REBOOK: a Network Resource Booking Algorithm

draft-montessoro-rebook-00

Pier Luca Montessoro, Riccardo Bernardini
montessoro@uniud.it, riccardo.bernardini@uniud.it

Multimedia Networking and Applications lab
DIEGM - University of Udine, Italy
The research group
(a multidisciplinary approach)

- Pier Luca Montessoro, coordinator, full professor in computer science (networking and software development)
- Franco Blanchini, full professor in controls (distributed control functions)
- Mirko Loghi, assistant professor in computer science (networking, hardware and software development)
- Riccardo Bernardini, assistant professor in telecommunications (multimedia encoding and networking)
- Daniele Casagrande, assistant professor in controls (distributed control functions)
- Stefan Wieser, research assistant in computer science (networking and software development)
Our possible contribution to ICN

- ICN can benefit from congestion- and flow-controlled transport of objects from a given location to the interested receiver

- REBOOK provides deterministic, dynamic and scalable resource reservation
  - maximum delivery time for generic NDOs
  - adequate transport performance for multimedia streaming services

- REBOOK can be useful for some instances of ICN
- (We are looking for feedbacks!)
REBOOK

- **IS NOT** another reservation protocol
- **IS** a distributed algorithm for efficient status information handling within intermediate nodes
- provides an *open framework* for congestion avoidance/control, fast packet forwarding and other features
- can be applied to existing or new protocols
- provides interaction and feedbacks between the network and the hosts/applications
- provides circuit performance for packet forwarding, for free
- high degree of flexibility (IPv4, IPv6, multicast)
REBOOK and ICN

- **REBOOK: new paradigm**
  - routers, senders and receivers cooperate and handle per-flow state information

- **ICN: new architecture**
  - routers, senders and receivers are merged
    - cooperation becomes natural
    - they can trust each other
  - REBOOK can be useful to improve the transport services for ICN based on packet switching

- **Deployment**
  - REBOOK is designed for incremental deployment
  - it works even along partially rebook-aware routes
  - we guess ICN represents an ideal environment for its implementation and deployment
REBOOK and ICN

name resolution, caching, ...

object

ICN

REBOOK

IP forwarding infrastructure

routing, forwarding, ...

IP forwarding infrastructure
The Question

“Routers cannot keep state information for each connection (flow) traversing a node. It does not scale”.

- In practical applications, is it still true with today’s technology?
A tale of space and time...

Available memory

Computation time
Space

In 4 GB of memory:

~86 millions of flow information @ 50 bytes per flow

86 millions of flows means:

~688 Gbps @ 8 kbps per flow
~33 Tbps @ 384 kbps per flow

Not an issue for the control plane of ICN nodes routing modules
Time: here comes REBOOK

The enabling algorithm:
DLDS (Distributed Linked Data Structure)

**During setup**
- store resource reservation information in routers
  AND
- keep track of pointers (memory addresses or indexes in tables) along the path

**Afterwards**
- use the pointers to access status information **without searching**
Resource reservation and pointers collection

N1 (sender)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>req=2, res=2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Resource Reservation Table

2 Mb/s

Request

N3

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Resource reservation ACK message

req=2, res=1

Resource Reservation Table

1 Mb/s

Request

N4 (receiver)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Routing

Cache

Application

ICN

Routing

Cache

Application

N2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Routing

Cache

Application

ICN

Routing

Cache

Application

N3

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>req=2, res=1</td>
</tr>
</tbody>
</table>

Routing

Cache
Fast packet forwarding
A few problems

- route changes, disappearing flows, end nodes or routers faults
  - high speed consistency check
  - highly efficient, low priority table cleanup process

- need to dynamically change assigned resource amounts
  - partial release
  - distributed control function for optimality and fairness
Does it work?

10 UDP “flows”, $R_{min}=15$ $R_{req}=25$

this link is down between $T_1$ and $T_2$

total packet rate per sender

number of booked flows per sender node

$T_1$: route change

$T_2$: route change
Does it work? (cont’d)

optimal and fair!
“... and running code”

- **Current prototype**
  - Extremely lightweight hosting protocol
  - Add-on modules for applications and routing engines
  - C/C++ static or dynamic link library
  - Multi-platform (Linux gcc, Microsoft Visual Studio)

<table>
<thead>
<tr>
<th>Module</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Router</td>
<td>30 KB</td>
</tr>
<tr>
<td>Sender</td>
<td>20 KB</td>
</tr>
<tr>
<td>Receiver</td>
<td>8 KB</td>
</tr>
</tbody>
</table>

- **Under development:**
  - Embedding in Linux kernel
  - Usage of unassigned IP Option Alert flag values
Prototype

**ROUTER**
- handle REBOOK message

**REBOOK ENGINE**
- get currently available resource
- notify available resource increase
- notify available resource reduction
- send rebook message

**SENDER**
- reservation request
- reservation upgrade request
- reservation removal request
- handle rebook message

**REBOOK ENGINE**
- notify reservation ACK
- notify reduction ACK
- notify reset
- send rebook message

**RECEIVER**
- handle rebook message
- partial reservation release request

**REBOOK ENGINE**
- notify reservation event
- send rebook message
## Performance

### CPU times (DLDS and resource reservation management)

<table>
<thead>
<tr>
<th>Activity</th>
<th>configuration</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup (incl. res. reserv.)</td>
<td>10,000 flows</td>
<td>200 ns once per flow</td>
</tr>
<tr>
<td>setup (incl. res. reserv.)</td>
<td>10,000,000 flows</td>
<td>250 ns once per flow</td>
</tr>
<tr>
<td>Keepalive message handling</td>
<td>10,000 flows</td>
<td>100 ns every 5 seconds</td>
</tr>
<tr>
<td>Keepalive message handling</td>
<td>10,000,000 flows</td>
<td>190 ns every 5 seconds</td>
</tr>
<tr>
<td>RR table entries release</td>
<td>10,000 flows</td>
<td>25 ns per flow</td>
</tr>
<tr>
<td>RR table entries release</td>
<td>10,000,000 flows</td>
<td>48 ns per flow</td>
</tr>
<tr>
<td>RR table cleanup</td>
<td>10,000,000 entries</td>
<td>100 ms every 15 seconds</td>
</tr>
</tbody>
</table>

### CPU times (direct access forwarding, including consistency check)

<table>
<thead>
<tr>
<th>Activity</th>
<th>configuration</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLDS forwarding table access</td>
<td>1,000,000 routes</td>
<td>10.57 ns per packet</td>
</tr>
<tr>
<td>DLDS forwarding table access</td>
<td>100,000,000 routes</td>
<td>10.65 ns per packet</td>
</tr>
</tbody>
</table>

### Traffic Overhead (relative to a 10-minutes 384 kb/s multimedia flow)

<table>
<thead>
<tr>
<th>Activity</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed linked data structure setup</td>
<td>0.0002 %</td>
</tr>
<tr>
<td>Keepalive message</td>
<td>0.08 %</td>
</tr>
<tr>
<td>Alert option, pointer and hop counter in data packets</td>
<td>0.6 %</td>
</tr>
</tbody>
</table>

*CPU times have been measured on a 1.6 GHz Intel® Core 2 computer*
Deployment

- No interaction with (nor change in) the underlying routing protocols is required
- Autonomous recovery of errors, faults and route changes
- If information stored in the DLDS becomes obsolete, packet handling is reverted to best-effort, lookup-driven forwarding
- Packets are never dropped nor misrouted
- It works even on partially REBOOK/DLDS-unaware paths
- It works across multiple Autonomous Systems
- It does not require any agreement between network managers
- It can be implemented in an extremely lightweight protocol
References


In the articles…

- Distributed control function for fairness and optimality
- Deployment
- Security
- Fast packet forwarding
- Implementation details
Conclusion

- **Some instances of ICN can use REBOOK**
  - for congestion- and flow-controlled transport of objects from a given location to the interested receiver
  - to provide fast packet forwarding in software-based routers or inexpensive hardware implementation

- **Why ICN? Why REBOOK?**
  - new architecture that overcome the rigid separation (and mistrust) between hosts/applications and the network
Thank you!
Other scenarios

Outside the cloud: Overlay Network
Other scenarios (cont’d)

Inside the cloud: REBOOK/DLDS-aware routers
Other scenarios (cont’d)

- REBOOK-aware client
- REBOOK-aware server

- REBOOK-unaware client
- REBOOK-aware proxy server
- REBOOK-aware proxy server

- REBOOK-unaware client
- REBOOK-aware traffic-shaping router
- REBOOK-aware traffic-shaping router

- REBOOK-unaware client
- REBOOK-unaware server
Performance
(access to the forwarding table)

<table>
<thead>
<tr>
<th>Reference</th>
<th>configuration</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>ART-16-8-8</td>
<td>~50 K routes</td>
<td>3</td>
</tr>
<tr>
<td>ART</td>
<td>~50 K routes</td>
<td>4.7</td>
</tr>
<tr>
<td>SMART</td>
<td>~50 K routes</td>
<td>4.7</td>
</tr>
<tr>
<td>CPE</td>
<td>~50 K routes</td>
<td>5.3</td>
</tr>
<tr>
<td>BSD Radix</td>
<td>~50 K routes</td>
<td>47</td>
</tr>
<tr>
<td>Binary trie</td>
<td>5,000 routes</td>
<td>138</td>
</tr>
<tr>
<td>LC-trie</td>
<td>5,000 routes</td>
<td>246</td>
</tr>
<tr>
<td>Modified LC-trie</td>
<td>5,000 routes</td>
<td>239</td>
</tr>
<tr>
<td>Prefix-tree</td>
<td>5,000 routes</td>
<td>131</td>
</tr>
<tr>
<td>DTBM</td>
<td>5,000 routes</td>
<td>191</td>
</tr>
<tr>
<td>7-FST</td>
<td>5,000 routes</td>
<td>114</td>
</tr>
<tr>
<td>2-MPT</td>
<td>5,000 routes</td>
<td>99</td>
</tr>
</tbody>
</table>