BSMP - Broadcast Satellite Multimedia Protocol

Andreas ROTTMANN

arott@cosy.sbg.ac.at

Praxis-Project absolved at

University of Salzburg, Austria

Department of Computer Science

Supervision: Hilmar Linder, hlinder@cosy.sbg.ac.at

OVERVIEW

① Multicast Introduction

② Reliable Multicast

③ BSMP Overview

④ Design & Implementation

⑤ Tools

MULTICAST INTRODUCTION

Why multicast:

- ➜ When sending same data to multiple receivers
- ➜ Save bandwidth over multiple unicast connections
- ➜ Receivers' addresses unknown

Multicast Applications:

- ➜ Audio/Video conferencing and "broadcasts"
- ➜ News distribution
- ➜ Software updates (e.g. clusters, workstation sets)
- ➜ Resource discovery/advertisement
- ➜ . . .

IP Multicast:

- ➜ RFC 1112 (Host Extensions for IP Multicast)
- ➜ Multicast groups, having IPv4 class D or IPv6 multicast addresses
- ➜ Members of groups may be located anywhere on the Internet
- ➜ Members join and leave groups and tell the routers via IGMP (RFC 3376)
- ➜ Senders do not need to be member of the multicast group
- ➜ Routers use multicast routing protocols to manage groups
- ➜ *IP-Multicast is best-effort (unreliable)*

Reliable Multicast

Problems:

- ➜ ACK-based schemes (like used in TCP) don’t scale
- ➜ Request explosion
- ➜ Reply explosion

Solutions:

- ➜ NAK-based scheme (receiver estimates a timeout and sends a NAK after that)
- ➜ NAK-filtering
- ➜ FEC encoding

BSMP Overview

What it is:

- ➜ Multicast Protocol, built upon UDP/IP Multicast
- ➜ Designed for the requirements of satellite transmissions
  - Long RTT
  - Asymmetric communication (receiver feedback via wired Internet)
- ➜ Offers different levels of reliability
- ➜ Implemented as a shared library offering an C API similar to POSIX sockets
- ➜ Successor of RRMP (Restricted Reliable Multicast Protocol), written by Hilmar Linder and Klaus Siegesleitner
- ➜ Complete re-write and re-design
  - Essentially the same features
  - Less than half of the core code size

Protocol Key Points:

- ➜ Built on UDP
- ➜ Preserves message boundaries
- ➜ Groups messages into *transmission groups* (TGs)
- ➜ TGs have a sequence number
- ➜ NAK-based retransmission scheme
- ➜ Retransmissions are FECs over a TG
- ➜ Per-receiver RTT estimation

Features:

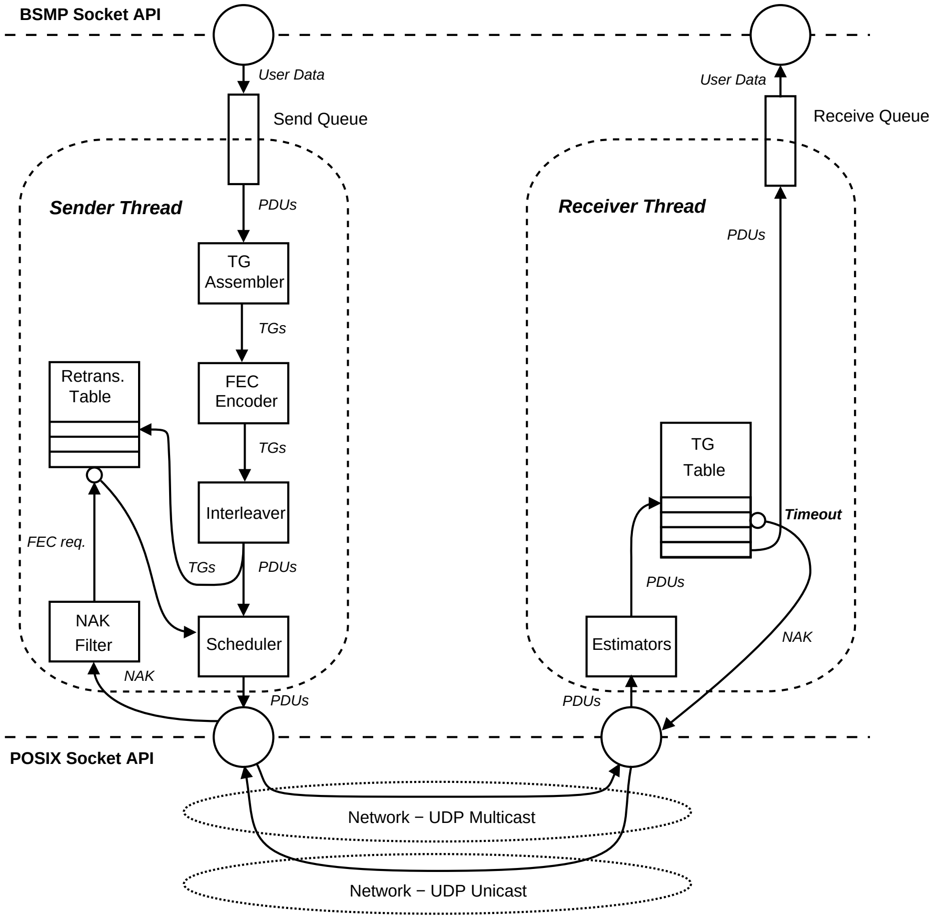
- ➜ Four modes (service classes)
  - Full reliability
  - Proactive
  - Limited reliability
  - Unreliable
- ➜ Designed for asymmetric communication
- ➜ Data-rate control
- ➜ Interleaving (protection against burst losses)
- ➜ Two NAK suppression algorithms (default, LE-SBCC)
- ➜ Portable (POSIX, Windows using WSA)

BSMP Socket API (simplified):

- ➜ `bsmp_socket(family, style)`
- ➜ `bsmp_join(sock, if_addr, mc_addr)`
- ➜ `bsmp_leave(sock, if_addr)`
- ➜ `bsmp_connect(sock, if_addr, mc_addr)`
- ➜ `bsmp_send(sock, data),`
- ➜ `bsmp_recv(sock, data)`
- ➜ `bsmp_close(sock)`

Design & Implementation

Overview:



Code Structure:

- ➜ Data structures (object-oriented C)
  - Timer queue
  - Data rate estimator
  - Socket address routines
  - PDU
  - Transmission group
  - Send queue, receive queue
  - Transmission group table
  - Virtual socket layer
- ➜ Sender logic
- ➜ Receiver logic

Timer queue:

- ➜ Core component, used for all timeouts
- ➜ Sender thread and receiver thread run in a loop, with an iteration at least about every 10 msecs
- ➜ There may be many outstanding timeouts (each TG has a timeout)
- ➜ Once each iteration, there is a timer queue “clock-tick”
- ➜ Clock-ticks *do not* need to be uniformly spaced
- ➜ All timers that have already expired at the clock-tick are invoked and removed from the queue
- ➜ Efficient implementation: timer start and remove are both $O(\log n)$ (balanced binary tree)

Virtual Socket Layer:

- ➜ Abstracts the operations needed on an underlying socket (normally POSIX or WSA)
- ➜ Used to implement sockets that communicate via thread-safe queues (in-process)
- ➜ Primarily for testing

Software and APIs used:

- ➜ GLib 2.0
- ➜ POSIX threads
- ➜ POSIX socket API (WSA on Windows)
- ➜ ISO C, written for POSIX APIs

TOOLS

Sample sender and receiver:

➜ Simple, general-purpose BSMP sender/receiver

➜ Allows access to all protocol options

Test Torture:

➜ Test program

➜ Uses virtual socket layer

➜ Generates randomly-sized packets with random content

➜ Checks if they are correctly received

➜ Can be configured via an XML config file:

- Link properties (delay, loss rate)
- Protocol options (mode, data rate, . . .)

THE END

Thanks for Your Attention!