Group Unicast for the Real World

Elad Lahav, Tim Brecht, Martin Karsten, Weihan Wang and Tony Zhao

David R. Cheriton School of Computer Science

May 29, 2008
Goal

Transmit identical data to multiple recipients

Applications Live broadcasting, games, VoIP conferencing, ...
Solution 1: Multiple Unicast Transmissions

sendmsg("Hello Group", A)
sendmsg("Hello Group", B)
sendmsg("Hello Group", C)

Problem Wastes both local and global resources
Solution 2: Multicast

[Deering 1990]

sendmsg("Hello Group", G)

Hello Group

Problem Lack of Internet support, costly for small groups
Solution 3: Group Unicast

[Karsten, Song, Kwok, Brecht 2005]

sendmsg("Hello Group", {A,B,C})

Benefits Reduces mode switches and memory copies in server
Important UDP only
Contributions

- Improved API and implementation
- Integration with a real-world media server
- Precise performance analysis
Interface Changes

Old Interface

- Group associated with UDP socket
- Use `send()` to transmit to group
- Use `setsockopt()` to manage group

New Interface

- New system call: `sendgroup()`
- No need for extra system calls to manage group
- Per-recipient private data
int sendgroup(int sd, struct giobuf* buf, size_t recnum, int flags, int* gerrno);

struct giobuf {
    struct iovec shared;       /* Shared data buffer */
    struct giovec recinfo[1];  /* Per-recipient info. */
};

struct giovec {
    struct sockaddr_in giov_dest; /* Destination address */
    struct iovec giov_prepend;   /* Prepended buffer */
    struct iovec giov_append;    /* Appended buffer */
};
Implementation

BSD: mbuf chain

Linux: sk_buff + page pointers
Implementation

Without Scatter-Gather I/O

With Scatter-Gather I/O
Micro-Benchmark: Improvement/Group Size

![Graph showing improvement vs. group size for Linux, Solaris, and FreeBSD systems.](image-url)
Micro-Benchmark: Packet Send-Time/Packet Size

![Graph showing send time (micro-secs) vs payload size (bytes) for Sendmsg and Sendgroup methods. The graph indicates that Sendmsg has a higher send time compared to Sendgroup as payload size increases.]
Micro-Benchmark: Packet Per-Recipient Data

![Graph showing send time in microseconds against percent of private data in payload for Sendmsg and Sendgroup.]
Show that `sendgroup()`:

- Is applicable
- Can be integrated
- Improves performance
The Helix Server

- Multimedia server from Real Networks
- Open source version of the Real Server
- Handles both on-demand and live content
- [https://helix-server.helixcommunity.org](https://helix-server.helixcommunity.org)
Helix Live-Broadcasting

Producer -> Server

Server -> Clients A
Server -> Clients B
Server -> Clients C

RTP/UDP
**Helix-sendgroup() Integration**

**Original Helix**

```
recv()

send() send() send()
```

**Modified Helix**

```
recv()

foreach client {
    send()
}
```

**Diagram:***

- **Original Helix**:
  - `recv()`
  - `send()`
  - `send()`
  - `send()`

- **Modified Helix**:
  - `recv()`
  - `foreach client {
        send()
    }
  ```
Helix Benchmark: CPU Utilisation/Group Size

CPU Utilisation (%) vs Group Size

- Sendmsg
- Sendgroup
Helix Benchmark: Client Rate/Group Size

Average Client Rate (Kbps) vs. Group Size

- Sendmsg
- Sendgroup

Group Size: 1000 to 6000
Performance Analysis

- **Micro-benchmarks:** Determine speed-up of `sendgroup()` over `sendmsg()` loop

- **Helix:** Measure execution time of `sendmsg()` loop

- **Amdahl’s law:** Predict overall Helix improvement

- **Confirmation:** Compare to observable overall Helix improvement
Amdahl’s Law in Action

- Group size: 1000
- Payload size: 1000 bytes
- `sendgroup()` speed-up: $s = 1.664$ (from micro-benchmarks)
- Fraction: $f = \frac{T_{sc} + T_{irq} + T_{bh} - T_{sleep}}{T_{exp}} = 0.791$
- Expected overall speed-up:

\[
r = \frac{1}{1 - f + \frac{f}{s}} = \frac{1}{1 - 0.791 + \frac{0.791}{1.664}} = 1.461
\]

- Observed speed-up: $\bar{r} = 1.45$
Conclusions

- `sendgroup()` has real-world applications
- Noticeable performance improvement in certain scenarios
- Avoiding mode switches is good
- Avoiding memory copies even better
  - Direct effect on system call execution path
  - Also beneficial to entire environment
- Better analysis of system calls $\Rightarrow$ more accurate predictions