PGM: Reliable General Multicast Implementation for Linux?

Christoph Lameter,
christoph@lameter.com

Overview

- Why “reliable” multicasting.
- “Middleware” for signaling
- Existing implementations
- The PGM standard in brief.
- A proposed sockets based implementation
- Kernel implementation challenges
- Future outlook

Why “reliable” Multicasting

- Need to signal events in the fastest way to a large group of changing recipients.
- TCP
  - Is slow and reliable
  - Consumes lots of resources
  - Can only reach one destination
- UDP multicast
  - Fast
  - Reaches unlimited number of destinations
  - But unreliable

Middleware

- Multicast data streams
  - IGMP
  - Pub/Sub with network hardware support.
- Sequence numbers
- Retransmission
- Protocols
  - IP based PGM native version
  - UDP encapsulated PGM

Existing Implementations

- **Proprietary**
- Tibco
- 29West
- IBM Websphere
- Wombat MAMA (not PGM)
- Microsoft Windows PGM (MS-PGM)

- **Open Source**
- Openpgm
- AMQP (not PGM)
- 0MQ (zeromq)

PGM standard (RFC 3208)

- Multicast packets with sequence ids.
- Receiver check sequence id.
- NAK on missing packets. Sender sends repair traffic.
- Heartbeat.
- Network element assist.
- Scalability issues.
- FEC capabilities.
- NCF
- ODATA
- RDATA
- Receive / Send Window

Hardware support for Native PGM

- Network “Elements”
- NAK suppression
- Optimize Repair traffic
- Exploit FEC.

- Cisco Switches
- Juniper Switches
- Nortel Networks Switches

A Linux Sockets Based Implementation

- Use socket like an UDP socket
- IPPROTO_PGM
- SOCK_RDM
- Reverse connection over UDP
  - Sender is not handshaking
  - Receiver waits for packet and then accepts.
- Based on Miru's openpgm code
- API Challenge since Miru uses library interface.
- MS-PGM interface suffered bit rot.
- Thus new API

Why in the kernel?

- Framework for creating new protocols exists in the kernel
- RAW sockets used to implement native PGM cause various issues.
- Middleware uses PGM over UDP and thus cannot use network elements for NAK suppression.
- Potential of high speed processing since we have direct access to the NIC and the queues.
- Natural priority over everything else.
- Able to detect dysfunctional network behavior.

What exists

- Documentation
  - Manpage
  - Basic howto
  - Variable structures
  - Integration of SOCK_RDM and IPPROTO_PGM
  - Openpgm implementation.
- Code
- Sample code
- Basic pgm skeleton partially completed.
- Nothing that is runnable yet. Keep getting distracted by other projects. :#

Fundamental Challenges

- Kernel socket API competes with bare metal implementations.
- Need alternative to message copying via send()/receive().
- Issues with Offload technology having a hard time getting acceptance.
- Receive/Memory usage in kernel for large receive windows
- Third party interactions
  - MS-PGM
  - Tibco
  - IBM Websphere
  - AMQP (not PGM?)

Future

- Having a kernel implementation would simplify a lot of things.
- Minimal implementations so that user space frameworks can be built on top.
- Need to address publish → large subscriber issues.
- Manpage / Definition
- Questions and Answers.