Dr. Multicast

Rx for Data Center Communication Scalability

LADIS, September 15, 2008

Ymir Vigfusson  Hussam Abu-Libdeh  Mahesh Balakrishnan  Ken Birman
Cornell University
Yoav Tock
IBM Research Haifa
IP Multicast in Data Centers

- IPMC is *not used* in data centers
IP Multicast in Data Centers

- Why is IP multicast rarely used?
Why is IP multicast rarely used?
- Limited IPMC scalability on switches/routers and NICs
IP Multicast in Data Centers

- Why is IP multicast rarely used?
  - Limited IPMC scalability on switches/routers and NICs
  - Broadcast storms: Loss triggers a horde of NACKs, which triggers more loss, etc.
  - Disruptive even to non-IPMC applications.
IP Multicast in Data Centers

- IP multicast has a bad reputation
IP Multicast in Data Centers

- IP multicast has a bad reputation
  - Works great up to a point, after which it breaks catastrophically.
IP Multicast in Data Centers

- Bottom line:
  - *Administrators have no control over multicast use ...*
  - *Without control, they opt for never.*
Policy: Permits data center operators to selectively enable and control IPMC

Transparency: Standard IPMC interface, system calls are overloaded.

Performance: Uses IPMC when possible, otherwise point-to-point UDP

Robustness: Distributed, fault-tolerant service
Terminology

- **Process**: Application that joins logical IPMC groups
- **Logical IPMC group**: A virtualized abstraction
- **Physical IPMC group**: As usual
- **UDP multi-send**: New kernel-level system-call

- **Collection**: Set of logical IPMC groups with identical membership
Acceptable Use Policy

- Assume a higher-level network management tool compiles policy into primitives
- Explicitly allow a process to use IPMC groups
  - *allow-join*(process, logical IPMC)
  - *allow-send*(process, logical IPMC)
- UDP multi-send always permitted
- Additional restraints
  - *max-groups*(process, limit)
  - *force-udp*(process, logical IPMC)
Overview

- Library module
- Mapping module
- Gossip layer
- Optimization questions
- Results
**MCMD Library Module**

- **Transparent.** Overloads the IPMC functions
  - `setsockopt()`, `send()`, etc.

- **Translation.** Logical IPMC map to a set of P-IPMC/unicast addresses.
  - Two extremes
MCMD Mapping Role

- MCMD Agent runs on each machine
  - Contacted by the library modules
  - Provides a mapping

- One agent elected to be a *leader*:
  - Allocates IPMC resources according to the current policy
Allocating IPMC resources: An optimization problem
MCMD Gossip Layer

- Runs system-wide
- Automatic failure detection
- Group membership fully replicated via gossip
  - Node reports its own state
  - *Future:* Replicate more selectively
  - Leader runs optimization algorithm on data and reports the mapping
MCMD Gossip Layer

- But gossip is slow...

- Implications:
  - Slow propagation of group membership
  - Slow propagation of new maps
  - *We assume a low rate of membership churn*

- Remedy: *Broadcast module*
  - Leader broadcasts urgent messages
  - Bounded bandwidth of urgent channel
  - Trade-off between latency and scalability
Overview

- Library module
- Mapping module
- Gossip layer

- Optimization questions

- Results
Optimization Questions

- First step: compress logical IPMC groups
Optimization Questions

- How compressible are subscriptions?
  - Multi-objective optimization:
    - Minimize number of collections
    - Minimize bandwidth overhead on network

- Ties in with social preferences
  - How do people's subscriptions overlap?
Optimization Questions

- How compressible are subscriptions?
  - Multi-objective optimization:
    - Minimize number of groups
    - Minimize bandwidth overhead on network
  - **Thm**: The general problem is NP-complete
  - **Thm**: In uniform random allocation, "little" compression opportunity.
  - Replication (e.g. for load balancing) can generate duplicates (easy case).
Optimization Questions

- Which collections get an IPMC address?
  - **Thm**: Ordered by decreasing traffic * size, assign P-IPMC addresses greedily, we minimize bandwidth.

- **Tiling** heuristic:
  - Sort L-IPMC by traffic * size
  - Greedily collapse identical groups
  - Assign IPMC to collections in reverse order of traffic * size, UDP-multisend to the rest

- Building tilings incrementally
- Insignificant overhead when mapping L-IPMC to P-IPMC.
Overhead

- Linux kernel module increases UDP-multisend throughput by 17% (compared to user-space UDP-multisend)
Policy control

- A malfunctioning node bombards an existing IPMC group.
- MCMD policy prevents ill-effects
Policy control

- A malfunctioning node bombards an existing IPMC group.
- MCMD policy prevents ill-effects
Network Overhead

- MCMD Gossip Layer uses constant background bandwidth

- Latency of leaves/joins/new tilings bounded by gossip dissemination latency
Conclusion

- IPMC has been a bad citizen...
Conclusion

- IPMC has been a bad citizen...
- *Dr. Multicast* has the cure!
- Opportunity for big performance enhancements and policy control.
Thank you!