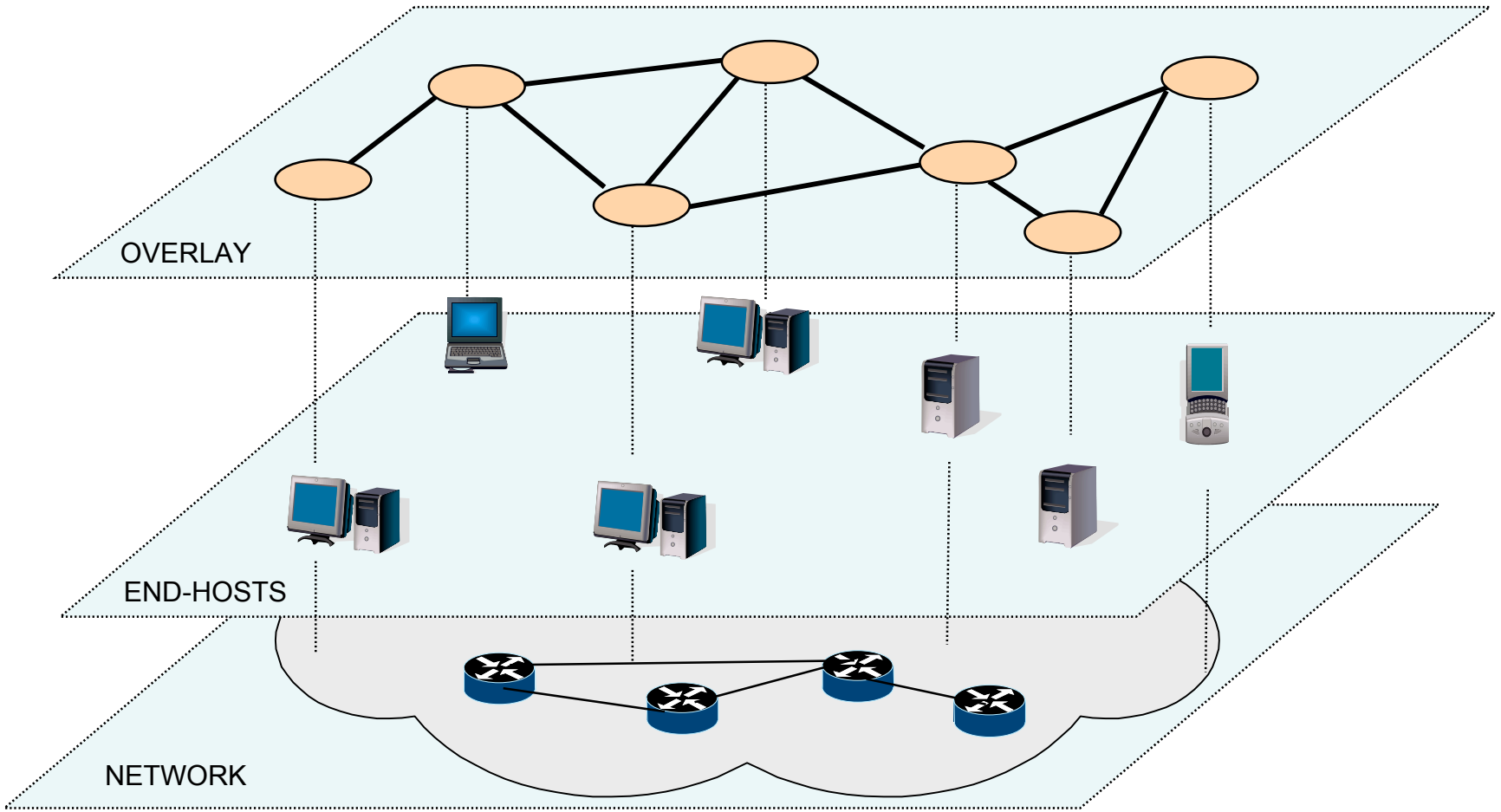
PROST: A Programmable Structured Peer-to-Peer overlay Network

Marius Portmann, Sébastien Ardon
Patrick Sénac

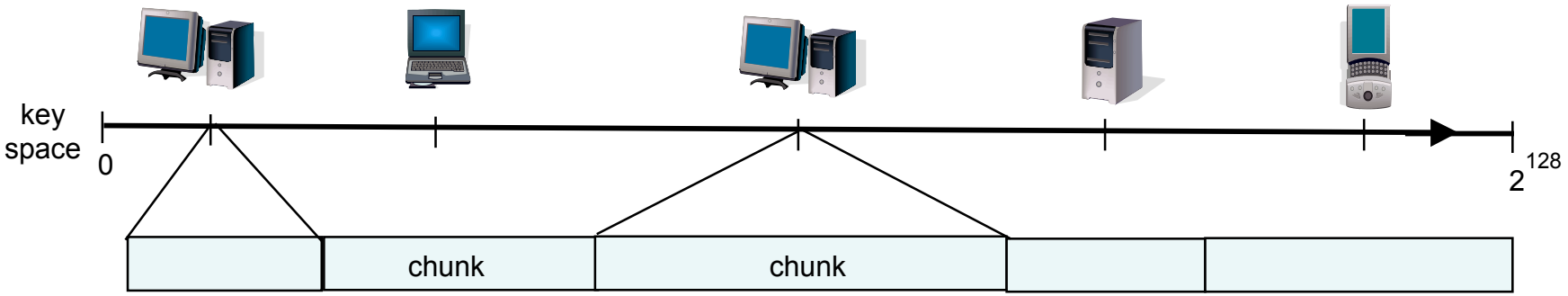
University of Queensland, Brisbane, Australia

ENSICA Toulouse, France

Context: overlay networks



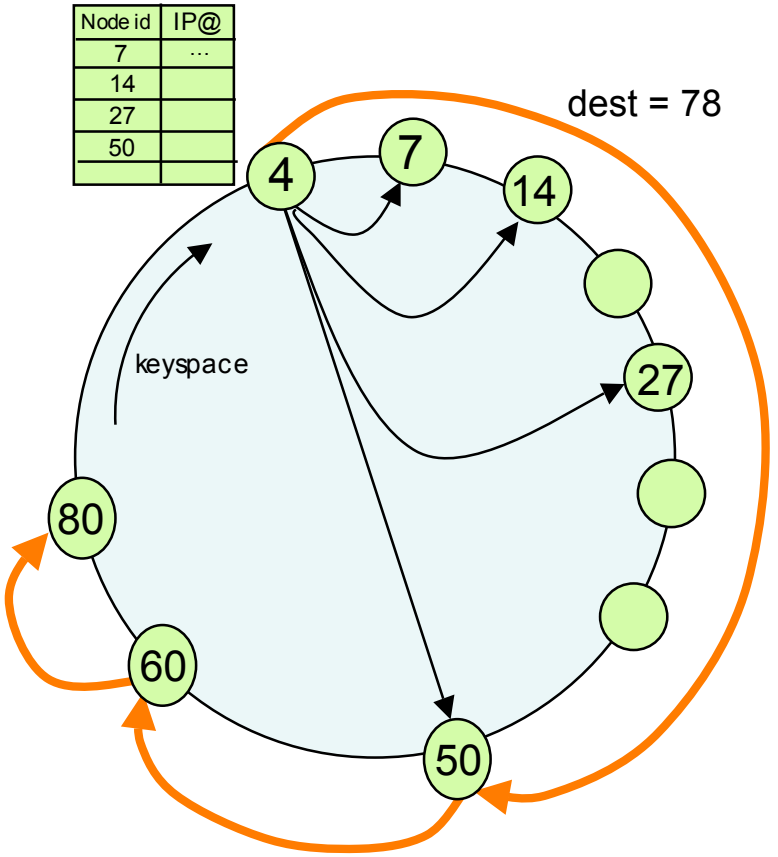
Structured peer-to-peer systems



- Each node assigned and responsible for a chunk of key space
- Mapping of resources to key space done using a hash function (e.g. SHA-1)
  - Most famous: Distributed Hash Tables

Example: DHT with CHORD

- Circular ID space (wraps)
- Each node stores (key,values) for all keys smaller than its node id but greater than previous node id.
- When node join/leave key,values are migrated to neighbor nodes
- hash function is SHA1: nice uniform random output (load balancing)
- Routing is used to find the node responsible for a particular key
- A node uses « shortcuts » along the ring: fingers
- Fingers are arranged exponentially starting from the node ID.
- Each node need to find the closest predecessor to the destination key
- O(log N) messages exchanged for routing from any source to any destination
- **Each node export simple hash tables manipulations functions (get/put)**



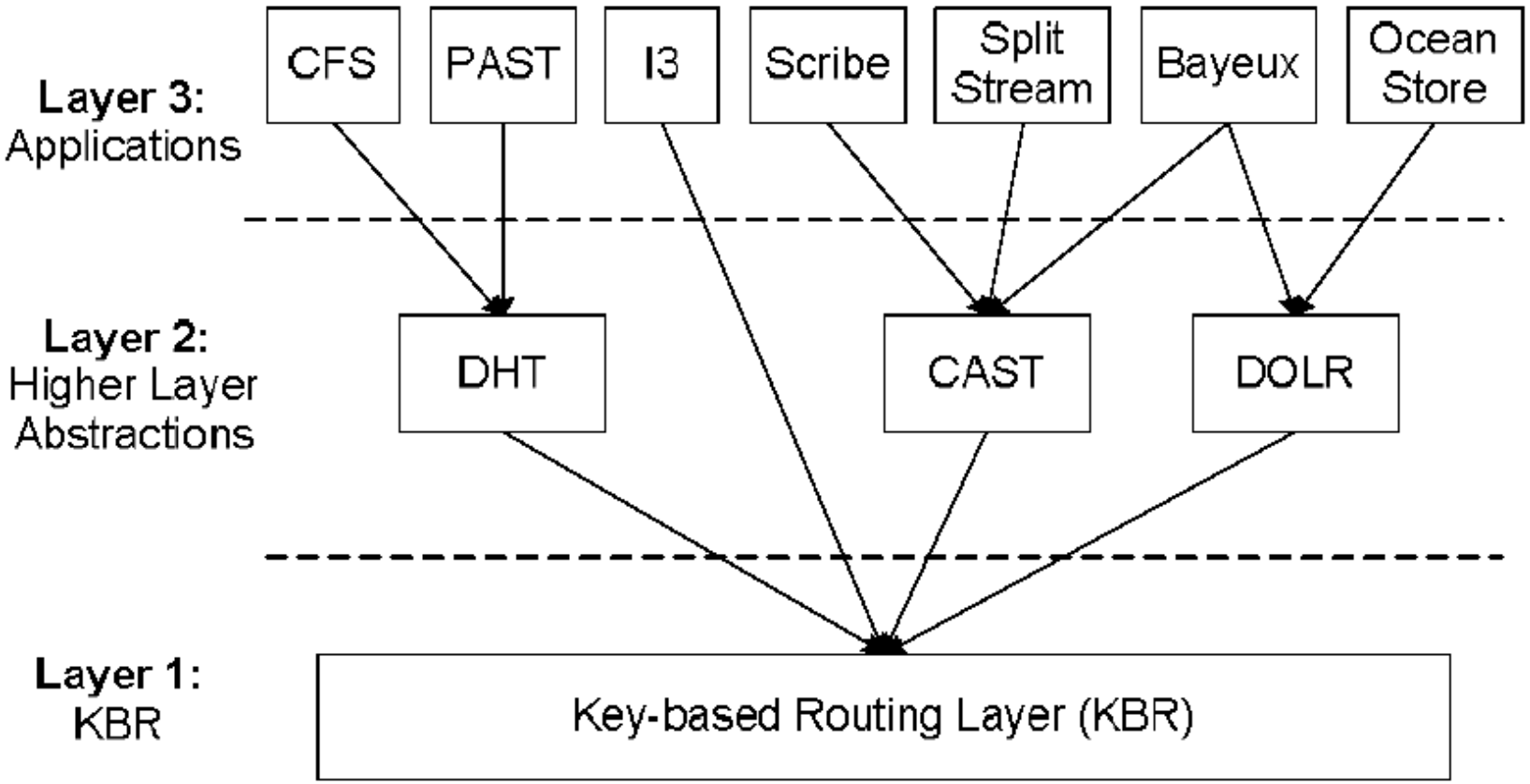
Structured P2P applications

- Cooperative File System (CFS)
  - File storage, uses CHORD at the block level
- PAST
  - large-scale, Internet-based, global storage utility that provides scalability, high availability, persistence and security
- I3
  - Communication primitive (indirection) based on CHORD
- SCRIBE
  - scalable application-level multicast infrastructure, event notification infrastructure
- SplitStream
  - High-bandwidth content distribution system based on application-level multicast
- BAYEUX
  - application-level multicast built on top of Tapestry
- OCEANSTORE
  - persistent and resilient data storage

PROST idea

- Proliferation of new “DHT”-based applications, each with their own topology
  - Re-inventing the wheel
    - One network per application!
  - Overhead: topology maintenance
- Idea: share the topology, routing and topology maintenance between applications
  - Common DHT API
    - Not enough: most applications need more than DHT
- Dynamically deploy application code on end-nodes (peers)
  - Take the programmable network paradigm to the peer-to-peer layer
  - Allow to deploy new applications on a large scale, by leveraging an existing peer base

A Layered model of structured p2p

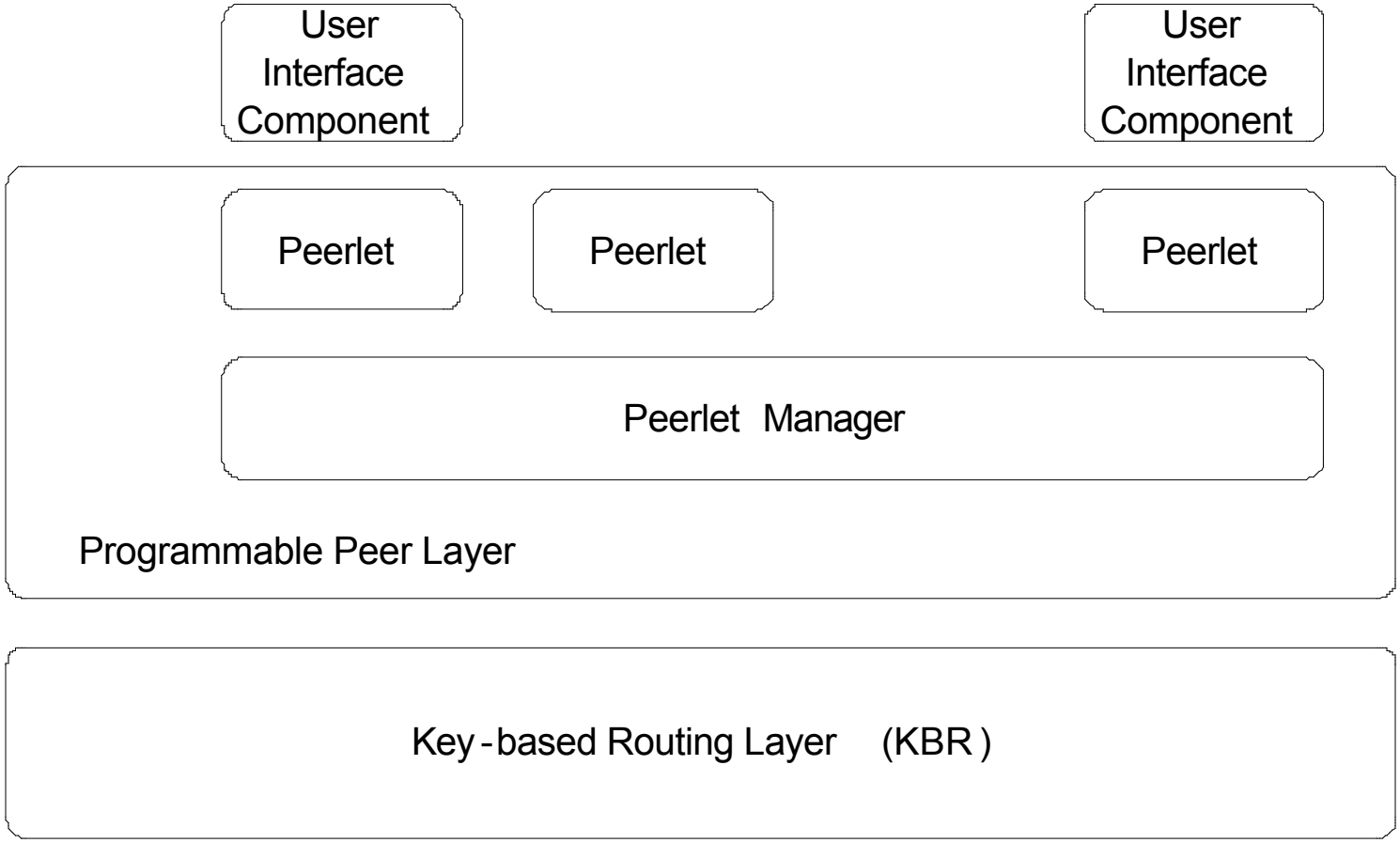


Dabek, F. et al, "Towards a common API for structured P2P overlays", IPTPS'03

PROST

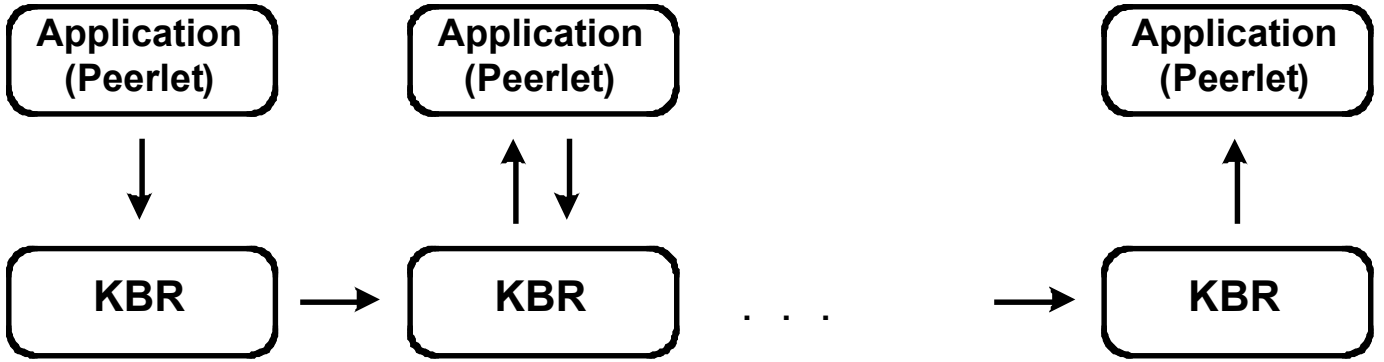
- Started as an enhanced DHT
  - Adding functionalities to DHT
  - Problem to maintain consistency and versioning
- Why not use DOLR?
  - Objects residing on peers to provide application functionalities
    - must be previously deployed
- Solution: dynamically deploy objects into peers, on-demand

Node architecture



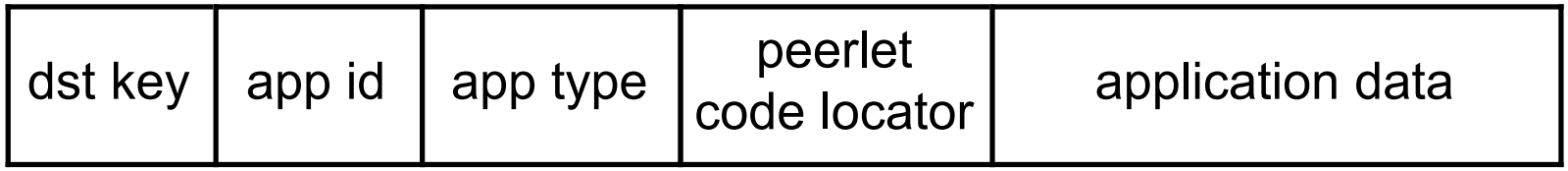
Two types of applications

End-node applications: file sharing, storage, etc



Per-hop applications: multicast communication, etc

PROST messages

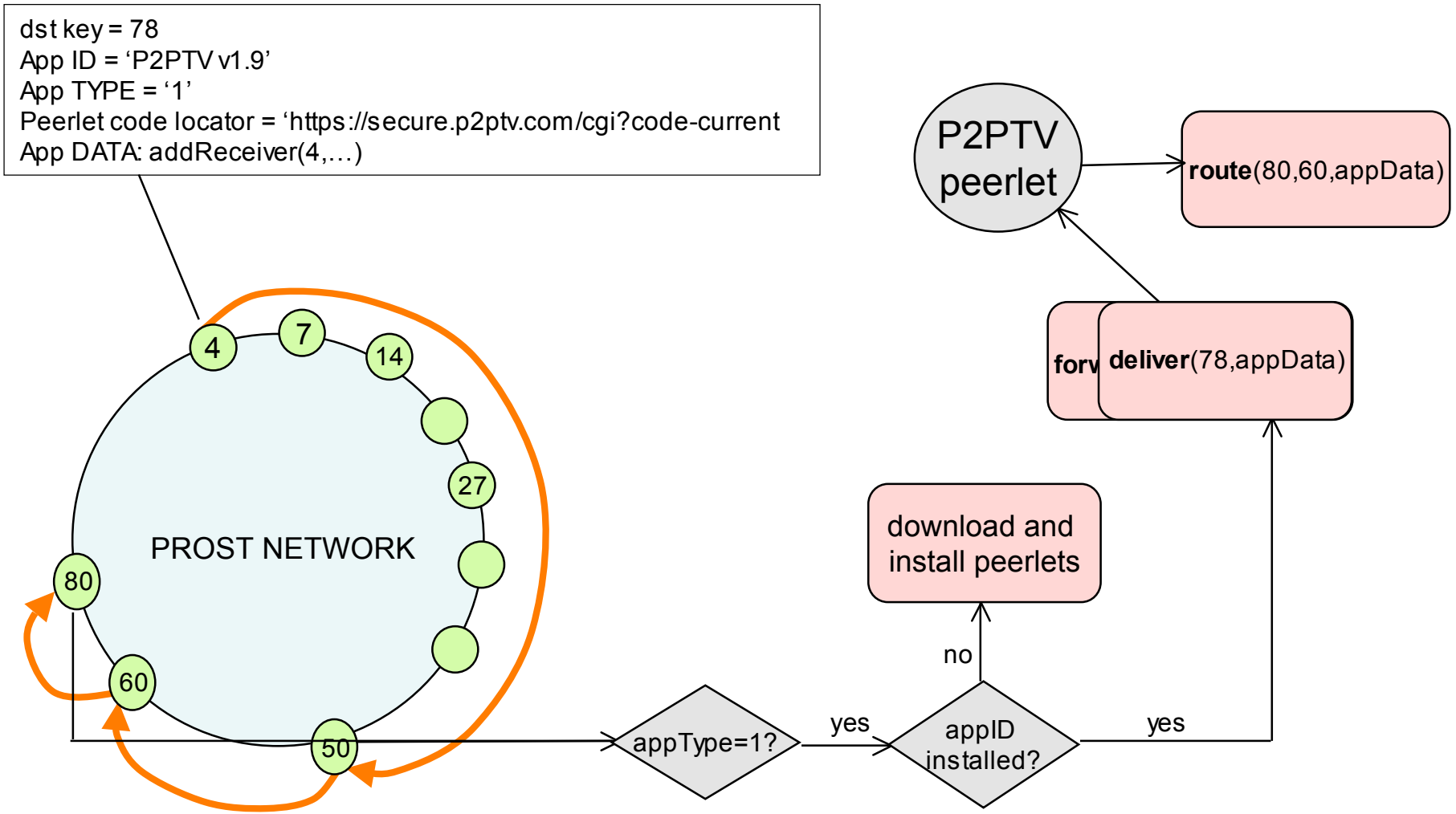


- DST KEY: destination key (ID) for this message
- APP ID: unique application-specific ID
- APP TYPE: 1: end-node application, 0: per-hop application
  - Analogy to Router Alert IP header field
- Peerlet Code Locator: where to download the code if needed
- Application Data: application-specific data, e.g. get/put operation and parameters

PROST API

- Based on Dabek et al. “Towards a common API for structured P2P overlays”
  - `route(k,msg,nextHopHint)`: send a message to destination key k
  - `forward(k,msg,nextHop)`: upcall to peerlet at intermediate nodes
  - `deliver(k,msg)`
  - Plus access to local routing information:
    - `local_lookup(…), neighbour_set(…), replicaSet(…), update(…), range(…)`

Typical operation



Node security policies

- Security policies are configured in each node
  - allow/deny peerlets to be installed, using
    - peerlet provider identity
    - application ID
    - local policy (deny/allow all)
- Allow graceful degradation of the service
  - if peerlet denied installation, try first node in replicaSet (recursively)

Challenges

- Similar to programmable networks:
  - resource management
    - admission control
  - security
    - trust model, code authentication
    - sandboxing
    - use to Crypto-based IDs
- New to p2p-based “active” networking
  - dynamicity of execution environment: nodes join and leave
    - process migration?

Implementation

- Implementation with CHORD, in java
- dynamic code execution through network class loader
- No resource management, plan to use existing system such as KaffeOS
  - support the OS abstraction of *process*
  - process separation
  - precise memory and CPU accounting
- No process migration, for the moment consider process are stateless

Future work

- further investigate the use of strong verifiable identities
- admission control, resource management
  - use results from the AN community
- develop and promote PROST as a P2P research experimentation platform
  - implementation using the Bamboo DHT
  - large scale deployment over planetlab

questions