Topological Addressing Enabling Efficient IoT Communication

Luigi Iannone, Zhe Lou, Li Guangpeng
Motivation

- Massive expansion of the network edge that is driven by the "Internet of Things" (IoT) technology
Usual Routing approach

- 6LowPAN/6Lo/LPWAN WGs, in the IETF, address many foundational issues for those type of deployments

- Existing solutions, however, may have some shortcomings:

<table>
<thead>
<tr>
<th>Technology</th>
<th>Problem (especially in large scale LLN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6lowpan-DHCPv6</td>
<td>Consume bandwidth and time before node working, by applying for address from centralized server through multiple hops</td>
</tr>
<tr>
<td>6lowpan-AutoConfig</td>
<td>Using large address space to reduce confliction, implicating longer address and larger routing table, thus limit scale of network</td>
</tr>
<tr>
<td>6lowpan-RH</td>
<td>RPL information causes extra overhead of packet. Routers consumes resource to advertise, store, manage routing table</td>
</tr>
<tr>
<td>6lowpan-IPHC</td>
<td>Context based address uncompressing consumes extra computing resource. 6lo-RPL (RFC 8138) avoids uncompressing hop-by-hop, however bring much more complexity in routing.</td>
</tr>
</tbody>
</table>


Does simpler and more efficient addressing/routing exist based on (but beyond) previous work?
Native Short Addresses (NSA) Architecture
Native Short Addresses (NSA) Architecture

The root node is responsible for the management of the overall NSA network. It functions as a gateway between the NSA domain and the Internet.
The root node is responsible for the management of the overall NSA network. It functions as a gateway between the NSA domain and the Internet.

A forwarder node is responsible for, as the name suggests, forwarding traffic between its parent and the children, according to NSA addressing.
The root node is responsible for the management of the overall NSA network. It functions as a gateway between the NSA domain and the Internet.

A forwarder node is responsible for, as the name suggests, forwarding traffic between its parent and the children, according to NSA addressing.

A leaf node has relatively simpler operation, as it is either a source or destination of traffic.
Address Allocation Function

Algorithm 1 NSA Allocation Function pseudo code

1: function $AF(\text{role}, f, l)$
2: \hspace{1em} $\text{pfx} \leftarrow \text{Current Node NSA Address}$
3: \hspace{1em} if (\text{role} = \text{forwarder}) then
4: \hspace{2em} $\text{sfx} \leftarrow \text{Concatenate}(b(f), 0)$
5: \hspace{2em} $f \leftarrow f + 1$
6: \hspace{1em} else
7: \hspace{2em} $\text{sfx} \leftarrow \text{Concatenate}(b(l), 1)$
8: \hspace{2em} $l \leftarrow l + 1$
9: \hspace{1em} end if
10: \hspace{1em} return $\text{Concatenate}(\text{pfx}, \text{sfx})$
11: end function

New Address = Parent Address + string of ‘1’ + (0 if forwarder; 1 if Leaf)
Address Allocation (1/5)

Algorithm 1 NSA Allocation Function pseudo code

1: function $A(f, l)$
2: \[ pfx \leftarrow \text{Current Node NSA Address} \]
3: \[ \text{if (role} = \text{forwarder}) \text{ then} \]
4: \[ sfx \leftarrow \text{Concatenate}(b(f), 0) \]
5: \[ f \leftarrow f + 1 \]
6: \[ \text{else} \]
7: \[ sfx \leftarrow \text{Concatenate}(b(l), 1) \]
8: \[ l \leftarrow l + 1 \]
9: \[ \text{end if} \]
10: \[ \text{return Concatenate}(pfx, sfx) \]
11: \[ \text{end function} \]

AR – Address Request

Node with no address
Root
Forwarder
Leaf
Address Allocation (2/5)

Border Gateway (root)

AR ({forwarder | leaf}, Nodeid=X)

AR – Address Request

Node with no address

Root

Forwarder

Leaf

Algorithm 1 NSA Allocation Function pseudo code

1: function Af(role, f, l)
2: \( pfx \leftarrow \text{Current Node NSA Address} \)
3: if (role = forwarder) then
4: \( sfx \leftarrow \text{Concatenate}(b(f), 0) \)
5: \( f \leftarrow f + 1 \)
6: else
7: \( sfx \leftarrow \text{Concatenate}(b(l), 1) \)
8: \( l \leftarrow l + 1 \)
9: end if
10: return Concatenate(pfx, sfx)
11: end function
Address Allocation (2/5)

**Algorithm 1** NSA Allocation Function pseudo code

1: function $Af(\text{role}, f, l)$
2:     $pfx \leftarrow \text{Current Node NSA Address}$
3:     if ($\text{role} = \text{forwarder}$) then
4:         $sfx \leftarrow \text{Concatenate}(b(f), 0)$
5:         $f \leftarrow f + 1$
6:     else
7:         $sfx \leftarrow \text{Concatenate}(b(l), 1)$
8:         $l \leftarrow l + 1$
9:     end if
10:    return $\text{Concatenate}(pfx, sfx)$
11: end function

**Diagram:**

- **Border Gateway** (root)
- **AR** — Address Request
- **Node with no address**
- **Root**
- **Forwarder**
- **Leaf**

*Neither forwarder nor root -> Drop AR silently*
Address Allocation (3/5)

Algorithm 1 NSA Allocation Function pseudo code

1: function Al(role, f, l)
2:   pfX ← Current Node NSA Address
3:   if (role = forwarder) then
4:       sfx ← Concatenate(b(f), 0)
5:       f ← f + 1
6:     else
7:       sfx ← Concatenate(b(l), 1)
8:       l ← l + 1
9:   end if
10:  return Concatenate(pfX, sfx)
11: end function

AR (forwarder | leaf, Nodeid=X)

AR – Address Request

Node with no address  Root  Forwarder  Leaf
Address Allocation (4/5)

Algorithm 1 NSA Allocation Function pseudo code

1: function Af(role, f, l)
2:     \( pf = \text{Current Node NSA Address} \)
3:     if (role = forwarder) then
4:         \( sf = \text{Concatenate}(b(f), 0) \)
5:         \( f \leftarrow f + 1 \)
6:     else
7:         \( sf = \text{Concatenate}(b(l), 1) \)
8:         \( l \leftarrow l + 1 \)
9:     end if
10:    return Concatenate(pf, sf)
11: end function

AR (forwarder, Nodeid=X)

AR – Address Request
Node with no address  Root  Forwarder  Leaf
Address Allocation (4/5)

1. Pfx = 1 (Root address)
2. B(f) = 1 (Second forwarder child)
3. Concatenate Pfx+B(f)+0 = 1+1+0
4. Offered Address: 110

Algorithm 1 NSA Allocation Function pseudo code

```
function AF(role, f, l)
    pfx ← Current Node NSA Address
    if (role = forwarder) then
        sfx ← Concatenate(b(f), 0)
        f ← f + 1
    else
        sfx ← Concatenate(b(l), 1)
        l ← l + 1
    end if
    return Concatenate(pfx, sfx)
end function
```
Address Allocation (4/5)

Border Gateway (root)

1. Pfx = 1 (Root address)
2. B(f) = 1 (Second forwarder child)
3. Concatenate Pfx+B(f)+0 = 1+1+0
4. Offered Address: 110

AR (forwarder, Nodeid=X)

1. Pfx = 10 (Forwarder address)
2. B(f) = ” (First forwarder child)
3. Concatenate Pfx+B(f)+0 = 10+”+0
4. Offered Address: 100

Algorithm 1 NSA Allocation Function pseudo code

```plaintext
function Af(role, f, l)
    pfx ← Current Node NSA Address
    if (role = forwarder) then
        sfx ← Concatenate(b(f), 0)
        f ← f + 1
    else
        sfx ← Concatenate(b(l), 1)
        l ← l + 1
    end if
    return Concatenate(pfx, sfx)
end function
```
Address Allocation (5/5)

Border Gateway (root)

Algorithm 1 NSA Allocation Function pseudo code

1: function Af(role, f, l)
2:     pfx ← Current Node NSA Address
3:     if (role = forwarder) then
4:         sfx ← Concatenate(b(f), 0)
5:         f ← f + 1
6:     else
7:         sfx ← Concatenate(b(l), 1)
8:         l ← l + 1
9:     end if
10:    return Concatenate(pfx, sfx)
11: end function
NSA Stateless Forwarding

Leaf Node
Forwarder Node
DA: Destination Address
CA: Current Node’s Address
NSA Stateless Forwarding

Leaf nodes just send packets to their parent.
NSA Stateless Forwarding

Leaf nodes just send packets to their parent.
NSA Stateless Forwarding

1. Parse DA from Packet

2. If $\text{Len}(\text{DA}) < \text{Len}(\text{CA})$:
   - Forward to Parent Node

3. If $\text{Len}(\text{DA}) = \text{Len}(\text{CA})$:
   - If $\text{CA} == \text{DA}$:
     - Forward to Parent Node
   - If $\text{CA} == \text{PrefixOf}(\text{DA})$:
     - Calculate Next-hop & Forward the packet

Leaf Node
Forwarder Node
DA: Destination Address
CA: Current Node’s Address
**NSA Stateless Forwarding**

- **Leaf Node**
- **Forwarder Node**
- **DA**: Destination Address
- **CA**: Current Node’s Address

### Algorithm:
1. Parse DA from Packet
2. If Len(DA) < Len(CA), then Next-hop is the node.
3. If Len(DA) = Len(CA) and CA == PrefixOf(DA), then Next-hop is the node.
4. If CA == DA, then Next-hop is the node.
5. Forward to Parent Node if no Next-hop is found.

### Diagram:
- Node 1 (Root) with DA:1101
- Nodes 10, 11, 110, 111, 1010, 1111
- Leaf nodes: 1000, 1001, 10010, 10011, 100101
- Forwarder nodes: 10, 11, 110, 111, 1010, 1111

---

**Example:**
- Node 1 (Root) with DA:1101
NSA Stateless Forwarding

Leaf Node
Forwarder Node
DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet

Len(DA) < Len(CA)

Len(DA) = Len(CA)

CA==PrefixOf(DA)

Calculate Next-hop & Forward the packet

Forward to Parent Node
NSA Stateless Forwarding

Leaf Node
Forwarder Node

DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet

Len(DA) < Len(CA)
Len(DA) = Len(CA)
CA==PrefixOf(DA)

CA==DA?

yes

no

yes

no

Calculate Next-hop & Forward the packet

Forward to Parent Node
NSA Stateless Forwarding

Leaf Node
Forwarder Node

DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet

Len(DA) < Len(CA)
Len(DA) = Len(CA)

CA==PrefixOff(DA)

CA==DA?

Calculate Next-hop & Forward the packet

Forward to Parent Node
NSA Stateless Forwarding

Leaf Node
Forwarder Node

DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet
Len(DA) < Len(CA)
Len(DA) = Len(CA)  CA==PrefixOf(DA)

Calculate Next-hop & Forward the packet

Forward to Parent Node

1000 1001 10010 10011 100101

DA: 1101
NSA Stateless Forwarding

Leaf Node: DA: 1101
Forwarder Node: CA: 1101

Parse DA from Packet

- Len(DA) < Len(CA)
- Len(DA) = Len(CA)
  - CA == PrefixOf(DA)
  - CA == DA?
  - CA == PrefixOf(DA)

Calculate Next-hop & Forward the packet

Forward to Parent Node
NSA Stateless Forwarding

- **DA**: Destination Address
- **CA**: Current Node’s Address

1. **Parse DA from Packet**
2. **Len(DA) < Len(CA)**
   - **yes**
     - **CA==PrefixOf(DA)**
       - **yes**
         - **CA==DA?**
           - **yes**
             - **Calculate Next-hop & Forward the packet**
           - **no**
             - **Forward to Parent Node**
         - **no**
           - **Len(DA) = Len(CA)**
             - **yes**
               - **CA==DA?**
                 - **yes**
                   - **Calculate Next-hop & Forward the packet**
                 - **no**
                   - **Forward to Parent Node**
             - **no**
               - **Len(DA) < Len(CA)**
                 - **yes**
                   - **CA==PrefixOf(DA)**
                     - **yes**
                       - **Calculate Next-hop & Forward the packet**
                     - **no**
                       - **Forward to Parent Node**
                 - **no**
                   - **Forward to Parent Node**

- **Leaf Node**
- **Forwarder Node**
NSA Stateless Forwarding

Leaf Node
Forwarder Node

DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet

Len(DA) < Len(CA)
Len(DA) = Len(CA)
CA==PrefixOf(DA)

yes
no
yes

no

CA==DA?

yes
no

yes
no

Forward to Parent Node

Calculate Next-hop & Forward the packet
NSA Stateless Forwarding

- **Leaf Node**
- **Forwarder Node**
- **DA (Destination Address)**
- **CA (Current Node's Address)**

**Flowchart:**
1. Parse DA from Packet
2. If Len(DA) < Len(CA), then yes and proceed to the next step; if no, then Forward to Parent Node.
3. If Len(DA) = Len(CA), then:
   - If CA == PrefixOf(DA), then yes and calculate next-hop and forward the packet; if no, then no and Forward to Parent Node.
   - If CA == DA, then yes and calculate next-hop and forward the packet; if no, then Forward to Parent Node.

**Diagram:**
- Nodes represent leaf and forwarder nodes.
- DA:1101 is highlighted in the diagram.
- The structure shows how packets are routed based on the comparison of DA and CA lengths and prefix comparison.
NSA Stateless Forwarding

Parse DA from Packet

\[ \text{Len}(\text{DA}) < \text{Len}(\text{CA}) \]

\[ \text{Len}(\text{DA}) = \text{Len}(\text{CA}) \]

\[ \text{CA} == \text{PrefixOf}(\text{DA}) \]

Calculate Next-hop & Forward the packet

Forward to Parent Node
NSA Stateless Forwarding

Leaf Node: DA:1101
Forwarder Node: CA: PrefixOf(DA)

- Len(DA) < Len(CA) → yes
- Len(DA) = Len(CA) → CA==DA?
  - yes → CA==PrefixOf(DA)?
    - yes → Calculate Next-hop & Forward the packet
    - no → no
  - no → Forward to Parent Node
- Len(DA) < Len(CA) → yes
- Len(DA) = Len(CA) → CA==DA?
  - yes → CA==PrefixOf(DA)?
    - yes → Calculate Next-hop & Forward the packet
    - no → no
  - no → Forward to Parent Node
NSA Stateless Forwarding

Leaf Node
Forwarder Node
DA: Destination Address
CA: Current Node’s Address

Parse DA from Packet
Len(DA) < Len(CA)

Len(DA) = Len(CA)
CA==PrefixOf(DA)

CA==DA?

Calculate Next-hop & Forward the packet
Forward to Parent Node
NSA Stateless Forwarding

**Leaf Node**

1000, 1001, 10010, 10011

**Forwarder Node**

10, 11, 110, 111, 1010, 1111

**DA: Destination Address**

1101

**CA: Current Node’s Address**

10, 11, 110, 111, 1010, 1111

1.

1. Parse DA from Packet

2. If Len(DA) < Len(CA)
   - Yes: CA == PrefixOf(DA)
   - No: Len(DA) = Len(CA)
3. If Len(DA) = Len(CA)
   - Yes: CA == DA?
   - No: CA == PrefixOf(DA)
4. If CA == DA?
   - Yes: Calculate Next-hop & Forward the packet
   - No: Forward to Parent Node

Leaf Node

Forwarder Node

DA: Destination Address

CA: Current Node’s Address
**NSA Stateless Forwarding**

**Leaf Node**
- DA: Destination Address
- CA: Current Node’s Address

**Forwarder Node**

- Parse DA from Packet
- Len(DA) < Len(CA)
- Len(DA) = Len(CA)
  - CA == PrefixOf(DA)
  - Calculate Next-hop & Forward the packet

- CA == DA?
  - yes
  - no

- Forward to Parent Node
NSA Stateless Forwarding

### Parse DA from Packet

1. **Len(DA) < Len(CA)**
   - **no**
   - **yes**

2. **Len(DA) = Len(CA)**
   - **no**
   - **yes**

3. **CA==PrefixOf(DA)**
   - **no**
   - **yes**

4. **CA==DA?**
   - **no**
   - **yes**

5. **Calculate Next-hop & Forward the packet**

**Leaf Node**

**Forwarder Node**

**DA: Destination Address**

**CA: Current Node’s Address**
NSA Stateless Forwarding

Leaf Node: 100, 1001, 10010, 10011, 100101
Forwarder Node: 10, 11, 110, 111, 1010, 1111

DA: Destination Address
CA: Current Node’s Address

- Parse DA from Packet
- Len(DA) < Len(CA)
- Len(DA) = Len(CA)
  - CA == PrefixOf(DA)
  - CA == DA?
  - CA == PrefixOff(DA)
    - yes
    - no
    - yes
    - no
- Calculate Next-hop
- Forward the packet
- Forward to Parent Node
NSA StatelessForwarding

Leaf Node: DA: Destination Address
Forwarder Node: CA: Current Node’s Address

Parse DA from Packet
Len(DA) < Len(CA)
Len(DA) = Len(CA)
CA == PrefixOf(DA)
Calculate Next-hop & Forward the packet
Forward to Parent Node

Leaf Node: 1000, 1001, 10010, 10011, 100101
Forwarder Node: 10, 11, 110, 111, 1010, 1111
NSA Stateless Forwarding

DA: Destination Address
CA: Current Node’s Address

1. Parse DA from Packet
2. Len(DA) < Len(CA)
3. Len(DA) = Len(CA)
4. CA == PrefixOf(DA)
5. CA == DA?
6. CA == PrefixOf(DA)

If yes, calculate next-hop & forward the packet

Leaf Node
Forwarder Node

Forward to Parent Node
**NSA Stateless Forwarding**

- **Leaf Node**: DA: Destination Address
- **Forwarder Node**: CA: Current Node’s Address

**Flowchart Explanation**:

1. **Parse DA from Packet**
2. **Len(DA) < Len(CA)**
   - **yes**: Forward to Parent Node
   - **no**: **Len(DA) = Len(CA)**
     - **yes**: CA==DA?
       - **yes**: Calculate Next-hop & Forward the packet
       - **no**: CA==PrefixOf(DA)
         - **yes**: Forward to Parent Node
         - **no**: no
3. **CA==DA?**
   - **yes**: Calculate Next-hop & Forward the packet
   - **no**: no

**Diagram Details**:

- **Leaf Node** 1000, 1001, 10010, 10011, 100101
- **Forwarder Node** 10, 11, 110, 111, 1010, 1111

**Example**:

- DA: 1101
- Len(DA) < Len(CA)
- CA==PrefixOf(DA)
- CA==DA?
- yes: Calculate Next-hop & Forward the packet
- no: no

**Flowchart**:

1. Parse DA from Packet
2. Len(DA) < Len(CA)?
   - yes: Forward to Parent Node
   - no: Len(DA) = Len(CA)?
     - yes: CA==DA?
       - yes: Calculate Next-hop & Forward the packet
       - no: CA==PrefixOf(DA)
         - yes: Forward to Parent Node
         - no: no
     - no: no
3. CA==DA?
   - yes: Calculate Next-hop & Forward the packet
   - no: no
Routing between NSA and IPv6 domains

• For NSA domain
  
  • The border router (root) can translate to IPv6 addresses
  
  • Concatenating IPv6 prefix and native short address (padding with 0s).
  
  • Trimming the IPv6 prefix and suppressing leading 0s in the suffix

• For external IPv6 addresses:
  
  • Border router maintain a table mapping external IPv6 destinations to a short address
  
  • Packet carries mapped address when transmitting in the domain
  
  • Border router will look up real IPv6 destination before sending the packets to IPv6 domain
6lo NSA Routing Header

- **RFC 8138:**
  - IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header
  - Allows to define special type of header prepended to the 6lo-compressed IPv6 header

- **6lo NSA Routing Header Format:**

  0                   1
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
  +------------------------------------------+--
  |1|0|0| size |0| Type     | Address 1 to 16 bytes |
  +------------------------------------------+--

  `<-- Length implied by Type/TSE -->`

**Size:** 4 bits

0: unused

**Type:** `<NSA_Type, I/O, MA>`

I/O: Inward/Outward

MA: Mapped Address

I/O = 0, MA = 0: Destination address not NSA, neither a mapped Address (aka external full IPv6 address)

I/O = 0, MA = 1: Destination address is an external mapped address

I/O = 1, MA = 0: Destination and source addresses are NSA

I/O = 1, MA = 1: Destination address is NSA, source address is an external mapped address
Take away

Topological addressing scheme
• Suitable for IoT networks with relative stable connections
• Can achieve smaller average address length (see figure)
  • even if it is not the main target
• Enables stateless forwarding
  • Enabling greener solution
  • Working on very constrained nodes

Future Research direction
• Reliability
• Quantification of gain w.r.t. routing
  • e.g. RPL
Any question/comment welcome!

THANKS!