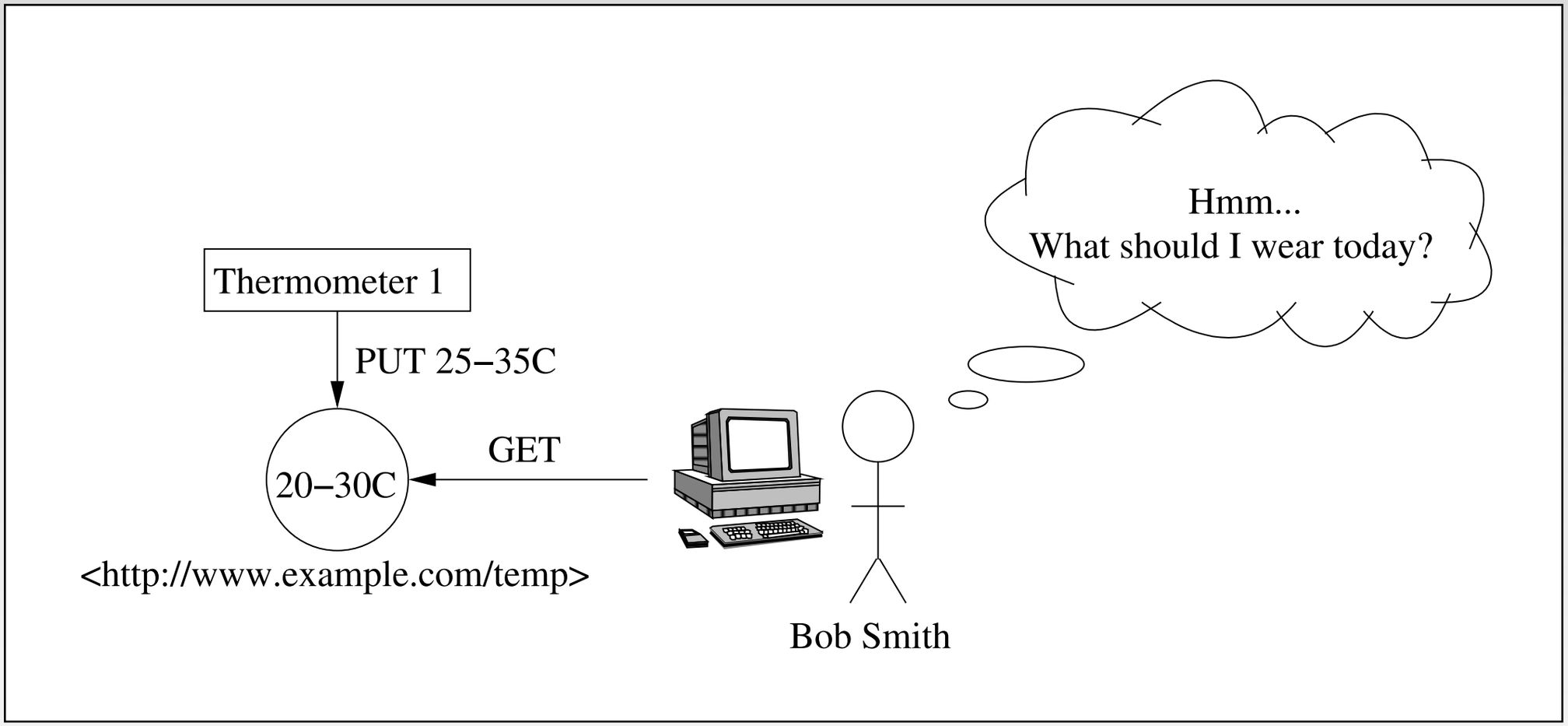
Distributed Data Propagator Networks

A “RESTful” Messaging System for Asynchronous Distributed Processing

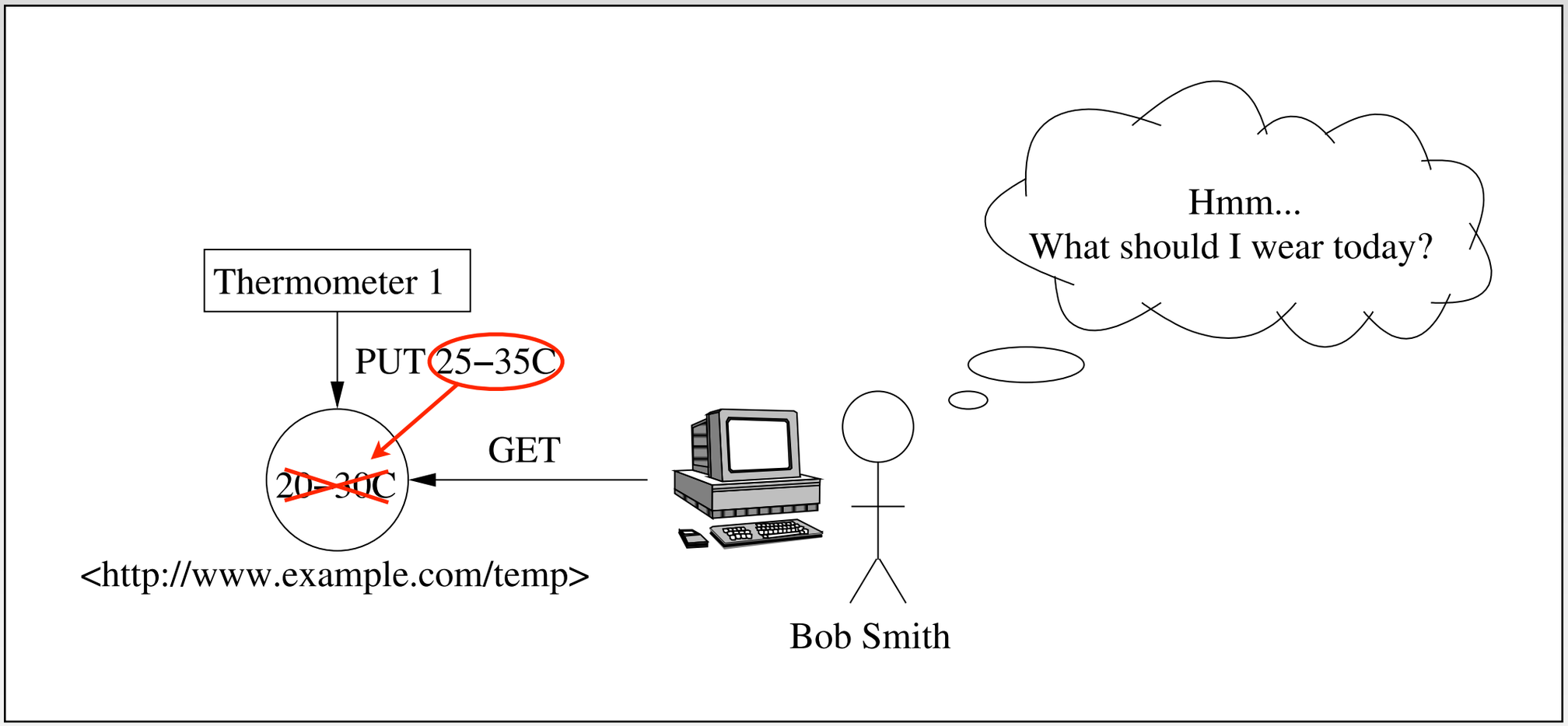
Ian Jacobi
MIT/CSAIL

Alexey Radul
MIT/CSAIL

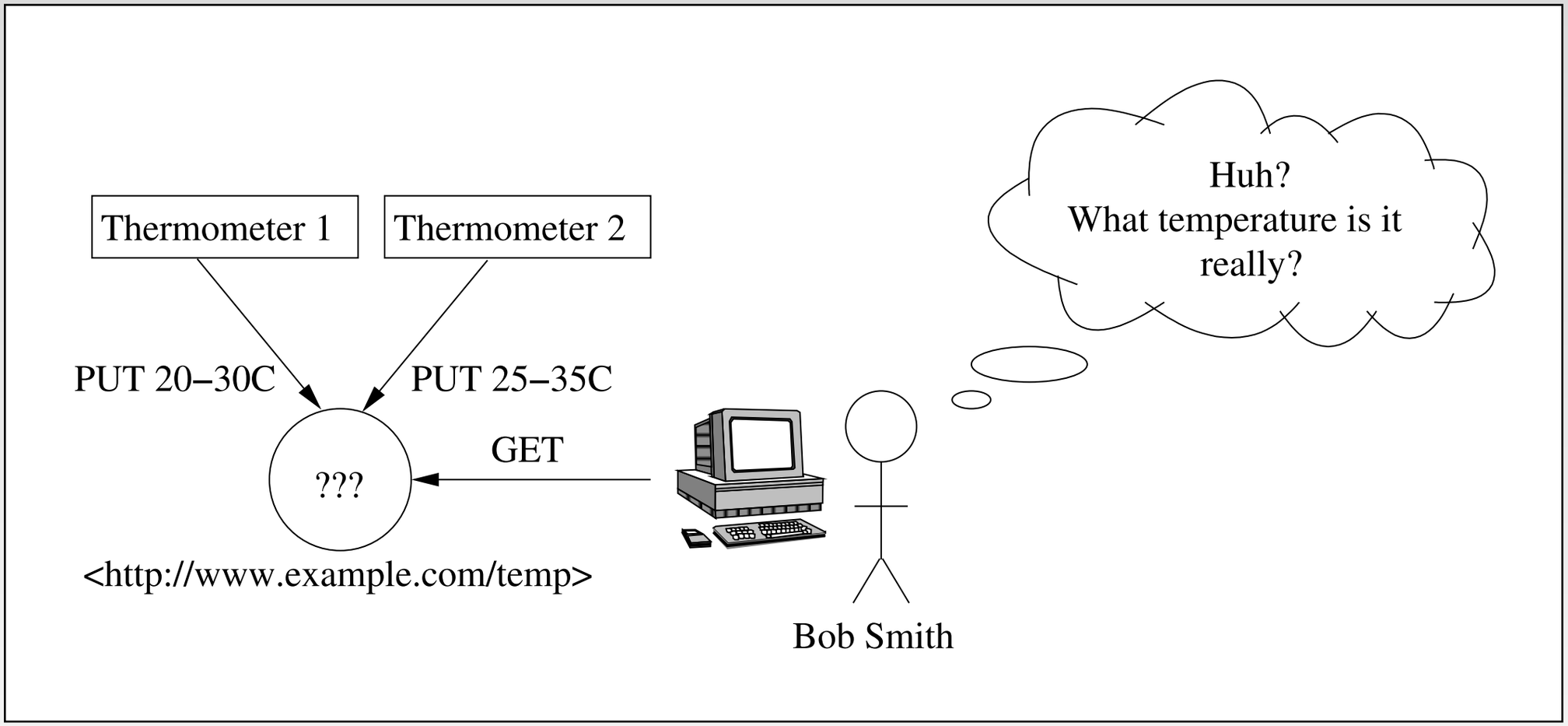
The Traditional RESTful Model



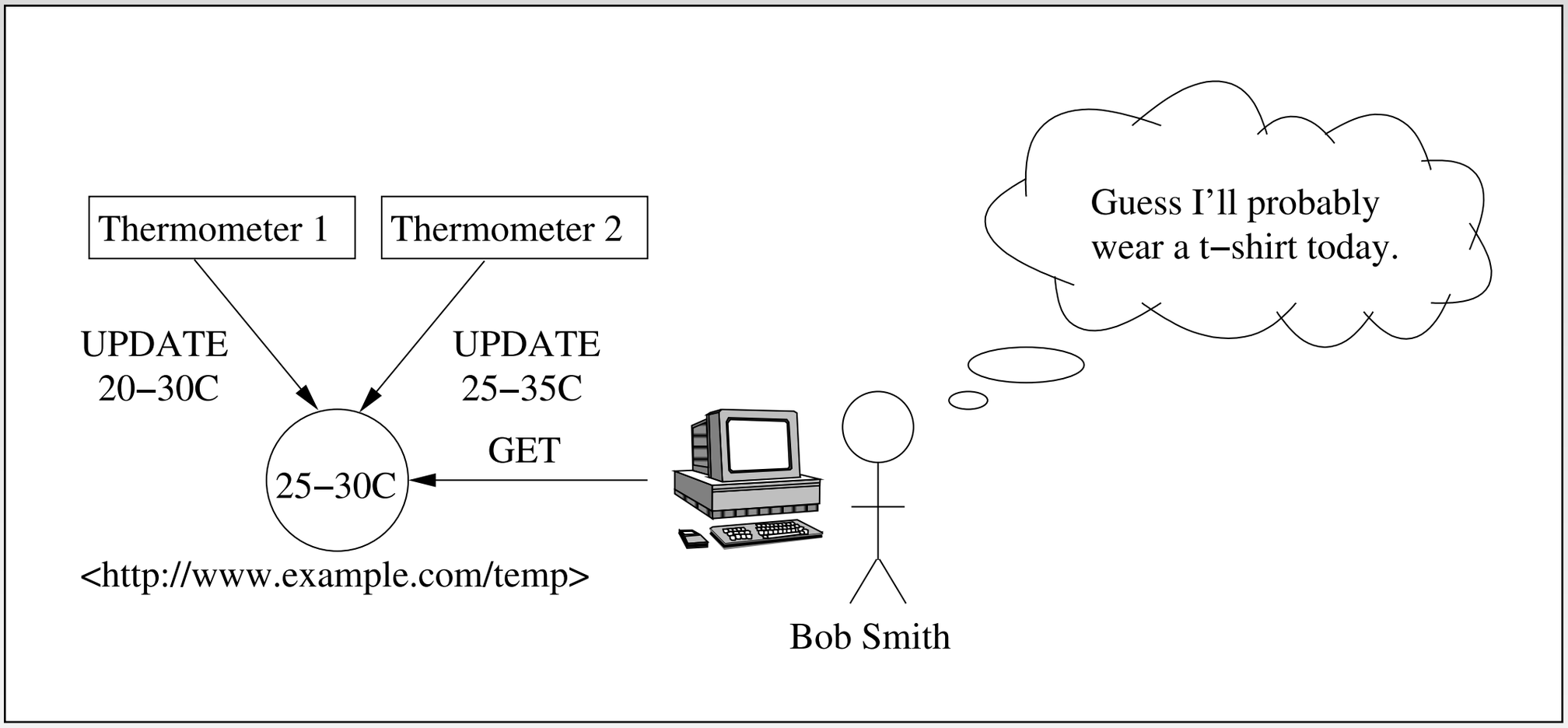
The Traditional RESTful Model



The Trouble With Multiple Inputs



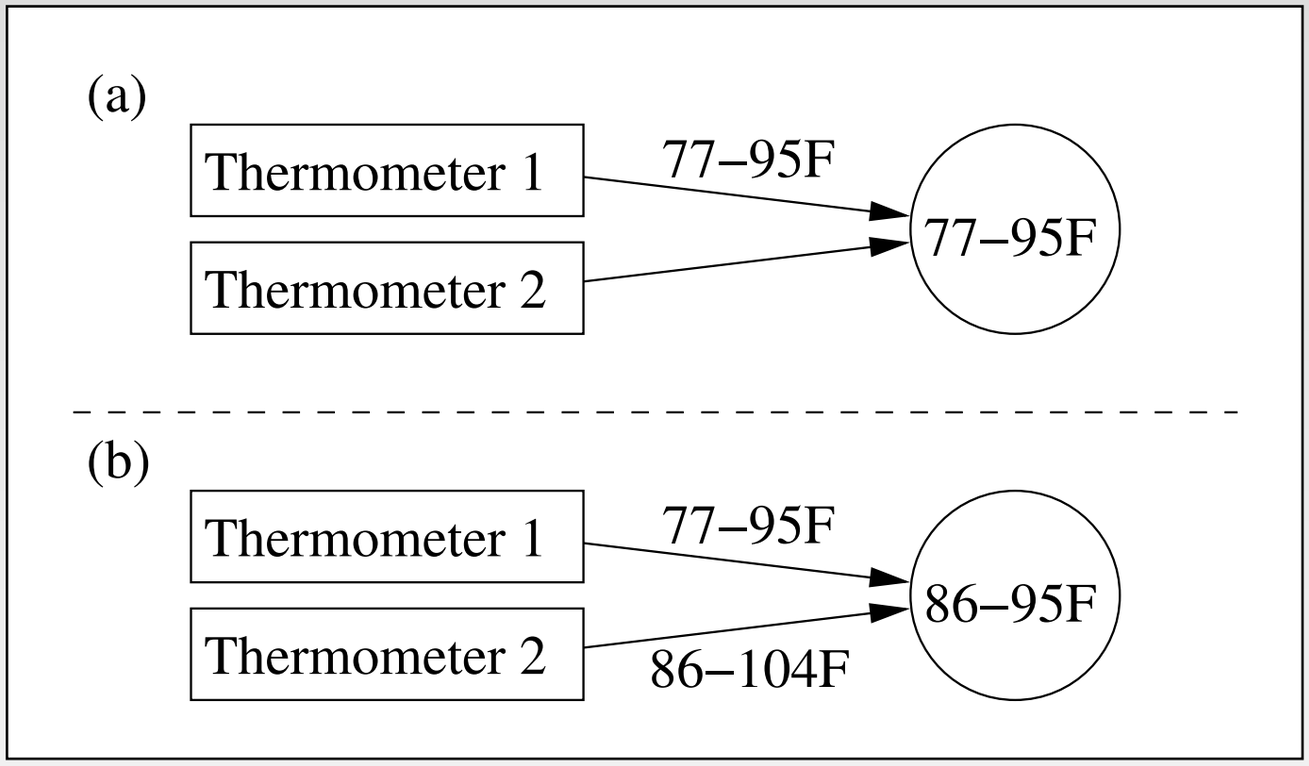
The Solution?
Data Propagation



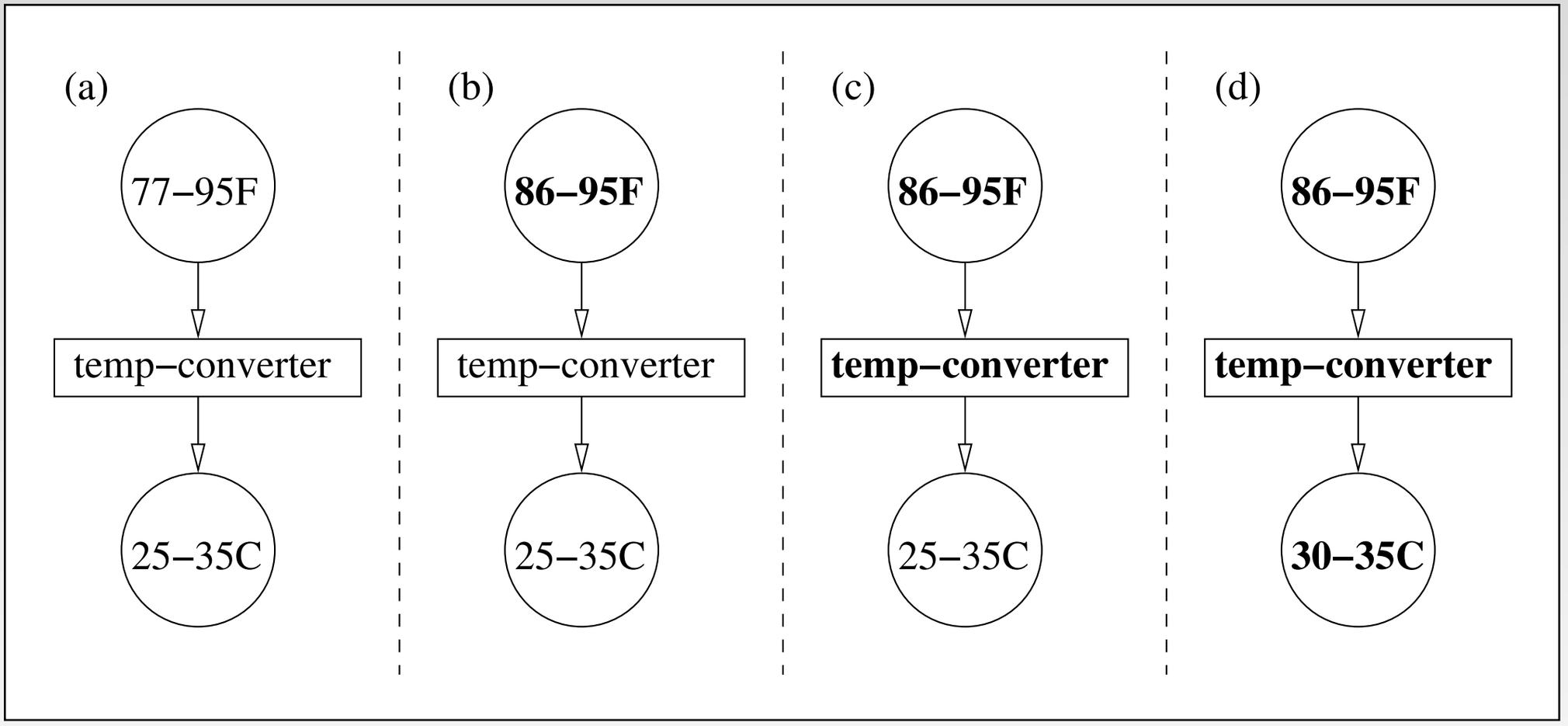
What Are Propagator Networks?

- Concurrent programming paradigm
  - Message-passing
  - Separates processing from state
- Network of propagators and cells
  - Propagators
    - Perform computation
    - Usually stateless
  - Cells
    - Contain state
    - Updated by propagators

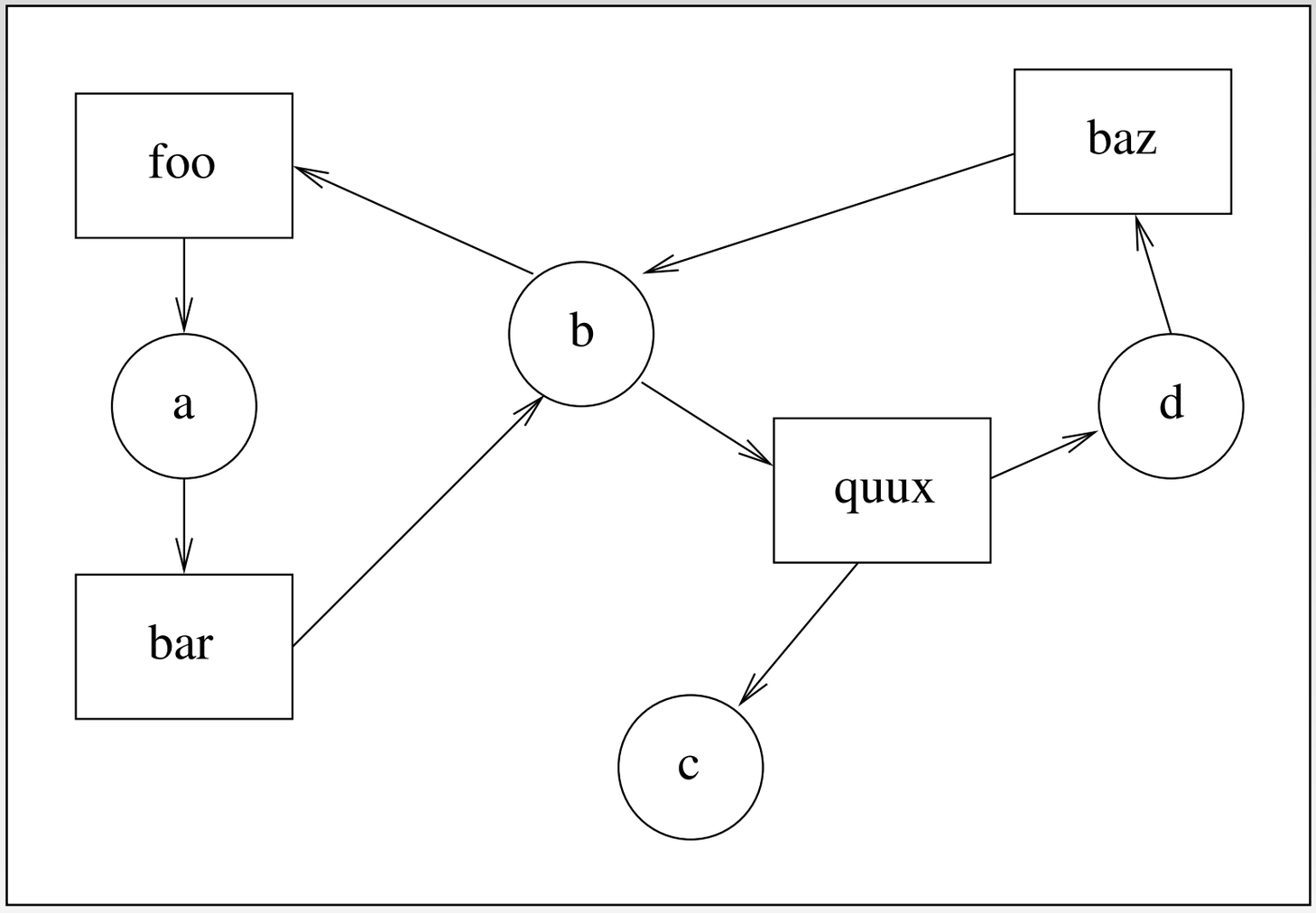
Cells Merge
Partial Information



Propagators Refine Cells



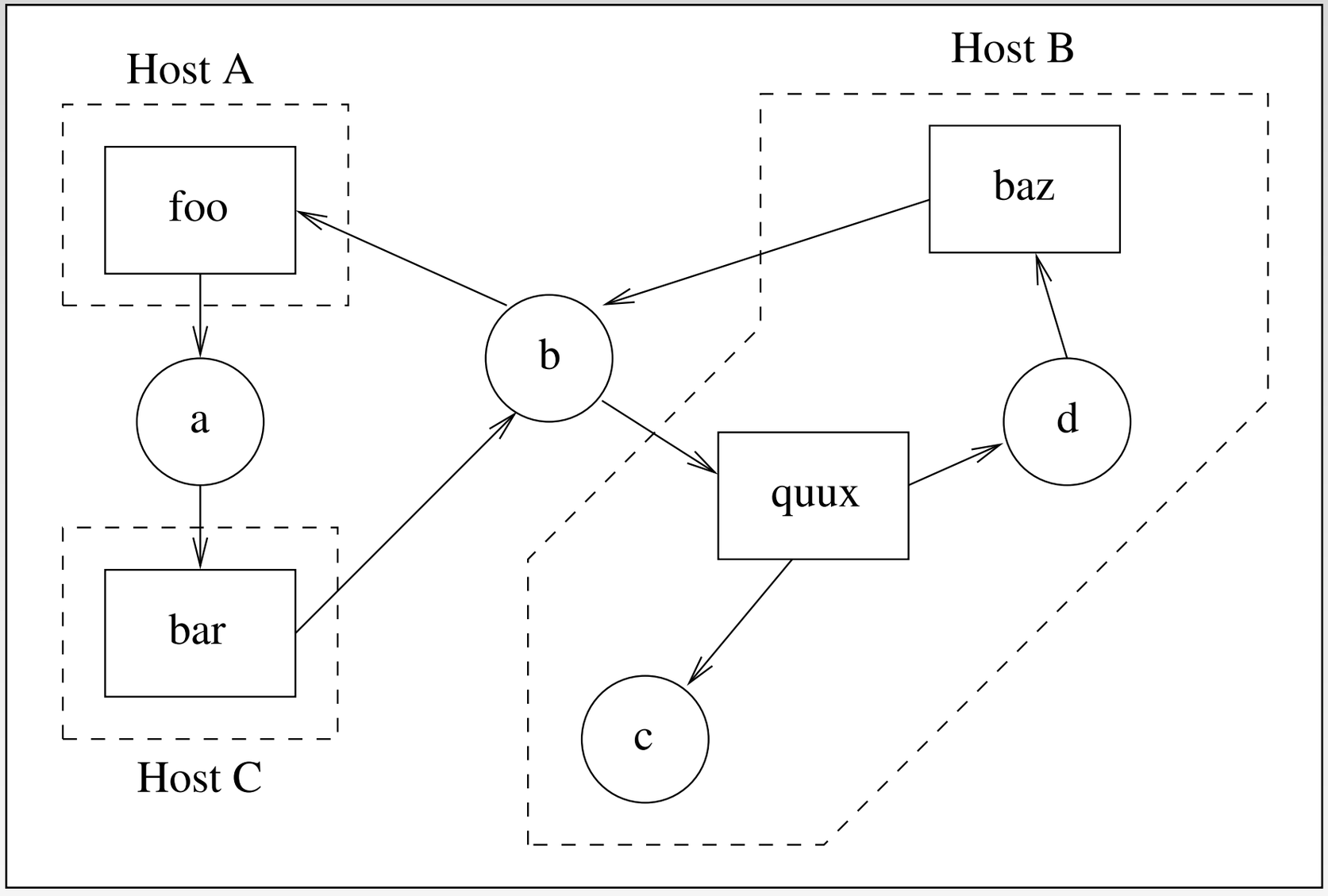
Propagator Networks May Be Complex



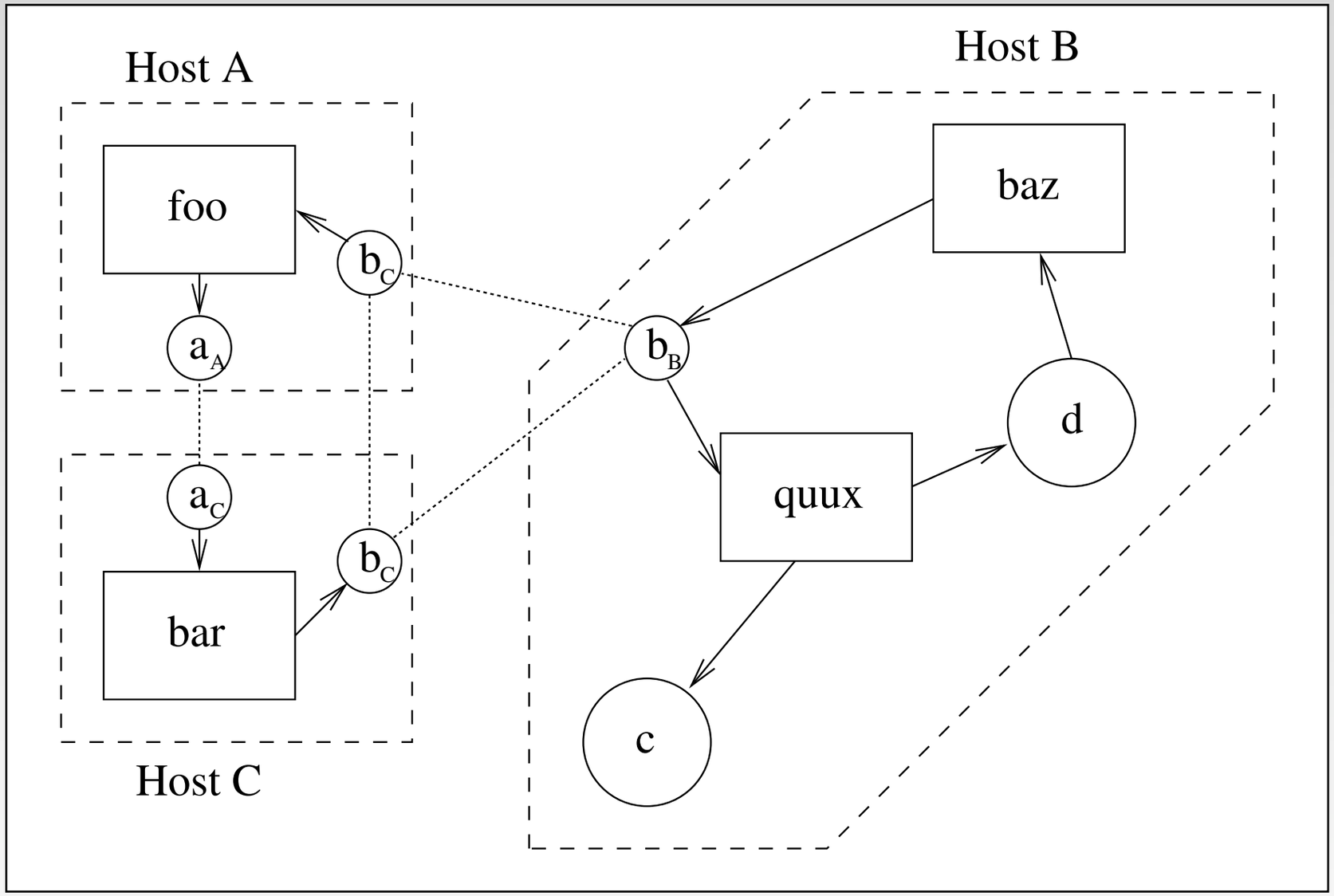
Distributed Computing with Data Propagation

- Distribute propagation by distributing cells
- Remote hosts can have different propagators
- Cells are duplicated by forwarding updates
- Propagators wake when updates received from remote cells

Distributing Propagation



Distributing Propagation



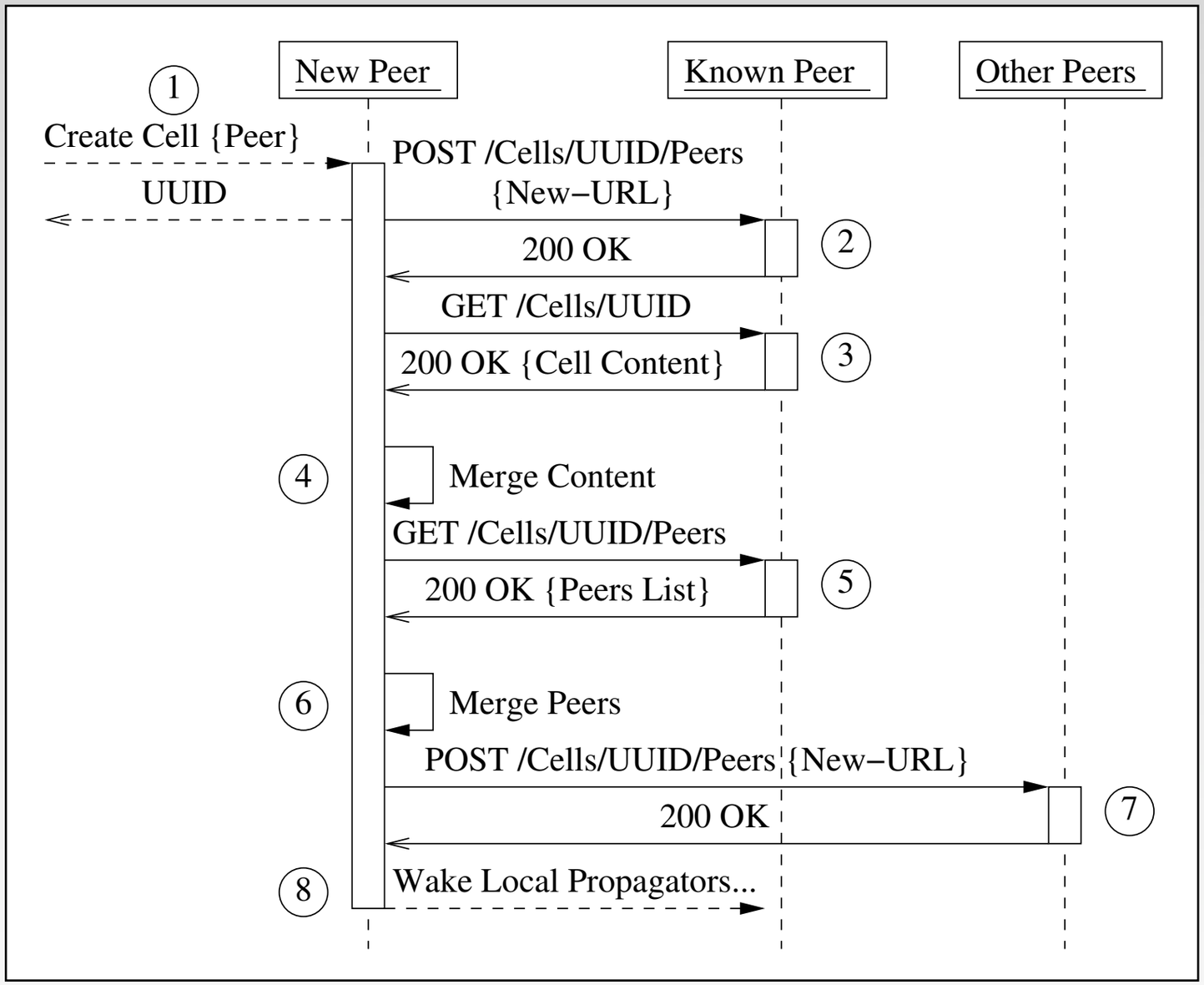
Refinement Constraints

- Idempotence
  - Don’t care how often we see the same update message
- Associativity
  - We can discard extra information
- Commutativity
  - Don’t care about ordering of update messages
- Monotonicity
  - Can refine partial-knowledge without deletion

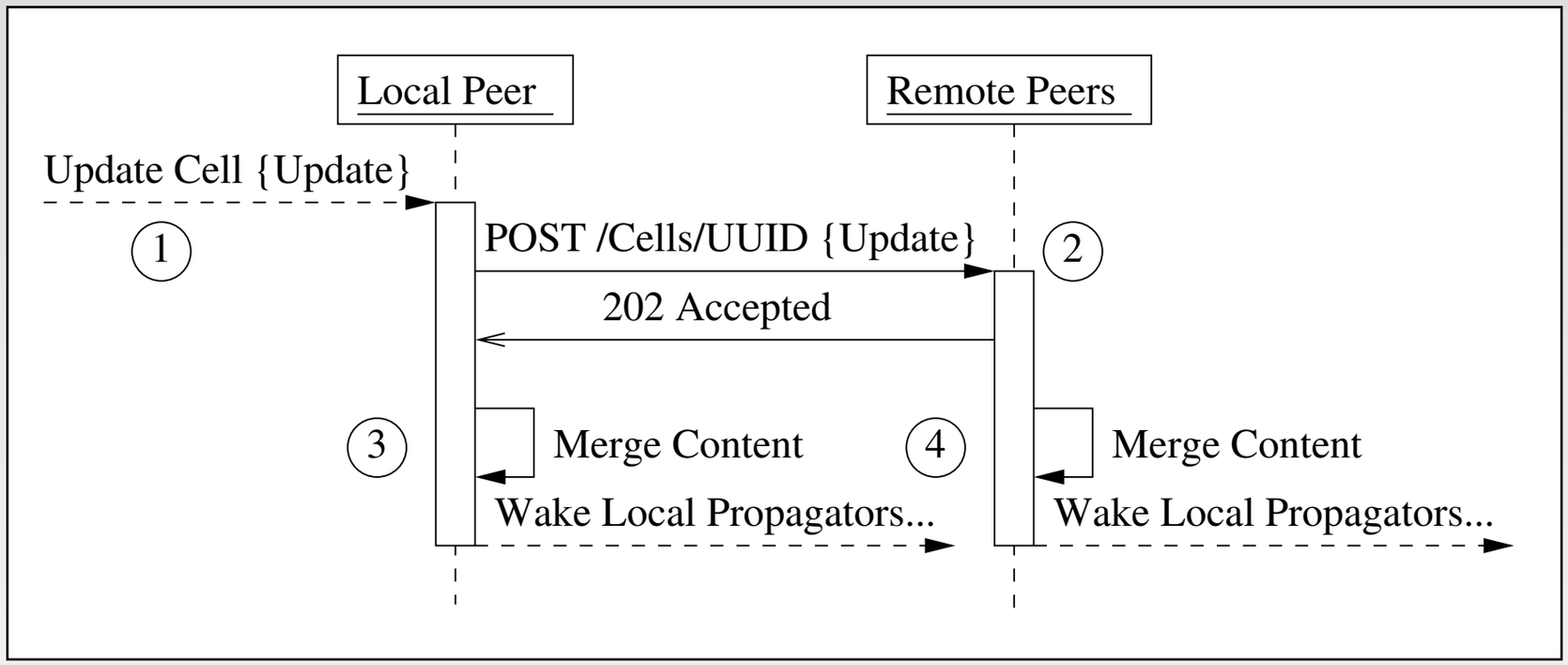
Moving Past REST

- “Representational State Refinement”
- Build on Client-Server with P2P overlay
  - Client-Server is too brittle, fault intolerent
  - Peer-to-peer better for distributed applications
- A new operation: Refine/Update
  - Needs an “idempotent POST”
  - PATCH (RFC 5789) has semantics, lacks idempotency

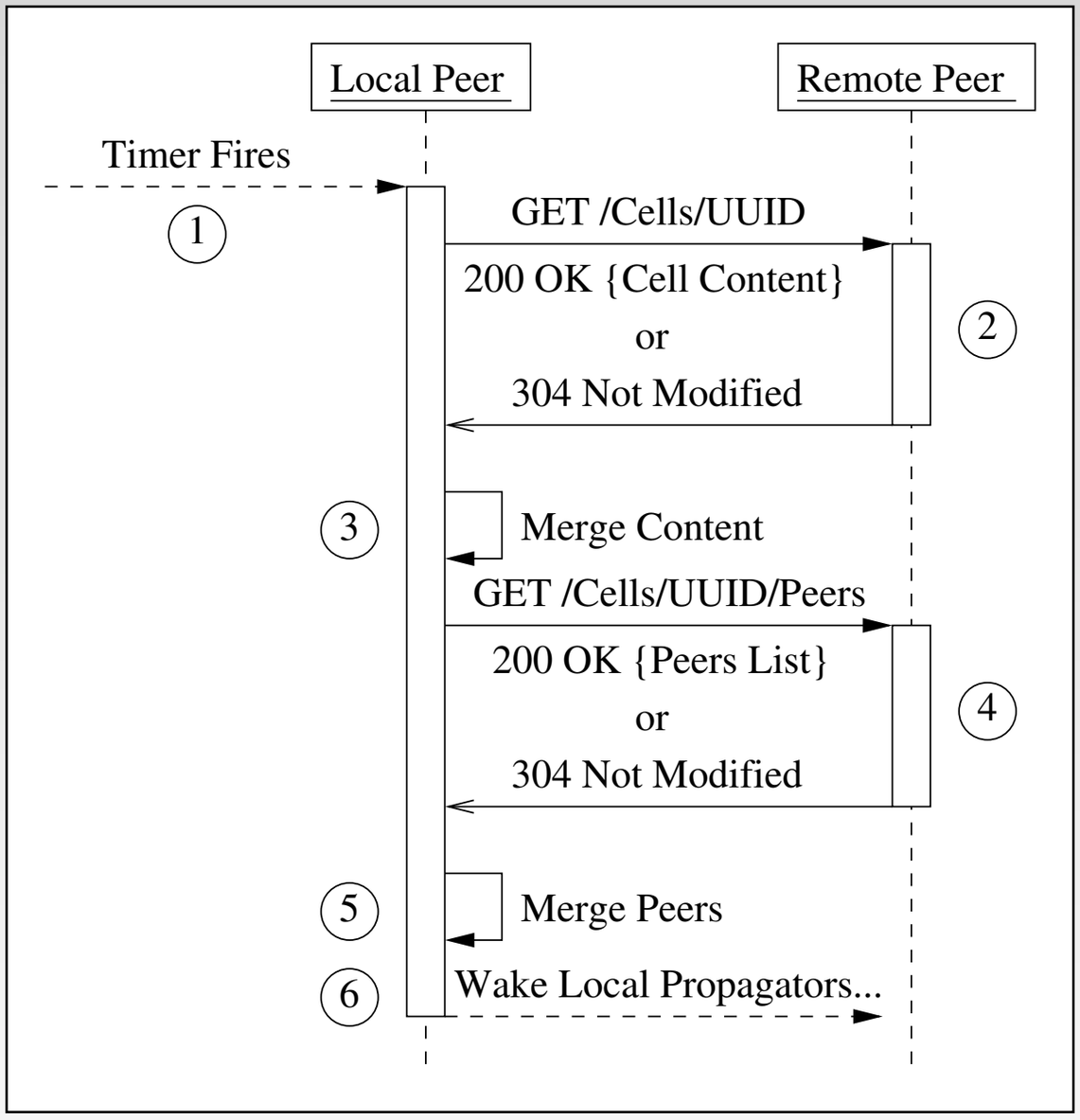
How It Works: Joining the Network



How It Works: Sending an Update



How It Works: Resynchronization

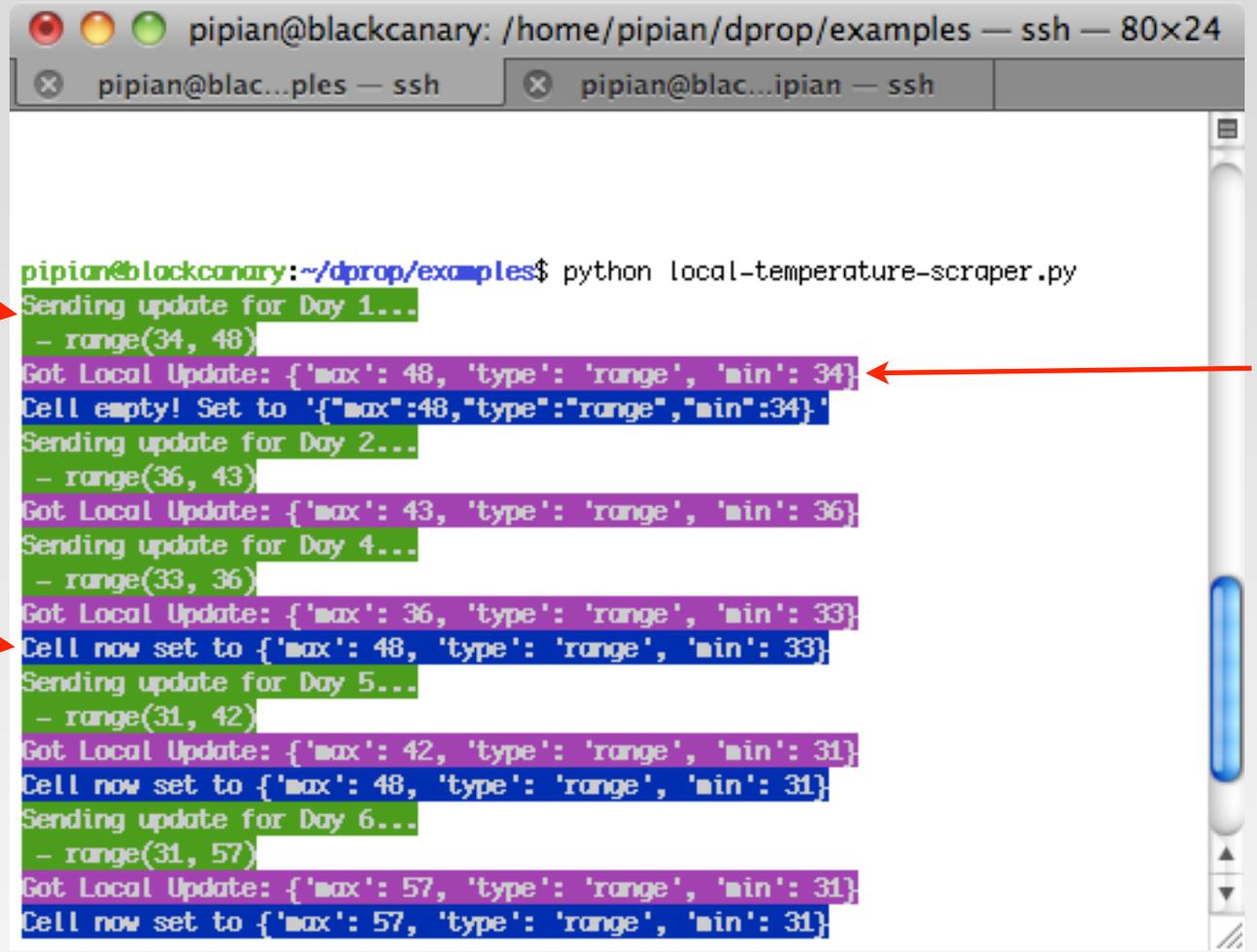


Demo

What to Watch For

Sending an Update

Refining the Cell



Receiving an Update

Behind the Scenes

Create the cell

Update peers

Send an update (local and remote)

Merge the update into the cell

pipian@blackcanary: /home/pipian — ssh — 80×24

pipian@blac...ples — ssh | pipian@blac...ipian — ssh

```
pipian@blackcanary:~/dprop$ python dpropman.py
Initializing DPropMan object
Received registerCell() call for c4add3d0-3cf3-11df-9879-0800200c9a66
Creating cell!
Setting up cell c4add3d03cf311df98790800200c9a66
New peersEtag: C6819D2F
c4add3d03cf311df98790800200c9a66 received updateCell() call
c4add3d03cf311df98790800200c9a66 sending out UpdateSignal
c4add3d03cf311df98790800200c9a66 attempting to update peers
Setting up peerUpdate thunks...
c4add3d03cf311df98790800200c9a66 received data() call
c4add3d03cf311df98790800200c9a66 received changeCell() call
c4add3d03cf311df98790800200c9a66 received updateCell() call
c4add3d03cf311df98790800200c9a66 sending out UpdateSignal
c4add3d03cf311df98790800200c9a66 attempting to update peers
Setting up peerUpdate thunks...
c4add3d03cf311df98790800200c9a66 received data() call
Traceback (most recent call last):
  File "dpropman.py", line 1190, in <module>
    main()
  File "dpropman.py", line 1187, in main
    mainloop.run()
KeyboardInterrupt
pipian@blackcanary:~/dprop$
```

Future Work

- Security
  - Secure data on network
  - Access Control
- Extending with provenance
- Scalability
  - Will it scale to large networks?
  - How can we generate networks on the fly?

Questions?