SCRAPI Applications Programming Interface for RSVP

Bob Lindell
USC/ISI

Introduction

- Simple "Bare Bones" API
- RSVP application support using a few lines of code
- Avoid the introduction of RSVPisms (data structures, errors)
- Portability - requires RAPI and Unix

A Comparison to RAPI

- SCRAPI has no concept of a session
  - Flows are (dst, proto, src)
- Senders have no templates or flow specifications
  - Senders provide a bandwidth parameter
- Receivers have no filter or flow specifications
  - Receivers request only service and style
- Simplied error model and no callback functions

The Simplified Error Model

- Tri-valued: Red, Yellow, Green
- Yellow - Operation is pending
- Green (CONFIRM, RESV) - Advance to Go, collect $200
- Red (PATH_ERROR, RESV_ERROR) - Go directly to jail, do not pass Go

API Description

- Reservations
  - scrapi_sender(), scrapi_receiver(), scrapi_close()
- Error handling
  - scrapi_get_status()
  - scrapi_errno(), scrapi_perror(), scrapi_errlist()
  - scrapi_stderr(), scrapi_debug()
- Asynchronous event loop
  - scrapi_poll_list(), scrapi_dispatch()
- Address manipulation (IPv4 and IPv6)

Applications

- Test tool - scrapit (similar to rtap)
- Performance analysis - netperf (plus some useful scripts)
- Multimedia - vat, vat6, vic (quasi generic Tcl/Tk SCRAPI interface)

Open Issues

- Simplified error model
  - Good enough?
  - Should we try harder (Reliable CONFIRMS)?
- Computing a flow specification
  - Peak rate = 2 * average rate
  - Bucket depth = 2 * average rate (2 seconds)
  - Minimum policed unit = 64
  - Maximum packet size = Max MTU of any IP interface
  - Rspec rate = average rate
  - Slack term = 0