Simple Ad hoc Routing
with LUNAR

2nd Swedish Workshop on Wireless Ad-hoc Networks
March 5–6, 2002

Christian Tschudin & Richard Gold
Uppsala University / FhG FOKUS Berlin
Abstract

In this talk we present a new ad hoc routing system based upon simple principles regarding the routing strategy and the implementation approach. In the routing area we (re-)introduce the end-to-end principle, letting the communicating end nodes make the decisions concerning the behaviour of transit nodes. The routing strategy that we adopted is a mixture of on-demand and pro-active routing which thereby minimizes the possible down-times of communication paths.

Implementation-wise we use explicit “resolution commands” sent to neighbour nodes, which steer functionality in a way which resembles remote patching. A freely available implementation has been produced that includes auto-configuration of IP network addresses and default gateway routing, making LUNAR a fully self-configuring ad-hoc routing solution which supports both unicast and broadcast styles of communication.
Overview

LUNAR = “Lightweight Underlay Network Ad hoc Routing”

- LUNAR simplicity:
  - Ad hoc routing = ARP forwarding

- “Underlay” (simplicity again):
  - “active networking inside”
  - supports multiple routing styles

- As simple as it can get: Self-Configuration
Is Wireless Ad-hoc Complex?

Research being undertaken since long (early 90ies)

- Lack of robust implementations with sensible licenses
- Never mind the code, read the Internet Drafts . . .
- Is IETF MANET aiming at the right target?
  (100 nodes and more)
- Immediate need for small wireless networks
  (8–12 nodes, 2–3 hops) : access, spontaneous NWs
LUNAR: Path Discovery

LUNAR’s core: ARP Rewriting

- Address Resolution Protocol, RFC 826:
  - node A has B’s IP address, needs B’s HW addr
  - A broadcasts an ARP request
  - B replies with its HW address

- LUNAR does multi-hop ARP:
  - intermediate nodes forward ARP requests
  - builds data path on-the-fly while replying
  (early DSR also was multihop ARP)
LUNAR Path Maintenance: Don’t ask – Do it!

- LUNAR is reactive:
  - path is created only on demand, triggered by ARP req

So far so good, but . . .

- Classic reactive protocols (e.g. AODV and DSR) rely on *path maintenance*

- LUNAR:
  - do not bother about broken links
  - just create a new path every 3 seconds (if \( \exists \) demand)

  “proactive path re–discovery”
Don’t try to make Ad-Hoc “normal” – abstract away from IP, i.e. L2.5 routing

LUNAR: creates IP subnet illusion for unicast and broadcast
LUNAR’s Underlay Network: SelNet

- Trapping all interactions at IP / subnet border:
  - ARP
  - data packets, HW addresses

- Rewriting into internal “language”:
  - eXtensible Resolution Protocol (XRP)
  - SelNet shim forwarding layer

- Different ARP rewriting strategies possible:
  - LUNAR, source route (DSR style), private variant, etc
XRP: eXtensible Resolution Protocol

- Make resolution an explicit, mandatory step:
  - resolve IP addresses
  - resolve URLs (TRIAD)
- create content tunnels, overlays, virtual NWs . . .

- Resolution is main mechanism for redirection
  (repointering the Internet)

- XRP as generic SelNet control interface:
  send a series of commands (active networking)
a) The node S wants to resolve the name for target T: first, it creates a remote tunnel entry R at node I

b) S sends the resolution request, I propagates it after having created a remote tunnel entry R’ at node T
c) T replies via R’, node I resolves by creating a tunnel entry D and replies via R to node S
d) S receives the reply, installs D’ as its tunnel entry: the resolution yielded a tunnel to target node T
LUNAR’s IP integration: NO start parameters needed

- Automatic address assignment
  - fixed IP subnet number
  - random host ID, probe for conflicts (Appletalk, APIPA)

- LUNAR Gateway node runs NAT, DHCP server:
  - copies non-subnet packets to wireside interface

- LUNAR “internal” nodes:
  - get DNS and default GW via DHCP_INFORM
LUNAR Implementation (user space daemon)

<table>
<thead>
<tr>
<th># of C lines</th>
<th>Module</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>550</td>
<td>lunar.c</td>
<td>main() program and self-config logic</td>
</tr>
<tr>
<td>420</td>
<td>netbox.c</td>
<td>OS glue and timer support</td>
</tr>
<tr>
<td>330</td>
<td>sapf.c</td>
<td>SAPF forwarding engine</td>
</tr>
<tr>
<td>500</td>
<td>xrp.c</td>
<td>XRP and resolution forwarding logic</td>
</tr>
<tr>
<td>160</td>
<td>xrp_pkt.c</td>
<td>XRP encoding/parsing</td>
</tr>
<tr>
<td>1960</td>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

Compare with DSR (2’500 lines), OLSR (8’000 lines).
Note: these implementations have no self-config.
Outlook

- LUNAR is about:
  - solve the common, useful case first
  - keep it simple, stupid

- Implementation in hours or days, not years, because:
  - simple routing style
  - IP subnet emulation
  - active networking style

- Overall: Surprisingly robust, good performance and available today . . .
LUNAR = “Lightweight Underlay Network Ad hoc Routing”

**Wireless Ad hoc:**
- providing virtual IP subnet
- dynamic creation, maintenance
- selfconfig (addr, gateway, NAT)
- AN inside: “remote patching”

**Demo floppy disk:**
- need 802.11 card, eth optional
- boot from floppy
- try ping, wget, khttp

[http://www.docs.uu.se/selnet/lunar/](http://www.docs.uu.se/selnet/lunar/)