A Patchwork of Multicast Regions

Qi Huang  Huazhong University of Science and Technology

Ýmir Vigfússon  IBM Research Haifa Labs

Ken Birman  Cornell University

Haoyuan Li  Cornell University
Pub/sub transport

- **Pub/sub over WAN has plethora of modern uses**
  - Facebook and Twitter real-time feeds
  - Web and cloud management
  - Massive multiplayer online games (MMOGs)
  - Component of numerous DEBS applications

- **What about multicast over WAN?**
  - *Multicast*: One-to-many message dissemination
  - A natural transport mechanism for pub/sub
  - **Wishlist:**
    i. Minimize redundant traffic
    ii. Minimize average latency of delivery yet with high throughput
    iii. Limit per-node storage requirements
    iv. Stay robust to node churn/failures
    v. Automatically adapt to the runtime environment
Status of Multicast

- **IP-multicast (IPMC)**
  - Disabled over WAN links
    - Security concerns (DDoS attacks)
    - Economic issues (how do you charge for IPMC?)
  - Enabled in many data centers
    - Possible to fix scalability and reliability issues

- **Application-level multicast (ALM)**
  - Iterated unicast does not scale ⇒ Use an overlay
  - Dissemination overlays ignore underlying topology and IPMC
  - Peer-to-peer structures usually vulnerable to churn
  - Mesh solutions have high overhead and increase latency

- **No known solution achieves all of our goals**
  - Can one size fit all?
Introducing Quilt

**Idea:** What if we *combine* multiple multicast solutions?

- **Quilt** weaves multicast *regions* into a patchwork:
  - Discovers context automatically
  - Optimizes efficiency
  - Exports a simple library interface for developers
  - Routes messages between regions
  - Allows administrators to impose policy (e.g. enable IPMC)
Quilt Overview

- Quilt exposes a simple multicast API to end-user applications
- The **multicast container** stores active protocol “objects”
Multicast Protocols

- Quilt prototype supports three multicast protocols
  - **IPMC**
    - Network-level IP multicast
  - **DONet (CoolStreaming)**
    - Mesh-structured multicast
    - BitTorrent-style content dissemination
  - **OMNI**
    - Tree-based latency-aware multicast
    - Optimizes average latency from source without burdening internal nodes
    - Quilt uses OMNI for global patch multicast
Quilt Overview

- Quilt exposes a simple multicast API to end-user applications
- The multicast container stores active protocol “objects”
- The detection service discovers environment properties
  - NATchecker, traceroute, latency + bandwidth statistics
  - Constructs environment identifier (EUID)
### Environment Identifier

| Connectivity Options: A list of tuples (D: Direction, T: Protocol) |
|-----------------|-----------------|
| **Direction**   | **Protocol**    |
| Bidirectional / Unidirectional | Transport protocols such as TCP or UDP |

<table>
<thead>
<tr>
<th>Local Topology: Routing path from an end host to the Internet WAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Router Stack</strong></td>
</tr>
<tr>
<td>A path consisting of routers</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

| Measured Performance: A list of triples (Type, Lower bound, Upper bound) |
|-----------------|-----------------|
| **Bandwidth**   | **Latency**    |
| Bandwidth range | Latency range   |
| **Others**      | **Ranges**     |

- For each NIC, what transport protocols are supported and in which direction?
- 74% of hosts run behind NAT boxes and firewalls
  - These hosts might be limited to a leaf-role
Environment Identifier

<table>
<thead>
<tr>
<th>Connectivity Options: A list of tuples (D: Direction, T: Protocol)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direction</strong></td>
</tr>
<tr>
<td>Bidirectional / Unidirectional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Local Topology: Routing path from an end host to the Internet WAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Router Stack</strong></td>
</tr>
<tr>
<td>A path consisting of routers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measured Performance: A list of triples (Type, Lower bound, Upper bound)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bandwidth</strong></td>
</tr>
<tr>
<td>Bandwidth range</td>
</tr>
</tbody>
</table>

- Trace the routing path to the local DNS server
- If two hosts share a DNS server *or* an intermediate router within 5 hops, then belong to the same AS with 85% probability
- Check for IPMC capabilities as well
### Environment Identifier

<table>
<thead>
<tr>
<th>EUID</th>
<th>Connectivity Options: A list of tuples (D: Direction, T: Protocol)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Direction</strong></td>
</tr>
<tr>
<td></td>
<td>Bidirectional / Unidirectional</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>Local Topology</strong>: Routing path from an end host to the Internet WAN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Router Stack</strong></td>
</tr>
<tr>
<td></td>
<td>A path consisting of routers</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th><strong>Measured Performance</strong>: A list of triples (Type, Lower bound, Upper bound)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Bandwidth</strong></td>
</tr>
<tr>
<td></td>
<td>Bandwidth range</td>
</tr>
</tbody>
</table>

- Periodically estimate network performance
- Fluctuates over time, so measurements encoded as *ranges* of values
Patch formation

- **Bootstrap sequence**
  - New host generates EUID for each NIC
  - Sends EUID to bootstrap server
  - Receives EUIDs for initial contacts in compatible regions

- **Rules**
  - Patches defined by EUIDs based on ALM rules
  - E.g. if IPMC enabled router is shared between $E_1$ and $E_2$, a region is formed with EUID $E_{1,2}$
  - Members eventually get a single maximal EUID

- **Global patch**
  - A wide-area overlay connects other regions
  - Overlaps in some **representative** node for each patch
Bootstrap server

- **Three main roles**
  1. Maintain partial membership for each patch
  2. Structure nodes into patches
  3. Ensure health of global patch

- **Off the critical path**
  - Only used on joins and when nodes become isolated

- **Gossip version of Quilt**
  - Distributed maintenance of membership.
  - Alleviates single-point-of-failure concerns
Churn resilience

- Each patch has a *representative* node
  - Tunnels traffic between the patch and the global patch
- What about churn?
  - The quilt overlay may partition if the representative leaves
- For robustness, Quilt has $k$ representatives in each patch
  - Here $k$ is a small number, like 2 or 3
  - Increasing $k$ in turn increases message duplication
- Quilt is able to recover after failure
  - Hosts periodically report to the bootstrap server for a fresh membership snapshot
  - Representatives are monitored and new ones appointed if they die
Duplication suppression

- More representatives $\Rightarrow$ more duplicates

- Suppressing duplicates *per host*
  - Each host maintains a Bloom filter, and marks incoming messages
  - Reset filter periodically depending on multicast data rate

- Suppressing duplicates *among representatives*
  - Gossip-based protocol links patch representatives within multicast region
  - Since $k$ is tiny, use simple 2-phase synchronization protocol
Motivation

Quilt Overview

- Environment Identifier (EUID)

Quilt Architecture

- Bootstrap server
- Churn resilience
- Duplication suppression

Evaluation

- Data Center Topology
- Internet Topology

Conclusion
Data Center Topology

- **Application:** Publish/subscribe event notification among data centers linked on Internet WAN.
- Grid5000 structure links 25 clusters (1531 servers) from 9 locations in France
- Sub-millisecond latencies within clusters, 4-6ms between sites
- **Assumptions:**
  - Single-source multicast
  - IPMC is enabled within each site, but not between them
Data Center Topology

- **Overhead:** Quilt has substantially less network overhead than pure OMNI for constructing the overlay.
  - The Quilt overlay trees are much smaller.
Latency: Quilt leverages IPMC to accelerate event dissemination unlike the system-wide OMNI tree.
Data Center Topology

- **Churn resilience:** Quilt recovers quickly even from catastrophic failures (50% of nodes randomly die), or within 10 seconds.

![Graph showing average throughput over time with different Quilt configurations.](image-url)
Data Center Topology

- **Duplication suppression**: Bloom-filters and 2-phase synchronization among representatives are effective.

50% of nodes die
Internet Topology

**Application:** Internet-wide dissemination
- A synthetic collection of 951 Internet hosts
- End-to-end latency between hosts from PeerWise
- Selected random hosts from CAIDA traceroute records with similar host-to-host latencies and gathered route information

**Assumptions:**
- Single-source multicast
- IPMC is not enabled
- Quilt uses DONet within patches and OMNI between them
Internet Topology

- **Latency:** Quilt disseminates data quicker than DONet by avoiding expensive BitTorrent-style scheduling cycles.
Internet Topology

- **Overhead:** Quilt avoids the BitTorrent-style overheads experienced by DONet.
Internet Topology

- **Churn resilience**: Many small regions makes recovery harder, but still relatively quick.

- **Duplication suppression**: 50% fewer duplicates when $k=2$, 63% lower when $k=3$. 

![Graph showing average throughput over time with different quilt k values]
Conclusion

- Impractical to create a one-size-fits-all multicast solution
  - Many internet nodes separated by WAN links
  - Small clusters of nodes residing behind NAT or firewall
  - Larger clusters in settings where IPMC is available
- Quilt weaves a patchwork of multicast regions
  - Automatic environment detection and region formation
  - Resilient to churn and suppresses duplicates
  - Lightweight and scalable multicast service
- Quilt actually works!
  - Free download at Cornell’s Live Distributed Objects site:
    - http://liveobjects.cs.cornell.edu