Reliable high-speed Grid data delivery using IP Multicast

Karl Jeacle & Jon Crowcroft
firstname.lastname@cl.cam.ac.uk
Rationale

- Would like to achieve high-speed bulk data delivery to multiple sites
- Multicasting would make sense
- Existing multicast research has focused on sending to a large number of receivers
- But Grid is an applied example where sending to a moderate number of receivers would be extremely beneficial
Multicast availability

- Deployment is a problem!
  - Protocols have been defined and implemented
  - Valid concerns about scalability; much FUD
  - “chicken & egg” means limited coverage

- Clouds of native multicast
  - But can’t reach all destinations via multicast
  - So applications abandon in favour of unicast

  - What if we could multicast when possible…
  - …but fall back to unicast when necessary?
Multicast TCP?

- TCP
  - “single reliable stream between two hosts”

- Multicast TCP
  - “multiple reliable streams from one to \( n \) hosts”

- May seem a little odd, but there is precedent…
  - SCE – Talpade & Ammar
  - TCP-XMO – Liang & Cheriton
  - M/TCP – Visoottiviseth et al
ACK implosion
Building Multicast TCP

- Want to test multicast/unicast TCP approach
  - But new protocol == kernel change
  - Widespread test deployment difficult

- Build new TCP-like engine
  - Encapsulate packets in UDP
  - Run in userspace
  - Performance is sacrificed…
  - …but widespread testing now possible
TCP/IP/UDP/IP

Sending Application

TCP
IP
UDP
IP

Receiving Application

TCP
IP
UDP
IP

If natively implemented

test deployment
TCP engine

- Where does initial TCP come from?
- Could use BSD or Linux
  - Extracting from kernel could be problematic
- More compact alternative
  - lwIP = Lightweight IP
  - Small but fully RFC-compliant TCP/IP stack

- lwIP + multicast extensions = “TCP-XM”
TCP
TCP-XM connection

- Connection
  - User connects to multiple unicast destinations
  - Multiple TCP PCBs created
  - Independent 3-way handshakes take place
  - SSM or random ASM group address allocated
    - (if not specified in advance by user/application)
  - Group address sent as TCP option
  - Ability to multicast depends on TCP option
TCP-XM transmission

Data transfer
- Data replicated/enqueued on all send queues
- PCB variables dictate transmission mode
- Data packets are multicast
- Retransmissions are unicast
- Auto switch to unicast after failed multicast

Close
- Connections closed as per unicast TCP
TCP-XM reception

- Receiver
  - No API-level changes
  - Normal TCP listen
  - Auto-IGMP join on TCP-XM connect
  - Accepts data on both unicast/multicast ports
  - `tcp_input()` accepts:
    - packets addressed to existing unicast destination…
    - …but now also those addressed to multicast group
What happens to the $cwin$?

- Multiple receivers
  - Multiple PCBs
  - Multiple congestion windows

- Default to $\text{min}(cwin)$
  - i.e. send at rate of slowest receiver

- Avoid by splitting receivers into groups
  - Group according to $cwin$ / data rate
API changes

- Only relevant if natively implemented!

- Sender API changes
  - New connection type
  - Connect to port on array of destinations
  - Single write sends data to all hosts

- TCP-XM in use:
  
  ```c
  conn = netconn_new(NETCONN_TCPXM);
  netconn_connectxm(conn, remotedest, numdests, group, port);
  netconn_write(conn, data, len, ...);
  ```
PCB changes

- Every TCP connection has an associated Protocol Control Block (PCB)

- TCP-XM adds:

  ```c
  struct tcp_pcb {
      ...
      struct ip_addr group_ip; // group addr
      enum tx_mode txmode; // uni/multicast
      u8t nrtxm; // # retrans
      struct tcp_pcb *nextm; // next “m” pcb
  }
  ```
Linking PCBs

- *next* points to the next TCP session
- *nextm* points to the next TCP session that’s part of a particular TCP-XM connection
- Minimal timer and state machine changes
TCP Group Option

- New group option sent in all TCP-XM SYN packets
- Presence implies multicast capability
- Non TCP-XM hosts will ignore (no option in SYNACK)
- Sender will automatically revert to unicast

<table>
<thead>
<tr>
<th>kind=50</th>
<th>len=6</th>
<th>Multicast Group Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td>1 byte</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>
Initial tests: speed
Initial tests: efficiency
Future

To do:

- Protocol work
  - Parallel unicast / multicast transmission
  - Fallback / fall forward
  - Multicast look ahead
  - Multiple groups
  - Globus integration

Experiments

- Local network
- UK eScience centres
- Intel PlanetLab
eScience volunteers…

1. In place
   - Cambridge
   - Imperial
   - UCL

2. Nearly
   - Oxford
   - Newcastle
   - Southampton

3. Not quite
   - Belfast
   - Cardiff
   - Daresbury
   - Rutherford

4. Not at all
   - Edinburgh
   - Glasgow
   - Manchester
All done!

- Thanks for listening!
- Questions?