Beyond Named Function Networking

A) Immutable Data

B) Concurrency Control

I share this view with JMS, UPenn: Every protocol is a distributed computation.
A) Immutable Data

- In Named Function Networking, it first looked like an embarrassing constraint: “only works for immutable data” (no latest-version, please)

- My lesson learned: Immutability is one of the major wins. *overwriting memory cells is the curse of computer science* seems natural among functionally inclined people …

- In ICN, we should **leverage immutable data much more**:
  - append-only repos (more on this on the next slide)
  - kill the “version” field (move it to an architecturally safe place), think in “time cuts”
  - time-aware namespace management (Dave Reed’s tech rep 205 on NAMOS, 1978, titled “NAMING AND SYNCHRONIZATION IN A DECENTRALIZED COMPUTER SYSTEM”)
  - this would be a cool demo: *reverse debugging over ICN*
B) Concurrency Control, incl Consensus

• The taboo of our community: the “publish” API
  --> there is the get(), called interest, but not the set(), duh

• Any non-toy distributed application has to solve “conditional publishing”:
  only set(A, contentA) if I can also set(B, contentB)
  classic example: only decrease the balance if sufficient credit. … = a transaction

• Distributed computing is about handling race conditions, handling consensus
  … no answer in the community (sync is not enough) — Where is RAFT for ICN?

• Lack of dialogue with database community, little distributed systems mindset
  --> DB Zeitgeist: “higher-order databases” — Where is the “higher-order ICN?”
C) The Elephant in front of the house

- Google **PubSub**: (pubsub was meant to be a WAN thing, at most)
  - global
  - reliable delivery
  - crash resilient/HA
  - at scale
  A dream for developers: Implementing complex distributed apps in a snap, all nasty low-level problems solved for you.

Once you can **disseminate named data** with so much ease: who wants to still look into ICN?

Risk to be swept way by high-level development.