FARADS

Forwarding Directives, Associations, and Rendezvous

Aaron Falk, Bob Braden
USC ISI
September 17, 2002

Outline

- Architecture Overview
- Design Choices
- Implementation Notes

I. FARADS Architecture Overview

Summary of the architecture as defined by Dave Clark

FARADS: Possible Newarch Functional Abstraction

- Separate location from identity
  - 1. Support general mobility
  - 2. Support wide range of routing/forwarding architectures
  - 3. Support diverse naming schemes
    - May include e.g., anonymity, local names as well as global names.
  - 4. Cleanly decouple 2. from 3.
- Support range of mechanisms for end-system authentication despite this separation.
  - Including lightweight authentication

The FARADS Architecture

- Abstractions:
  - Entity
  - Association
- Mechanisms
  - Forwarding Directives (FDs)
  - Rendezvous
  - Slot

Entity

- The Entity abstraction generalizes the traditional application.
  - Might be: process, process group, entire machine, or cluster of machines
- Entities communicate with each other, using association(s).
  - Contains communication state for its association (as well as other state that is relevant to their higher-level function.)
  - Question: what about cwnd? MTU? rcvbuf?
- Entities are the unit of mobility – an entity moves as a unit.

Association

- Association = logical comm link between two entities
  - Sequence of data packets
  - Shared communication state
- An entity may have multiple concurrent associations
- Association within a particular entity is labeled with a local Association Identifier (AID)
  - A handle for locating associated comm state
  - Unique within entity, not necessarily within node or across nodes. Hence, must be local to entity.
- AID is invariant during mobility, i.e., as FD changes
  - A "fate-sharing region"

Forwarding Directive

- Tells the "network" how to deliver a packet to an entity – or more strictly, to a slot within which the entity is instantiated.
- FD supports a range of forwarding mechanisms
  - Might specify globally-unique address, e.g., a network attachment point (IP address); FD ~ (IP addr, port#), or
  - Might specify a path/explicit route.
  - Might be inherently reversible, or not.
  - Might change in flight
  - May be independent of sender, or not.

Forward Directive (2)

- FD contents are opaque to entity.

The Red Line

- A "red line" separates forwarding (network) knowledge from entity (application) knowledge
- FD provides packet delivery (below the line)
- AID identifies association state (above the line)
- Some messiness in FD management
  - E.g., obtaining FDs, mobility awareness, etc
  - Network congestion needs to be shown to the association

Slots

- A slot is the local operating system interface to an entity.
- An FD actually delivers data to a slot, and hence to the entity, if any, currently occupying that slot.
  - If an entity moves to a different slot in the same (or different) end-system, the FD changes
  - Slots are like dynamically-allocated ports
  - *ISSUE: Can slots be well known? May be stable, but form of slot specification might be specific to one OS, for example.*

Rendezvous

- Establishing an association generally requires a procedure/mechanism called rendezvous.
- Entities wishing to initiate an association send a rendezvous string (RS)
- RS contains anything the receiving entity needs to establish an association
  - Examples:
    - TCP initialization
    - URL click-through tags
    - Authentication

FD Management

- FD Mgmt straddles red line
  - Tells entity things about the network
    - E.g., translates entity QoS needs to route preferences
  - Tells network things about the entity
    - E.g., notifies entity that packets from other end contain new source FD to prompt authentication
- Performs FD negotiation
- Performs site preparation for mobile entities

Mobility

- Several types:
  - Entity Mobility: entity moves to a new end-system
  - Physical Mobility: end-system moves to new network attachment point
  - Virtual Mobility: entity moves to a different slot (think “port”) in current end-system
    - Or: path changes during a connection
- All require FD changes
- Mobile entities can be found using agents

Agents

- Agents are a special type of entity that act as a helper for mobility
  - Required when mobile entity wants to be found in DNS
  - May be useful at other times (e.g., unexpected FD changes)
- Agents are special: they operate below the red line (they munge FDs) but have entity-like properties
  - E.g., they have associations with the mobile entity to maintain the FD mapping

Agents (2)

- An entity may have multiple agents
  - All agents require updating when FDmobile changes
- An agent may support multiple entities
- The agent function may be located anywhere along the path, including within the sender or receiver
  - Locating the agent within the network has preferable scaling properties

Problem & Undefined Areas:

- N-way associations (n>2)
  - E.g., middleboxes
- Multicast
- Quality of Service
- Routing Subsystem
- Overlays

- Consider i3?

II. Design Choices

Choosing an interesting and useful point in the space defined by the FARADS architecture

NewArch DNS (nDNS)

- An optional – albeit handy – way to obtain FDs and create RS
- Very similar to traditional DNS
- Returns globally reachable FD and a rendezvous template (RT)
  - RT tells the entity how to create an RS, possibly requiring local information

FD Negotiation

- An entity can request a path change via FD definition or negotiation
- Used for
  - expression of route preferences (WAN provider selection)
  - server selection (load balancing)
  - mobility
- Need a protocol here…

Agents – How it works

- A mobile entity, using a private association, loads a mapping (FDagent -> FDmobile) into the agent
- The mobile entity publishes FDagent in the DNS
- Two possible behaviors may be supported:
  - Incoming packets to FDagent are rewritten with FDmobile and sent out
  - Incoming packets to FDagent trigger a redirect message to the sender
- As FDmobile changes, the agent is kept up to date for new associations

Mobile End Systems

- If an entity knows it's going to a new FD, existing associations are notified (via FD Mgmt) that the source FD of the ME is going to change
- For unexpected mobility, the agent can be used as a meeting place
  - If an entity stops getting responses from a known ME, it can send a query to FDagent

Entity Moves to New End-System

- Locate & prepare a slot (how?)
  - Acquire new FD
- Provide new FD to entities engaged in associations
- Collect & move state to new location (how?)
- From new location, send an FD change to remote entities

Resynchronization

- Resynchronization needs to occur after an entity moves
  - Accounts for packets that might go to wrong FD
- End-to-end, i.e., agent not involved
- Could be a simple exchange of sequence numbers

Route Subsystem

- Currently assuming black box which assembles a working FD
- Implies a method of expressing route preferences
- FDs are composed of route fragments reflecting path preferences/new location
- May be nimrod-like using route fragments
  - Some work by Xiaowei Yang at MIT

Security

- Want to preserve “lightweight” nature of TCP pseudo-header
- Candidate solution: DCCP connection nonce
  - Each entity exchanges a random number at the beginning of a connection
  - When a nonce challenge is received, the XOR of the two random numbers is returned
  - When FD management indicates packets have arrived on an existing association with a new source FD, the connection nonce is exchanged
- Alternate, more secure solution: purpose-built keys (?)

Examples (TBS)

- Simple connection establishment
- Simple plus nDNS
- Mobile endpoint
- Route preference negotiation

Implementation

FARADS implementation performed at ISI

Overview

- Entities • processes
- fKernel • user-level process
- Network • overlay network of fKernels

Implementation Details

- C++
- User space for ease of debugging
- FARADS packets sent over IP with new protocol number
- BSD firewall code used to grab packets fKernel
  - (courtesy Ted Faber)
- FARADS kernel (fKernel) routes packets to correct slot
- FD Management, DNS, and simple apps exist as separate entities

Implementation - Packet Format

- FD = IP address + port number

Implementation – Status

- Ted's playground defines fKernel
- First apps:
  - Ping
  - Simple, unreliable file push
  - Simple DNS

Implementation – Plans

- Mobility
- Path negotiation
- Demonstrate simple scenarios
- Security stuff? HIP/IPSEC?

The End