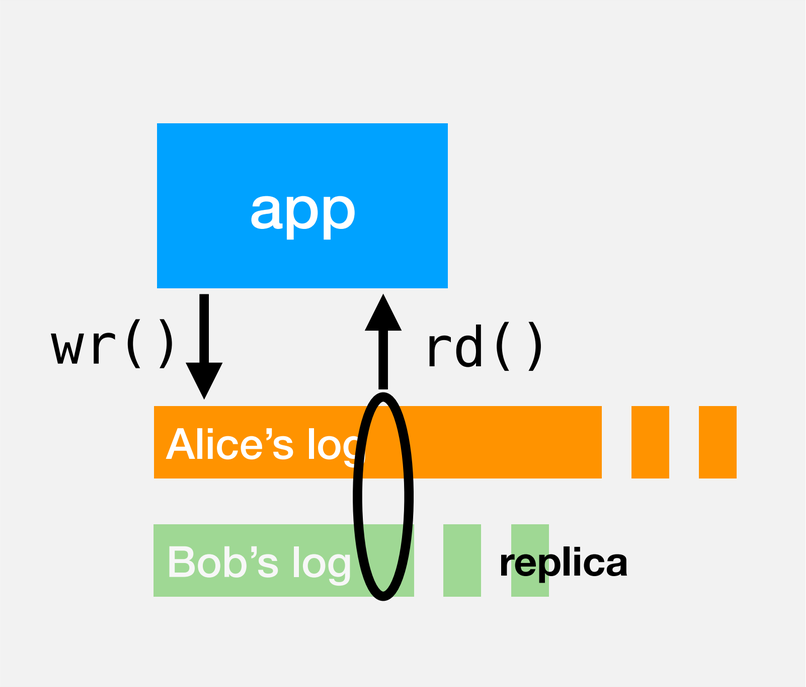
New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

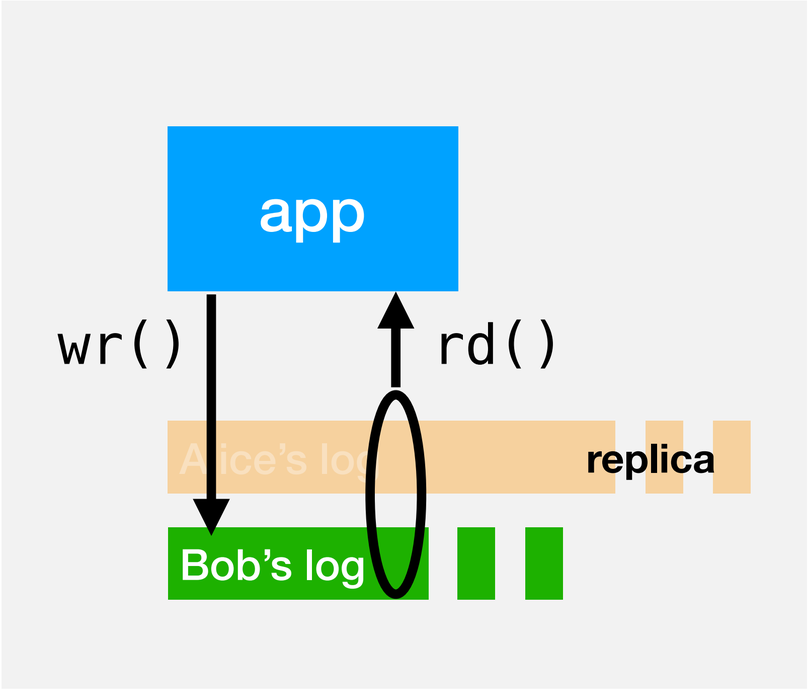
app-to-log, log-to-log, log-to-app



New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

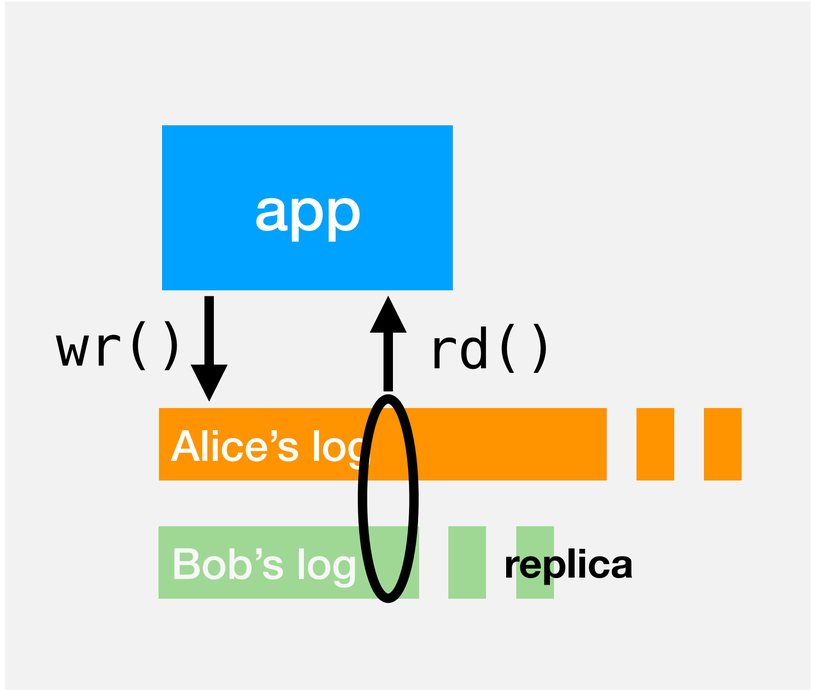
app-to-log, log-to-log, log-to-app



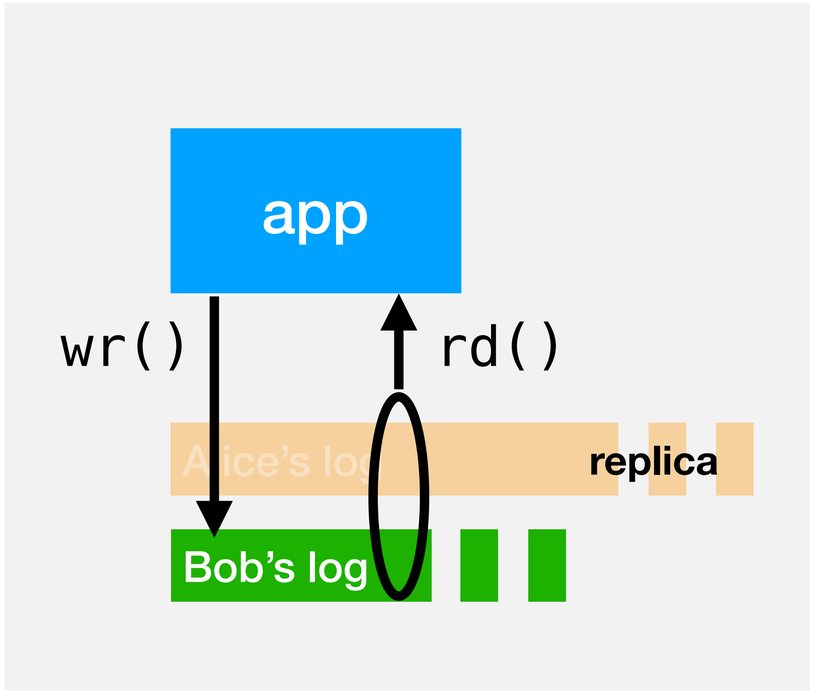


New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

app-to-log, log-to-log, log-to-app

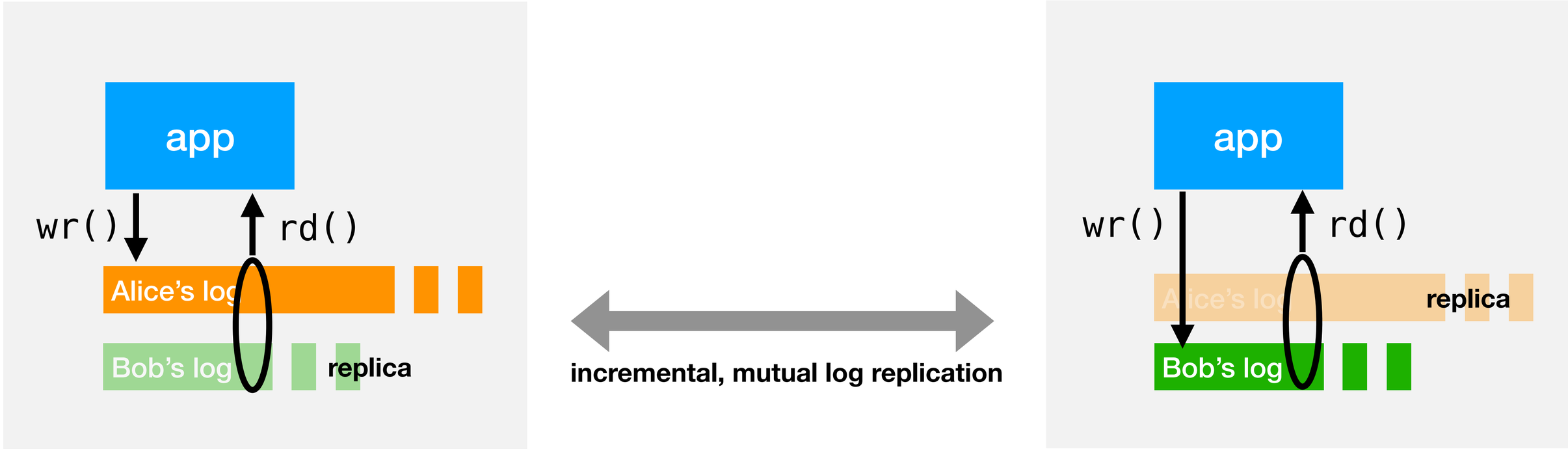


incremental, mutual log replication



New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

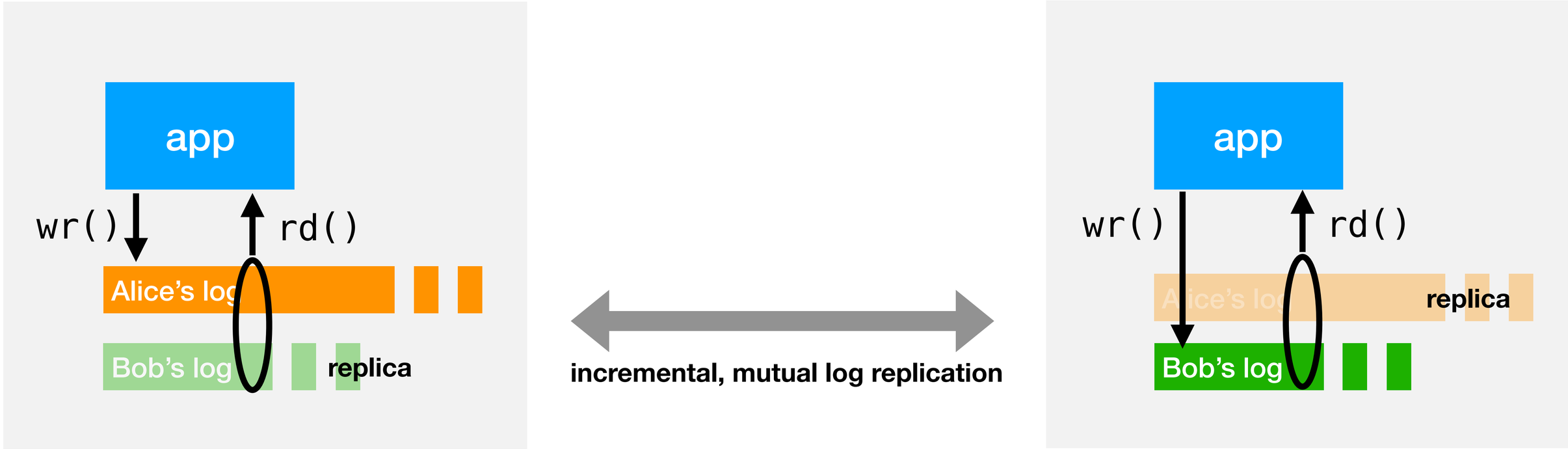
app-to-log, log-to-log, log-to-app



- In such a peer-to-peer setting, it's all about log replication:
  no `push()` or `pull()` question — "anything goes"

New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

app-to-log, log-to-log, log-to-app

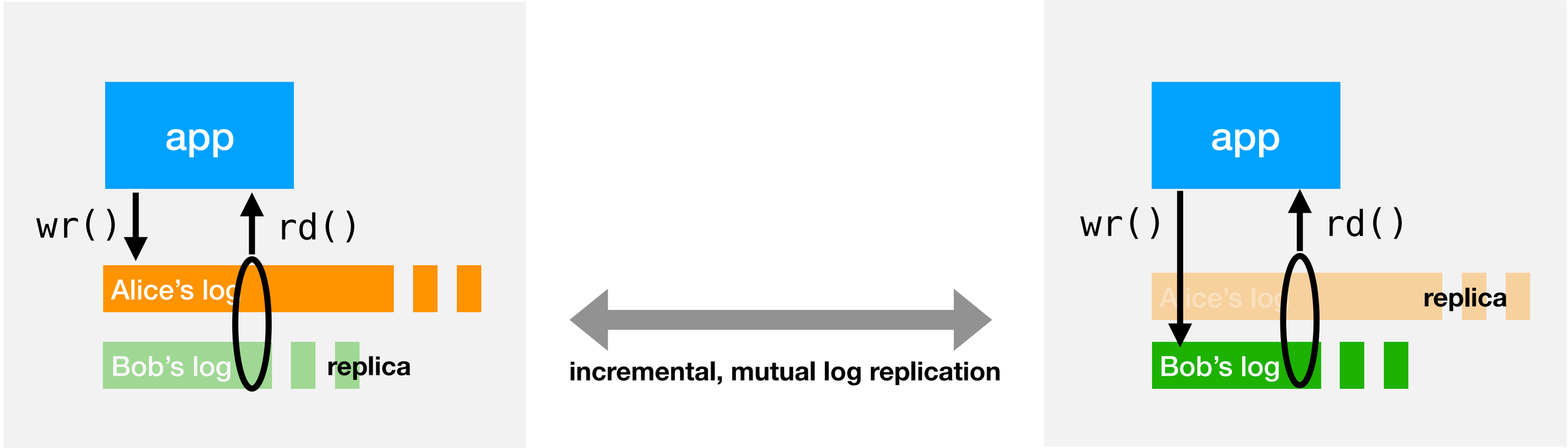


- In such a peer-to-peer setting, it's all about log replication:
  no `push()` or `pull()` question — "anything goes"

- The append-only logs make "set-difference" trivial — **much much** better than the "bag"

New style: decentralized apps! Two slides from September 2018 (ACM ICN'2018 panel):

app-to-log, log-to-log, log-to-app



- In such a peer-to-peer setting, it's all about log replication:
  no `push()` or `pull()` question — "anything goes"

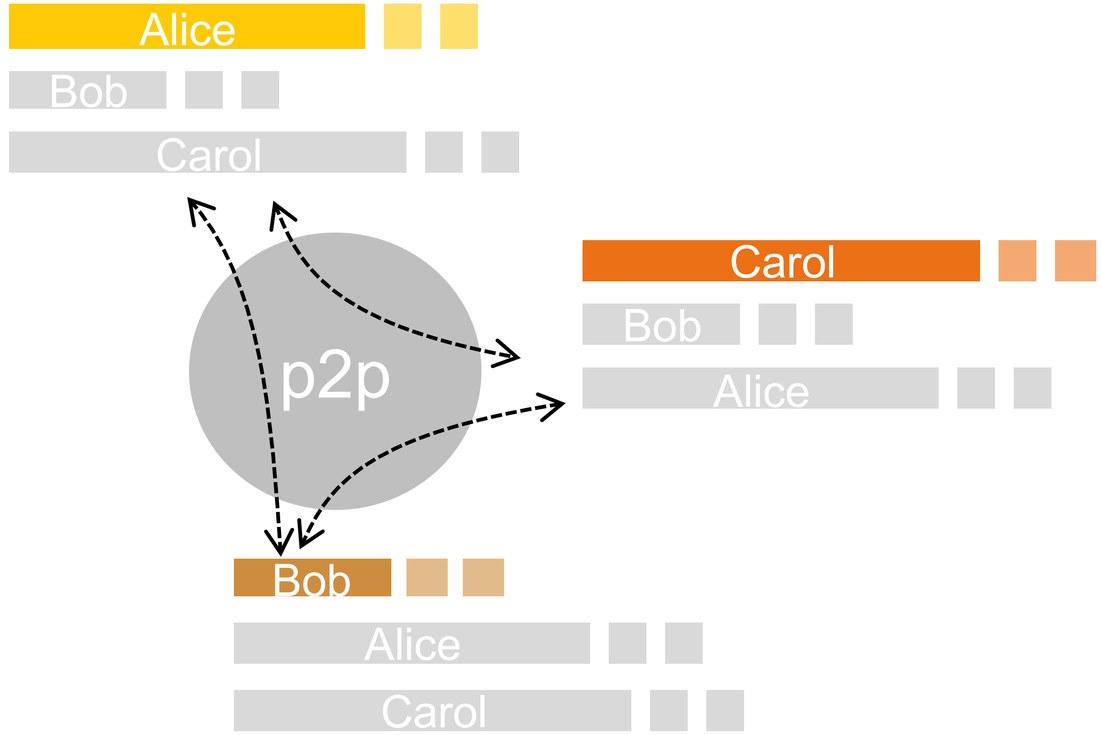
To me, 6 months later,
PUSH is the solution, PULL is the problem

- The append-only logs make "set-difference" trivial — **much much** better than the "bag"

Replicated Logs and “Subjective Readers"

Secure Scuttlebutt: Ground truth are the individual **append-only logs**

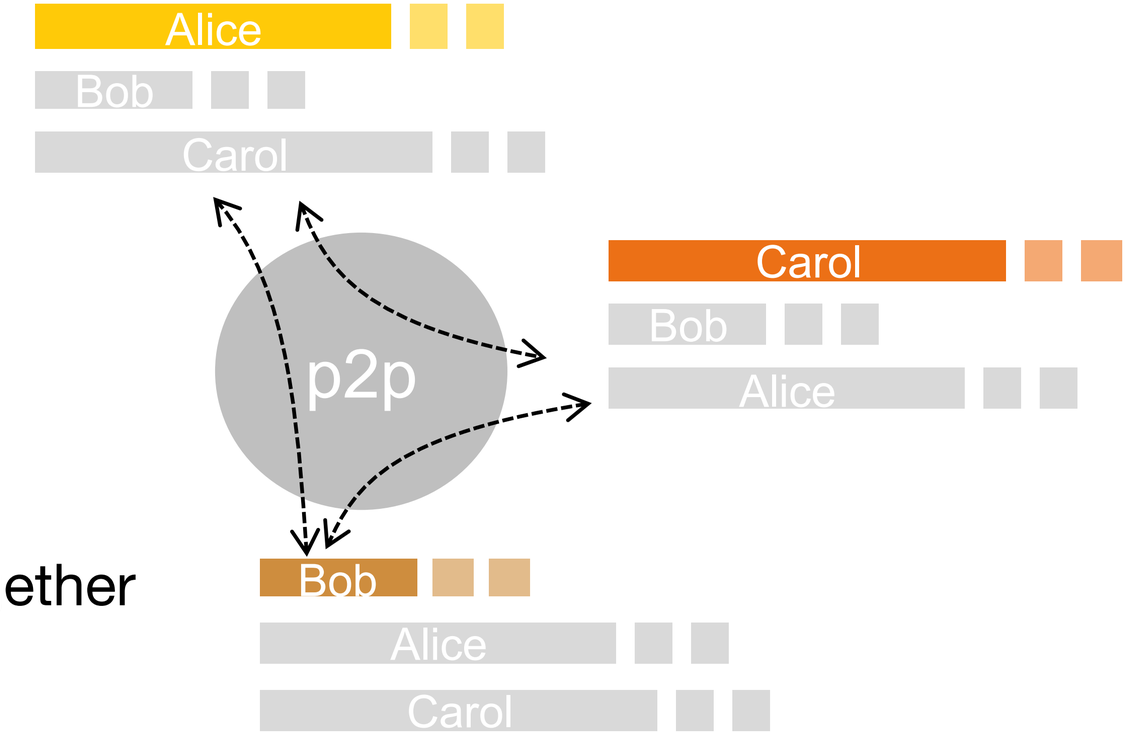
- hash-chained signed messages
- replication via peer-to-peer fabric



Replicated Logs and “Subjective Readers"

Secure Scuttlebutt: Ground truth are the individual **append-only logs**

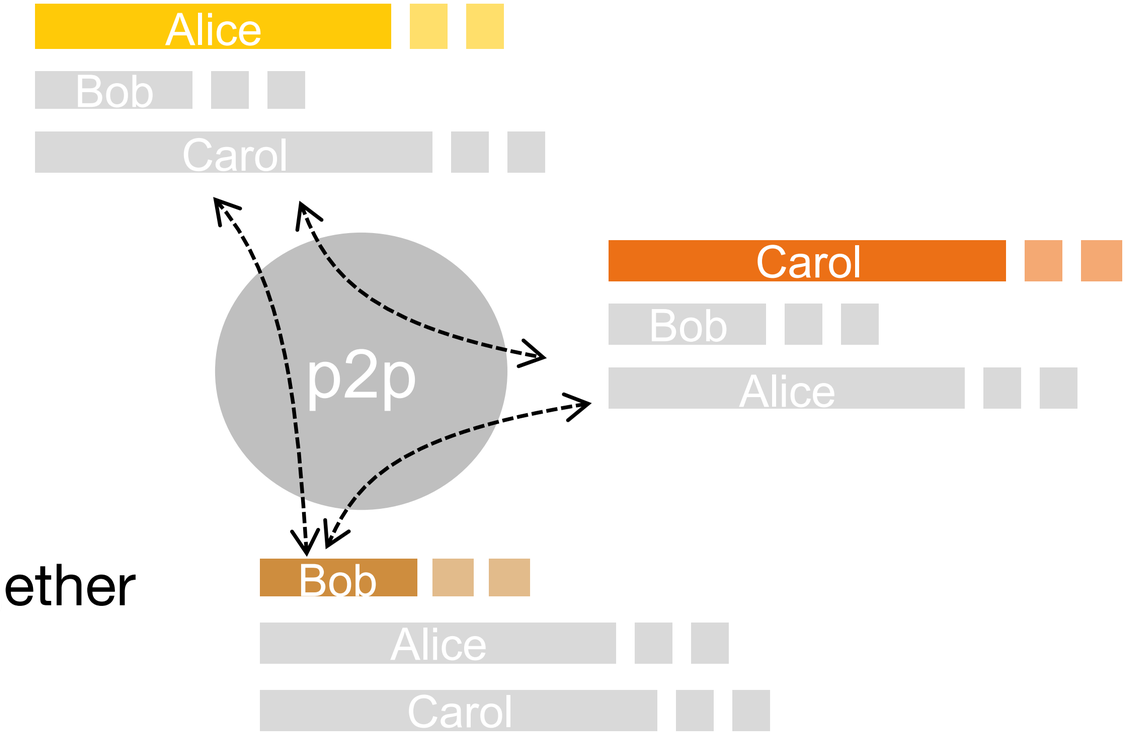
- hash-chained signed messages
- replication via peer-to-peer fabric
- “subjective reader”: ***locally*** reconstruct ADT (e.g. chat dialogue) from stitching together entries from each participant’s log



Replicated Logs and “Subjective Readers"

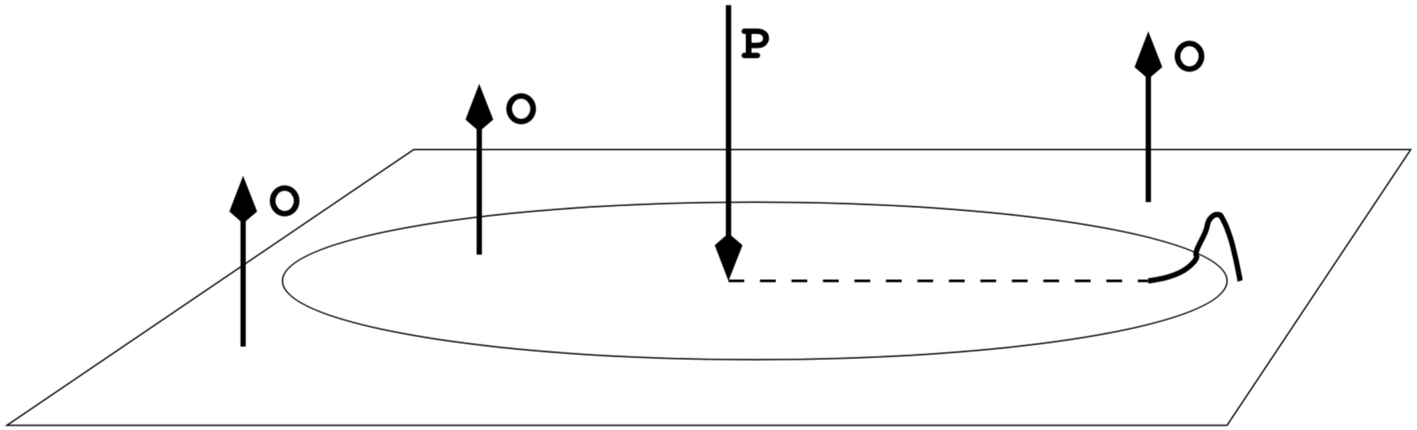
Secure Scuttlebutt: Ground truth are the individual **append-only logs**

- hash-chained signed messages
- replication via peer-to-peer fabric
- “subjective reader”: ***locally*** reconstruct
  ADT (e.g. chat dialogue) from stitching together entries from each participant’s log
- In SSB, distributed app = locally (!)
  - write to your own log
  - read from all relevant peers’ logs



Comm: from analog perturbation..

Solitary waves (solitons) as an ideal communication model for ICN

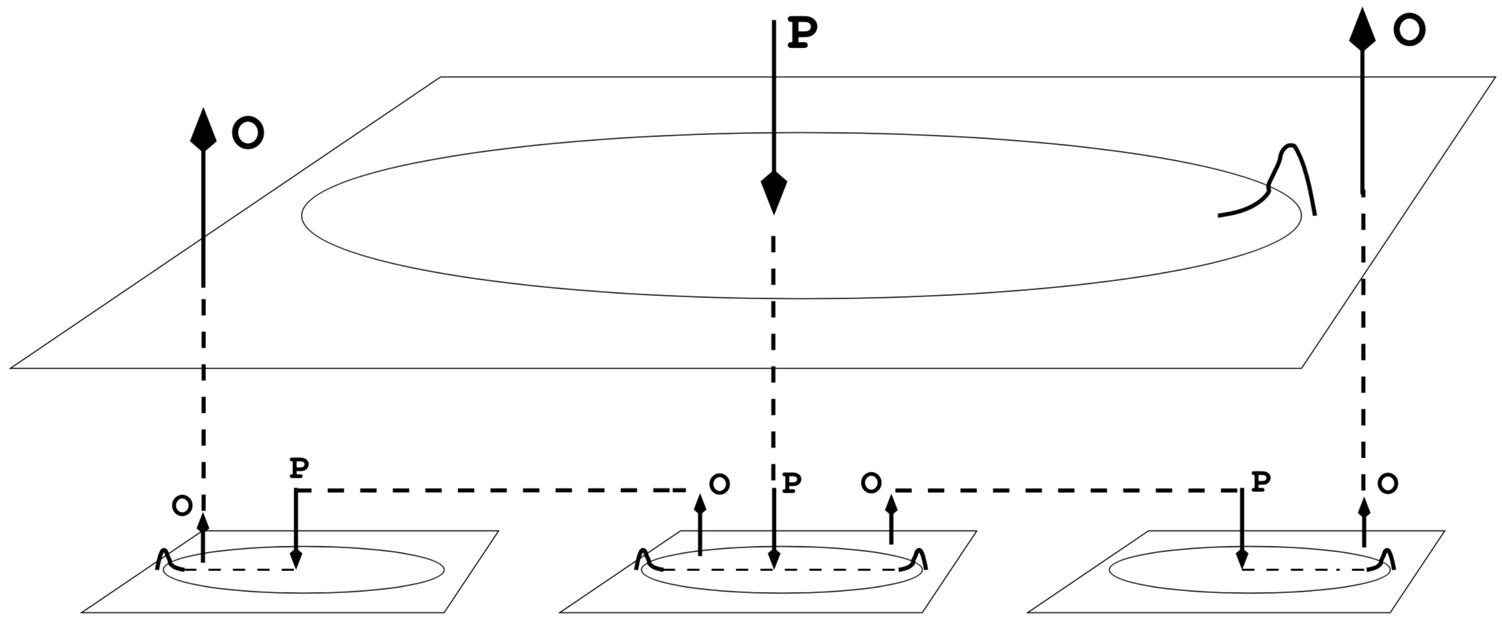


- **P**roducer initiates perturbation wave
- solitary wave: no trace after passage / infinite omnidirect. propagation
- passive **O**bservers, at arbitrary places

Service: reliable (exactly-once), ordered event delivery for all observers

.. to a global broadcast abstraction

Concatenate local broadcast domains, to form a global service



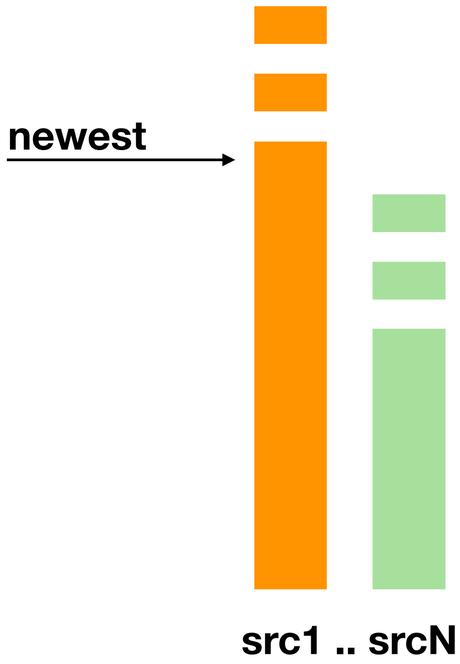
- place repeaters (observers that act as producers) where needed
- simply re-flood
- normal case: observers see the same perturbation multiple times

Approach: only forward first perturbance —> synching on the frontier

- requires a way to identify source and perturbation —> src id + event reference

Universal* Soliton Repeater

incoming
packet <src,ref,val>



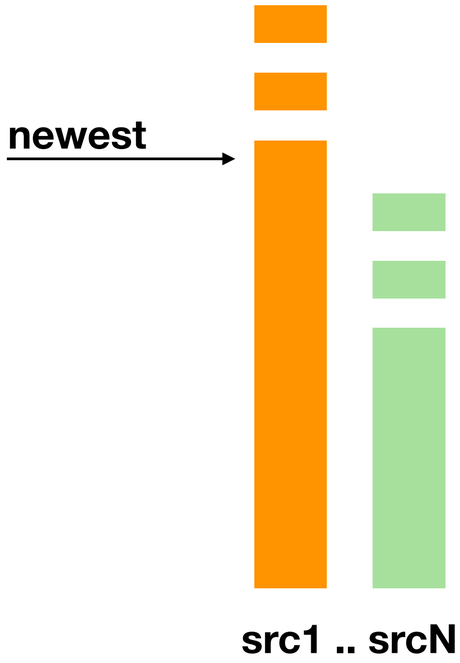
per source logs

```
1   Append_only_forwarding:
2     // arbitrary network topology, dynamics, delay, loss
3
4     log[] // complete perturbation history, per source
5
6     on_sense(P=<src,ref,val>):
7       if next_ref(log[P.src].newest) == P.ref:
8         broadcast(P)
9         log[P.src].append(P)
10        // observer upcall for P.val goes here
11
12    on_regular_intervals:
13      for all src:
14        broadcast(<src,ARQ,next_ref(log[src].newest)>)
15        // ARQ is a fixed non-reference value
16
17    on_sense(P=<src,ARQ,ref>):
18      if exists Q in log[P.src] with Q.ref == P.ref:
19        broadcast(Q)
```

*) under some symmetry assumption

Universal* Soliton Repeater

incoming
packet <src,ref,val>



per source logs

```
1   Append_only_forwarding:
2     // arbitrary network topology, dynamics, delay, loss
3
4     log[] // complete perturbation history, per source
5
6     on_sense(P=<src,ref,val>):
7       if next_ref(log[P.src].newest) == P.ref:
8         broadcast(P)
9         log[P.src].append(P)
10        // observer upcall for P.val goes here
11
12    on_regular_intervals:
13      for all src:
14        broadcast(<src,ARQ,next_ref(log[src].newest)>)
15        // ARQ is a fixed non-reference value
16
17    on_sense(P=<src,ARQ,ref>):
18      if exists Q in log[P.src] with Q.ref == P.ref:
19        broadcast(Q)
```

*) under some symmetry assumption

“comm model induces data struct"

Handling *all* nastiness of asynchronous communication forces log keeping:

- arbitrary multi-path - only propagate first perturbation
- arbitrary delays - duplicate detection requires full log
- arbitrary loss patterns - must be able to replay any log position

Good news: *we get really nice properties for building distributed apps*

- strict progress (information wavefront), efficiency:
  once replicated, content is never requested again
- `<src,ref>` (DONA!) plus `next_ref()` ideal for causal ordering of events
  (DAG and tangle data structures, log as a blockchain)

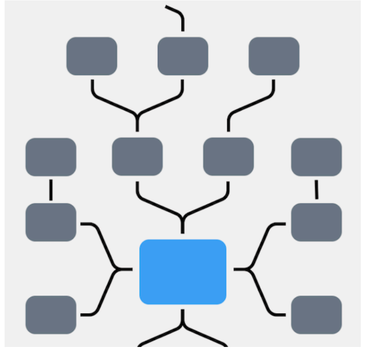
Global PUSH exists, today !

Pointing out three systems:

- Secure Scuttlebutt (see decent. app example before, ICNRG interim in Boston)
- PKI (next slide)
- Google Cloud Pub/Sub (next slide)

Yet another radar beep:

- Facebook uses log replication, must “sync” the frontier
  (do a ducksearch for *“homomorphic hashing”*)



Have a second look at TCP:

- TCP replicates two byte streams=logs (but only keeps a clipped log,
  has no crypto-assurance about segment names, and is unicast)

PKI… and log replication

- PKI = X.509 web certificates, certificate forest, roots at certificate authorities - the trust backbone of the Web (TLS)
- Some CAs caught in foul play, or were hacked: cert mis-issuance
  —> “Certificate Transparency” (CT), RCF 6962, June 2013

- CT implemented by collecting *all* certs, see e.g. https://crt.sh/, and https://www.certificate-transparency.org/

PKI… and log replication

- PKI = X.509 web certificates, certificate forest, roots at certificate authorities - the trust backbone of the Web (TLS)
- Some CAs caught in foul play, or were hacked: cert mis-issuance
  —> “Certificate Transparency” (CT), RCF 6962, June 2013

- CT implemented by collecting *all* certs, see e.g. https://crt.sh/, and https://www.certificate-transparency.org/

- Instead of central database:
  - **fully replicated append-only log**
  - gossip-style replication (possible because of monotonic growth)
  - computing trust out of the log, in a trustless way … like SSB, only single app

PKI… and log replication

- PKI = X.509 web certificates, certificate forest, roots at certificate authorities - the trust backbone of the Web (TLS)
- Some CAs caught in foul play, or were hacked: cert mis-issuance
  —> “Certificate Transparency” (CT), RCF 6962, June 2013

- CT implemented by collecting *all* certs, see e.g. https://crt.sh/, and https://www.certificate-transparency.org/

- Instead of central database:
  - **fully replicated append-only log**
  - gossip-style replication (possible because of monotonic growth)
  - computing trust out of the log, in a trustless way … like SSB, only single app

- Gasser et al: *In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements*, PAM 2018 conference

Google Cloud Pub/Sub

Pub/Sub = event notification bus to coordinate distributed apps

- Google's global service: *"a secure, durable, highly available and scalable many-to-many messaging system"*

- durable means: **delivery guarantee** even if all Google servers crash at the same time, hence Google must store an event until all consumers fetched it.

- Crash-resistant storage solutions (e.g. RAFT protocol, WAL) … use logs

—> ICN services today **are** relying on logs, but not exposing them to the app layer

Assessment

Pull-based ICN (e.g. NDN) an awkward "slicing" through the trade-off space:

- notification not available (“long-lasting interest” hack -> invitation to push)
- *sending* data only possible via interest-abuse: stuff data into interest,
  or put cmd in interest to let “repo” call you back
- it’s called “receiver-driven”, but repo cannot protect against interest flood
  in other words: prefix registration is another “long-lasting interest” hack

Assessment

Pull-based ICN (e.g. NDN) an awkward "slicing" through the trade-off space:

- notification not available (“long-lasting interest” hack -> invitation to push)
- *sending* data only possible via interest-abuse: stuff data into interest,
  or put cmd in interest to let “repo” call you back
- it’s called “receiver-driven”, but repo cannot protect against interest flood
  in other words: prefix registration is another “long-lasting interest” hack

But can we do the tradeoffs in a different way, have real global brcast?

Chances of getting Broadcast-only

- LAN looks good: (re-) flooding **is** feasible, first steps have been explored e.g. McCauley et al: *The Deforestation of L2,* SIGCOMM 2016

Observation: network has **one** knob - **rate**. Delay a producer at will, without breaking contract.

Long distance brings bottlenecks, needs content selection -> set some producers to rate 0. How to select? —> via subscription, this requires a reverse channel:

- instead of *repeated* interests:
  receiver also acts as producer, puts “I need replica of producer X” in its log *once*, is replicated via broadcast (and consulted by the network), request is valid until revoked.

- More areas to explore: multicast, …

Summary: Push is for Gods, Pull is for Mortals

- “Global broadcast-only” induces “replicated append-only logs”

Summary: Push is for Gods, Pull is for Mortals

- “Global broadcast-only” induces “replicated append-only logs”
- Say NO to “arbigrams”. Better use replicate logs, DONA names `<src,ref>`

Summary: Push is for Gods, Pull is for Mortals

- “Global broadcast-only” induces “replicated append-only logs”
- Say NO to “arbigrams”. Better use replicate logs, DONA names `<src,ref>`
- Broadcast-only is reality
  Secure Scuttlebutt, PKI, Google Pub/Sub, inside Facebook

Summary: Push is for Gods, Pull is for Mortals

- “Global broadcast-only” induces “replicated append-only logs”
- Say NO to “arbigrams”. Better use replicate logs, DONA names `<src,ref>`
- Broadcast-only is reality
  Secure Scuttlebutt, PKI, Google Pub/Sub, inside Facebook
- Embrace streams (soliton waves) and multicast, instead of flow-balance

Summary: Push is for Gods, Pull is for Mortals

- “Global broadcast-only” induces “replicated append-only logs”
- Say NO to “arbigrams”. Better use replicate logs, DONA names `<src,ref>`
- Broadcast-only is reality
  Secure Scuttlebutt, PKI, Google Pub/Sub, inside Facebook
- Embrace streams (soliton waves) and multicast, instead of flow-balance
- Look into “batch-oriented” high-perf-networks, beyond TCP
  (lightpath switched networks, “truck full of SSD drives”)