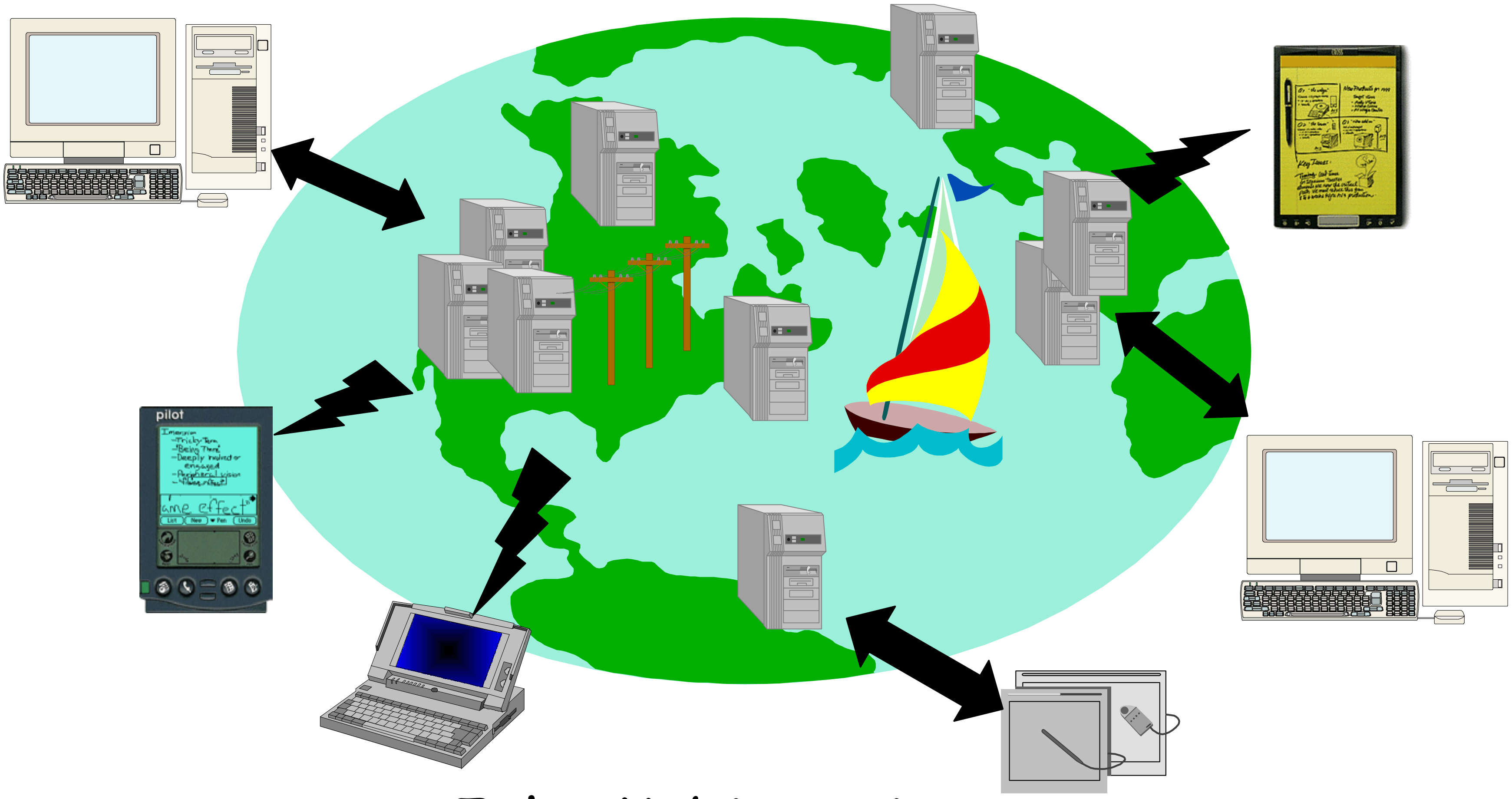
OceanStore: Data Security in an Insecure world

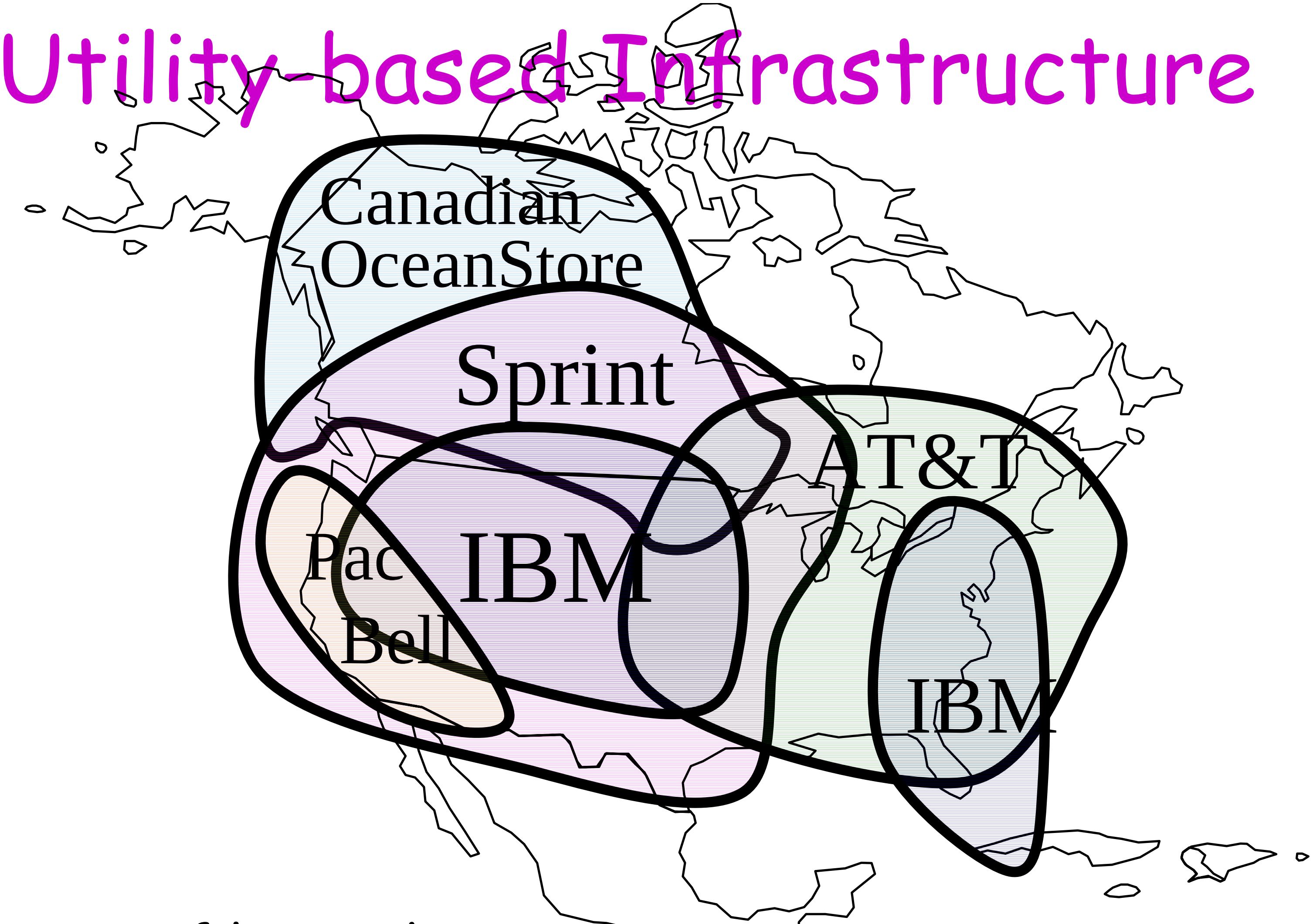


John Kubiatowicz

OceanStore Context: Ubiquitous Computing

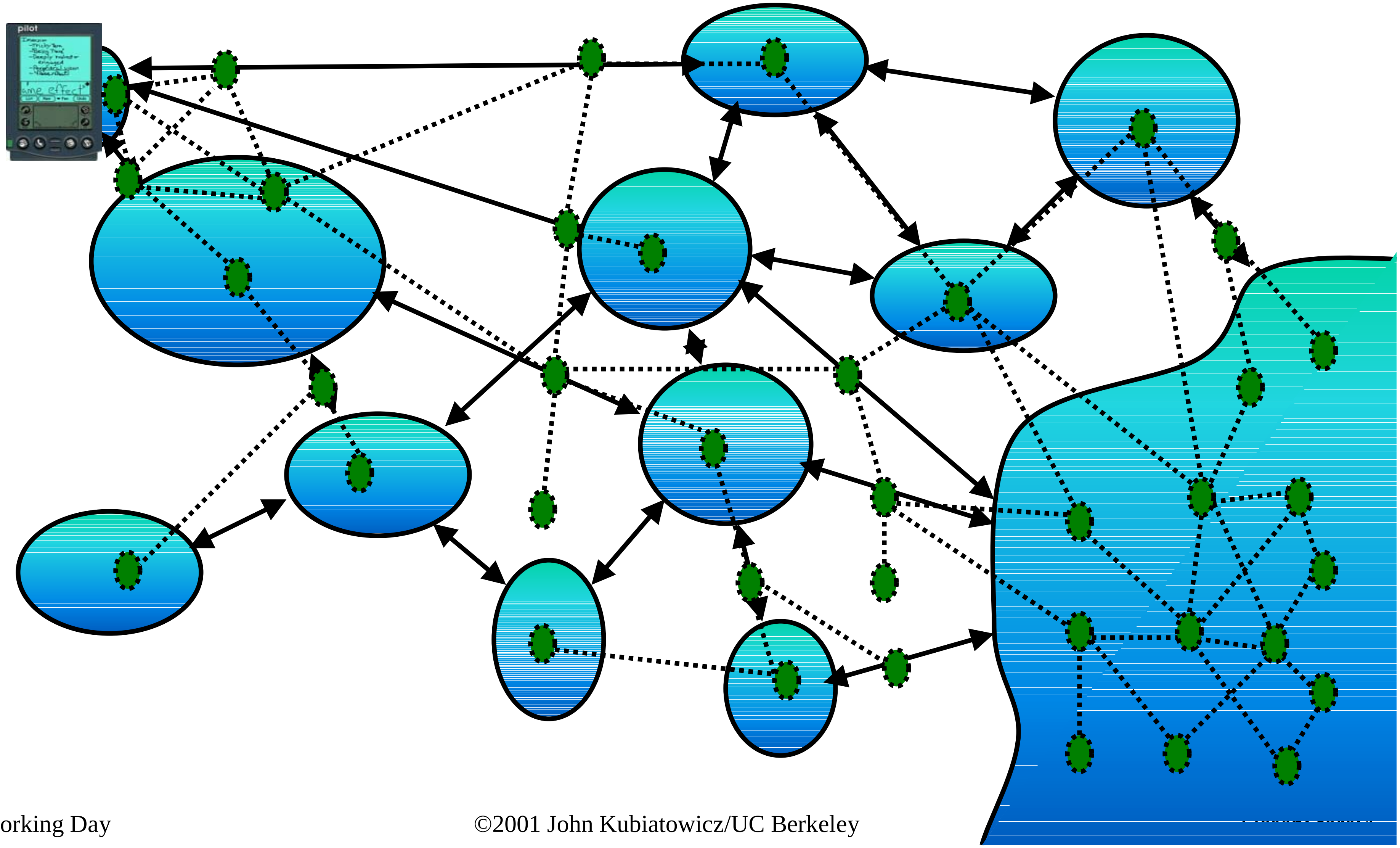
- Computing everywhere:
  - Desktop, Laptop, Palmtop
  - Cars, Cellphones
  - Shoes? Clothing? Walls?
- Connectivity everywhere:
  - Rapid growth of bandwidth in the interior of the net
  - Broadband to the home and office
  - Wireless technologies such as CMDA, Satelite, laser
- But: Where is persistent information?
  - Must be the network!
  - Utility Model

Utility-based Infrastructure



- How many files in the OceanStore?
  - Assume 10^{10} people, 10,000 files/person (very conservative?)
  - So 10^{14} files in OceanStore!
  - If 1 gig files (ok, a stretch), get almost 1 mole of bytes!

Basic Structure: Untrusted, Peer-to-peer Model

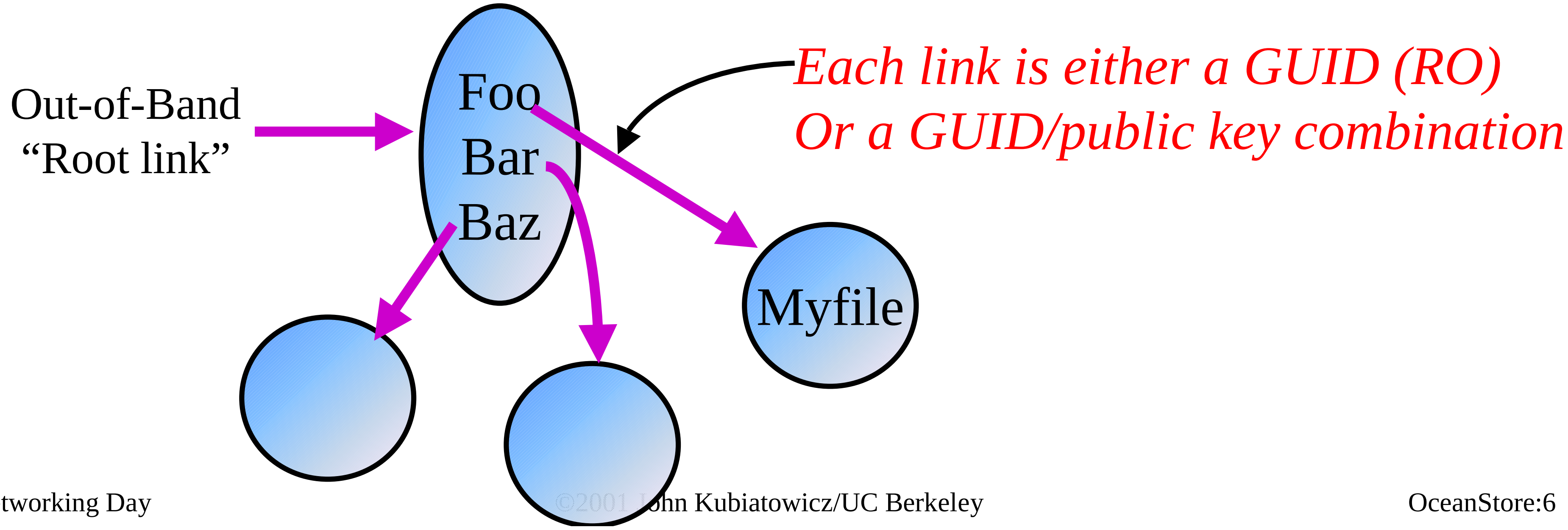


But What About Security?

- End-to-End and Everywhere Else!
  - *Protection at all levels*
  - *Data Protected Globally*
  - *Attacks recognized and squashed locally*
- How is information protected?
  - *Encryption for privacy*
  - *Secure Naming and Signatures for authenticity*
  - *Byzantine commitment for integrity*
- Is it Available/Durable?
  - *Redundancy with continuous repair*
  - *Redistribution for long-term durability*
- Is it hard to manage?
  - *Automatic optimization, diagnosis and repair*

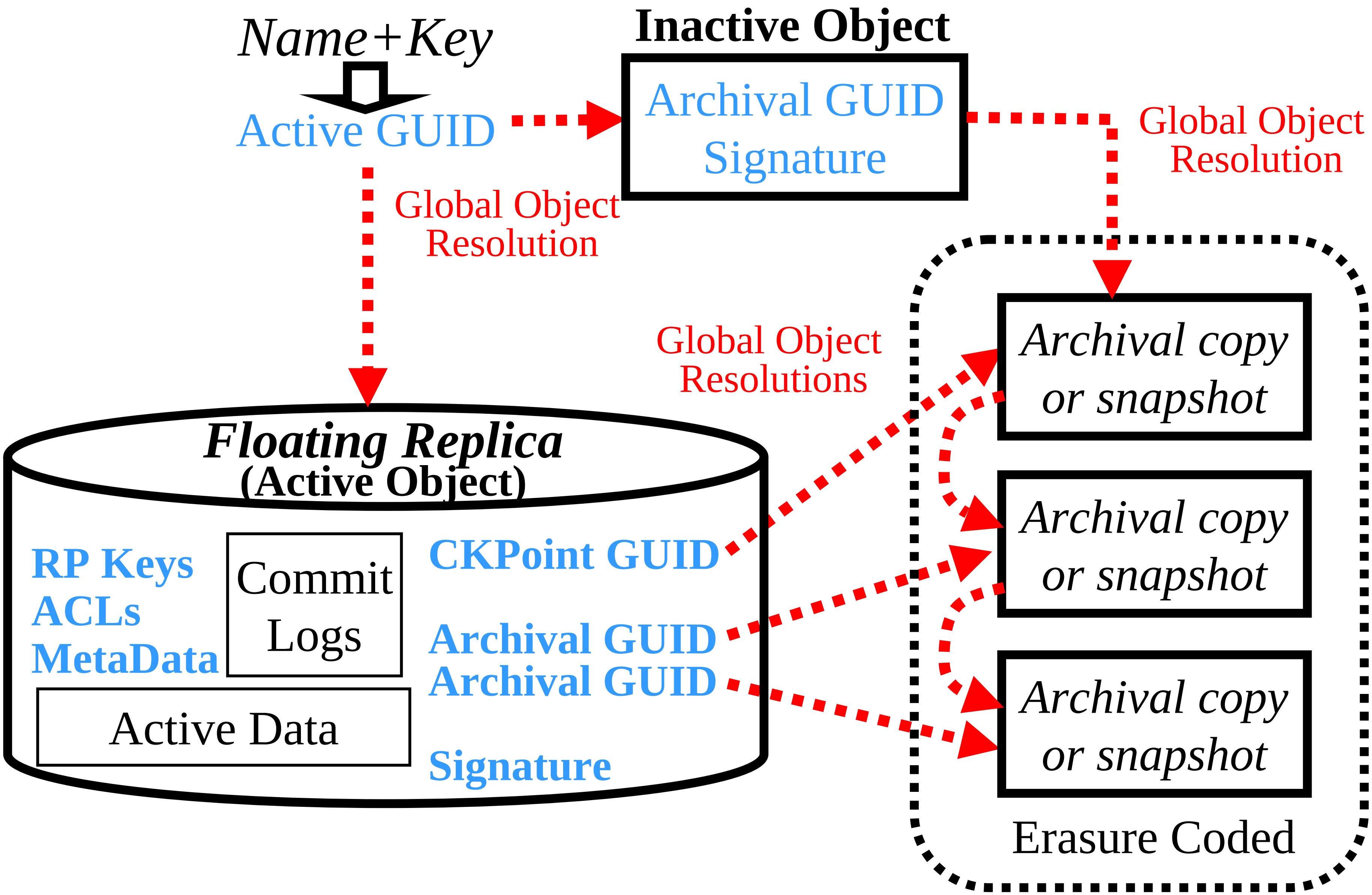
Secure Naming

- Unique, location independent identifiers:
  - Every *version* of every unique entity has a permanent, *Globally Unique ID (GUID)*
  - GUIDs derived from *secure hashes* (e.g. SHA-1)
  - All OceanStore operations operate on GUIDs
- Naming hierarchy:
  - Users map from names to GUIDs via hierarchy of OceanStore objects (*ala SDSI)*



Each link is either a GUID (RO)
Or a GUID/public key combination

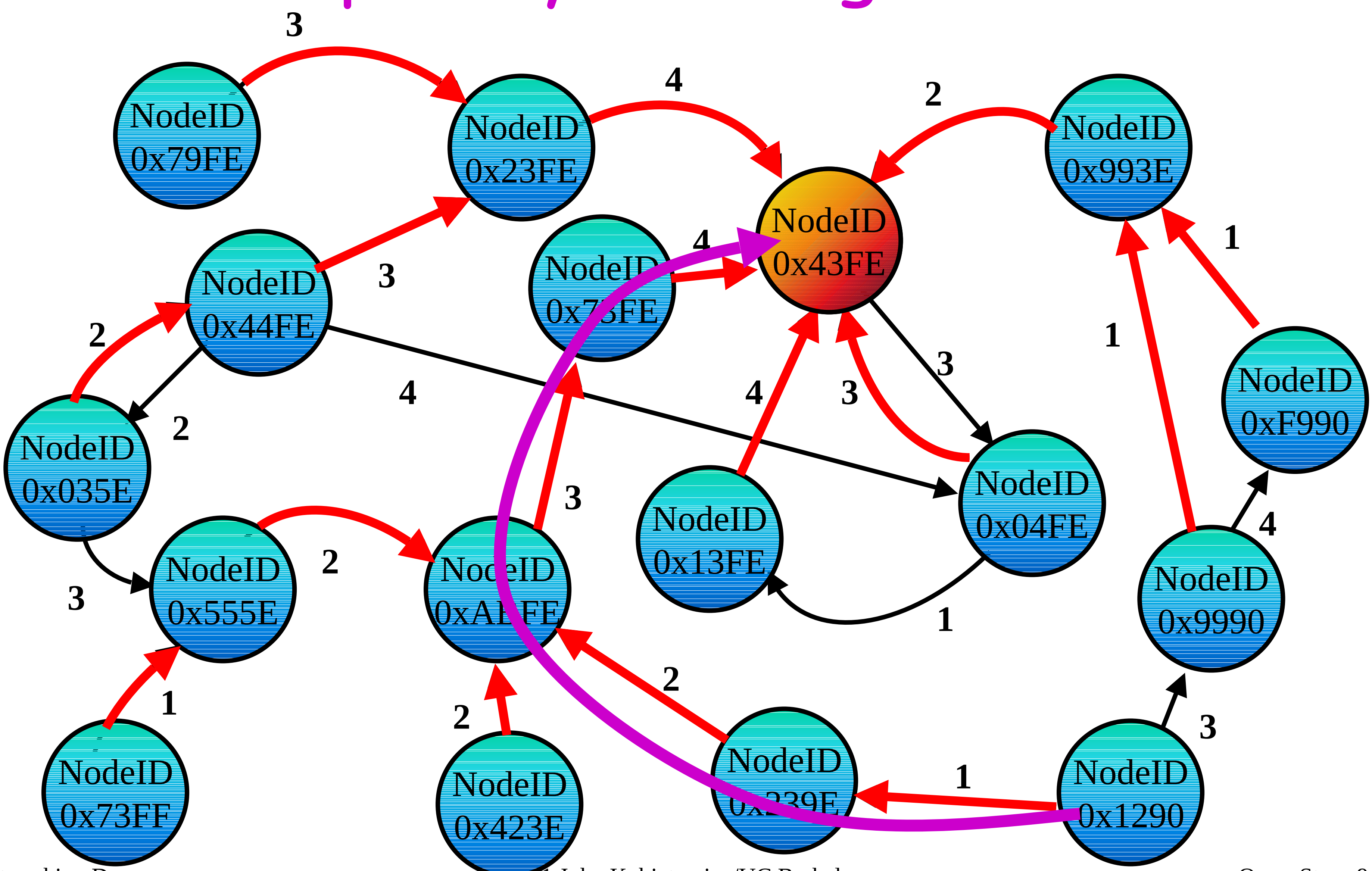
GUIDs ⇒Secure Pointers



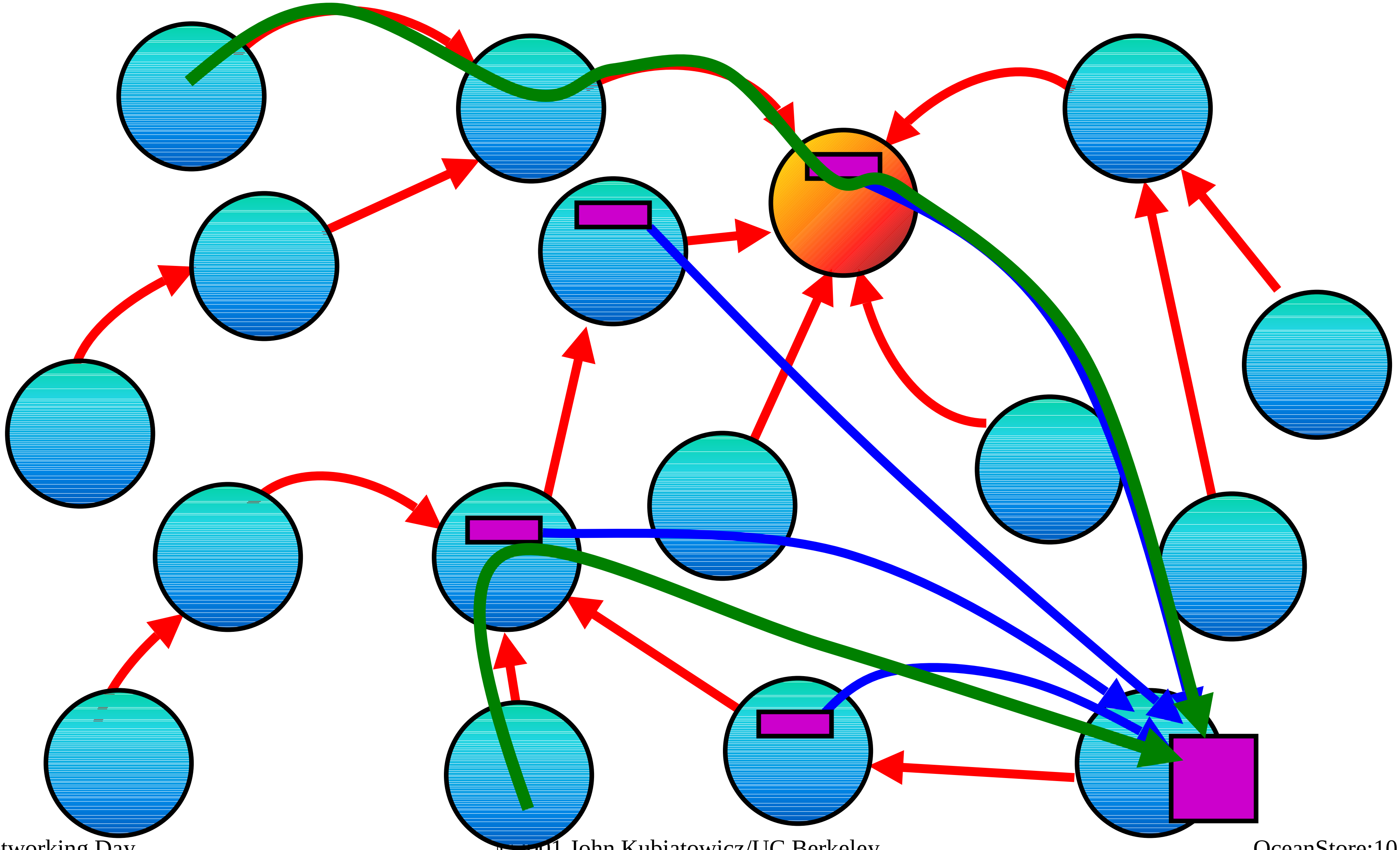
But What About the Red Arrows?

Location-Independent Routing!

Start with:
Tapestry Routing Mesh



Then add:
Location-Independent Routing



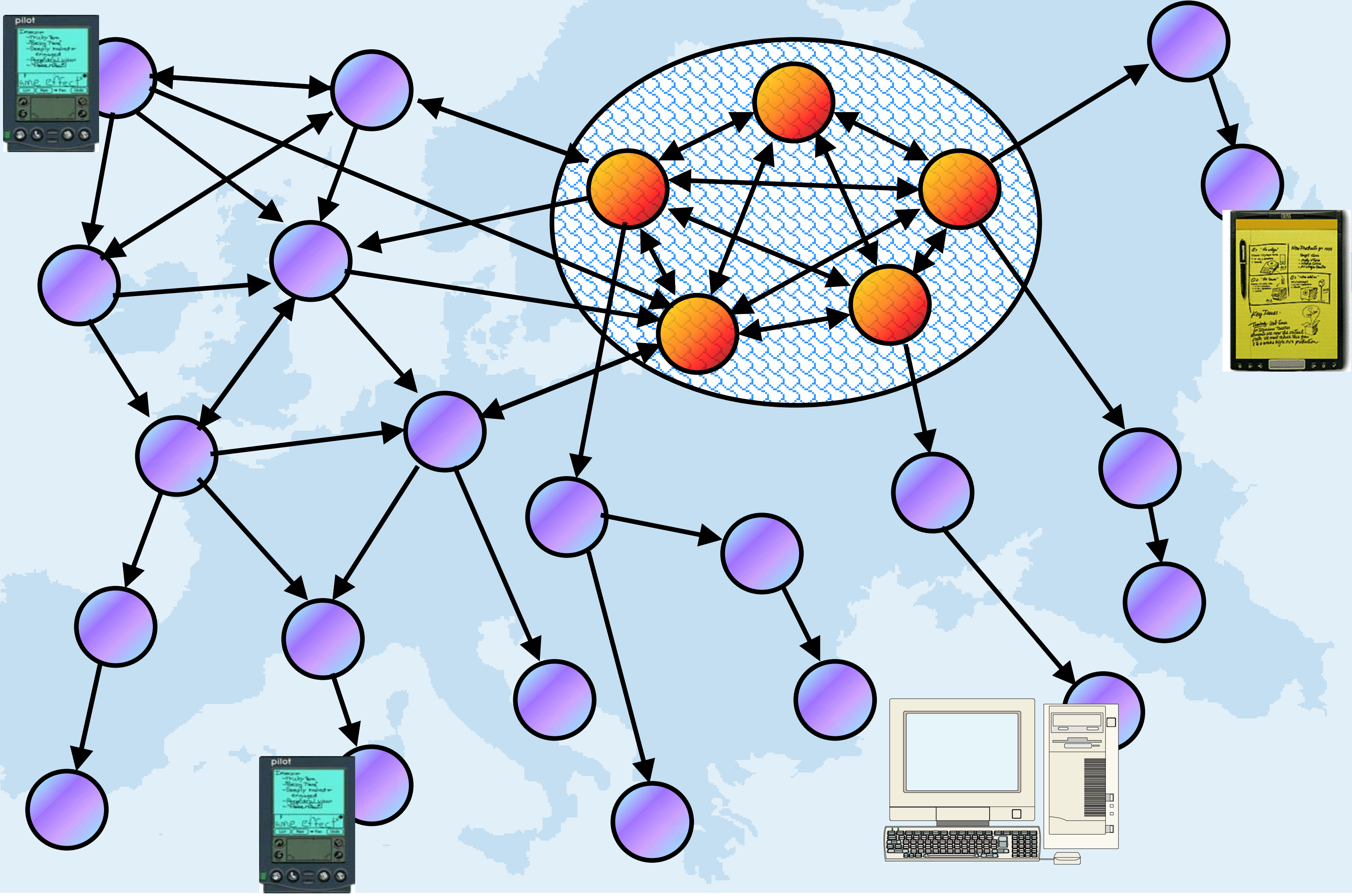
Secure Routing

- Node names are hash of public key
  - *Requests can be signed*
  - Validate Responses in Request/response pairs
- Data validation built into network:
  - *Pointers signed*
  - Publication process verified
  - Responses from servers verified by checking GUIDs
- Denial of Service resilence: locality/redundancy
  - *MACs along all links*: local suppression of DoS
  - *Multiple roots* to avoid single points of failure
  - *Multiple links* for rapid recovery
  - Pointers provide *locality:* Find closest version of object

What about Update Integrity?

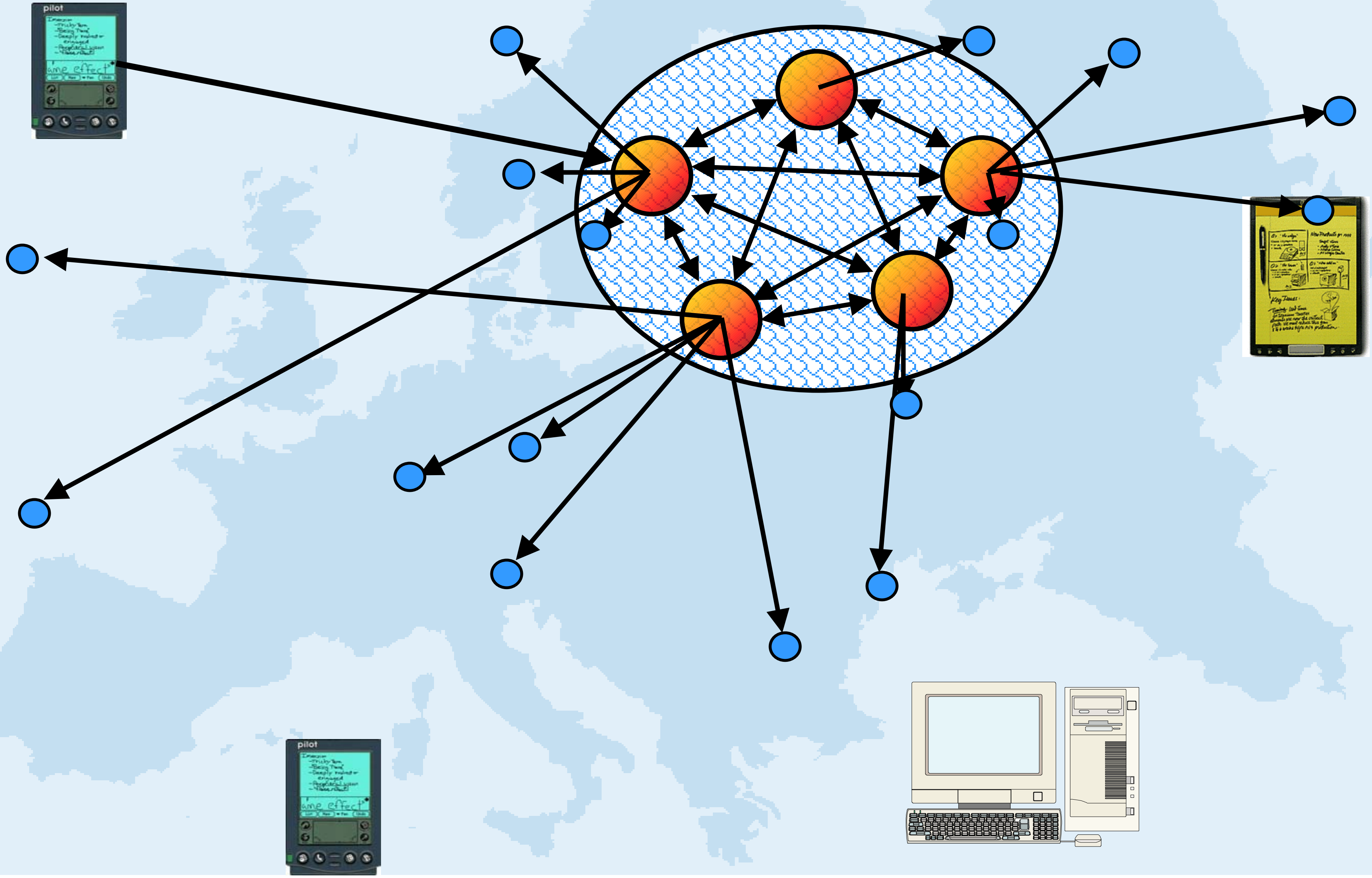
Byzantine Agreement!

The Path of an OceanStore Update



Consistency Mechanism applied directly to encrypted data!

Archival Dissemination Built into Update



Conclusion:
End-to-End and Everywhere Else

- Secure read-only data
- Secure the commitment protocols
- Secure the routing infrastructure
- Continuous adaptation and repair

For more information:
http://oceanstore.cs.berkeley.edu/