Fighting the Reliability Problem
or
Who Cares about Routing in WSN!

by Pawel Gburzynski
Popular ad-hoc routing schemes ...

... are all based on point-to-point forwarding ...

... i.e., setting up hop-by-hop “paths” in the network
How do they work?

Suppose that some two nodes, call them A and B, want to exchange packets.
How do they work?

The network must identify a path, i.e., an exact sequence of intermediate nodes to forward those packets.
How do they work?

This involves lots of distributed negotiations, evaluations, and bookkeeping.
Details depend on the scheme

**Reactive:** only look for a path when a specific connection is required

**Proactive:** constantly negotiate paths for all possible (or anticipated) connections to be ready in advance

Many of them are hybrid; some maintain multiple (alternative) paths

Trading complexity (overhead) for responsiveness
Once the path has been established...

... the packet will be addressed on each hop to the single, specific, pre-determined, *next* node
Why is this bad?

- The procedure for discovering the path(s) is tricky and complex.
- The nodes must remember some descriptions of all active (pro-active?) paths passing through them.
- If one node on a path fails, the path becomes broken (its fixing may be as complex as finding a new path).
- Nodes come and go; they are unreliable.
This approach is a legacy of wired networking

If the links are in fact wires, and the nodes don’t move, that idea works quite well (see the stationary Internet)
But there are no wires in the wireless world!

You **never** send your packet to the precisely one next hop node.
The P-P schemes view this as a nuisance that must be fought …

... through various collision avoidance techniques that isolate “uninterested” neighbors from the one supposedly “linked” to the sender

Those techniques only work (somewhat) if data packets are long; they are useless when sending small amounts of data ...
Such schemes do not fit the poor reliability model of ad-hoc systems

P-P paths are contrived and brittle: you either have a (full) path, or have no path at all (note that the remnants of a broken path may be completely useless)

Nodes are inherently unreliable. Don’t whine about it! This is OK! This is their charm!

Just learn to avoid contrived and fragile solutions, i.e., ones whose global integrity depends on the reliability of a single component!
TARP does away with all that; it assumes that:

- The broadcast nature of the medium is a **feature**, not a flaw! **No wires!** Deal with it!
- Instead of wasting **time, memory, and bandwidth** on discovering and recovering the illusory paths, you should be sending the damn packets!
- Wisdom will emerge as the regular, useful, packets propagate and are **overheard** by all the nodes than can **rightfully** hear them
- No single item of that wisdom is critical (in the sense that the fate of an elaborate “connection” would entirely depend upon it)
The simple idea:

I am A and I have a packet addressed to B; never heard of B before; what to do?

A P-P scheme would say: hey, let’s send some queries around and wait until everybody learns exactly how to go

TARP says: let’s send the packet right away

I do have to send it eventually, right?

no matter where I think it goes, all my neighbors will hear it anyway, right?

so why bother with the stupid queries?
This is a difference in paradigm; the forwarder’s dilemma:

- Where should I forward the packet?
- How can I learn the identity of the next node on the path?
- How do I make sure to know that identity at all times?
- Should I transmit (broadcast) the packet?
- Will I help when I do that?
- Won’t my assistance be redundant?
TARP is extremely proactive in the cheapest, most natural sense:

I cannot forward unless I know exactly how and when:

- I must explicitly receive the packet first
- Then I must know where to send it next

I cannot stop forwarding unless I am sure my help is not needed:

- Nobody tells me what to receive: I receive what I hear
- I forward packets by default
- Unless I have learned that I am not helpful
In simple words:

P-P is obsessive about learning how to help

It has to be obsessive, because it cannot forward at all unless it has the knowledge

The knowledge is brittle and elaborate (it relates to a volatile path within the mesh)

The knowledge must be complete to be of value
In simple words:

TARP is **not so** obsessive about learning when **not** to help

It doesn’t have to be obsessive, because the lack of knowledge is not as harmful

Partial knowledge is meaningful! A better informed node will use less bandwidth ...

... which means that any node can help, according to its means
TARP does connote with flooding (boo!), but don’t let them fool you!

There is no way to get something for nothing! All those complicated P-P schemes need flooding at least for path discovery!

When TARP knows nothing, it starts by naive flooding; when they know nothing, they are down to (necessarily naive) flooding, too; so we are even there

But we can do much better than that!
TARP is driven by a chain of rules.

- **Received packet**
  - **Am I the recipient?**
    - **Yes** → **Receive and forget**
    - **No**
      - **rule 1** → **ignore**
      - **rule 2** → **ignore**
      - **rule N** → **ignore**
      - **don’t know**
  - **Rebroadcast**
  - **Ignore**
TARP is driven by a chain of rules

- Received packet
  - seen already?
    - No
      - rule 1
        - don’t know
        - rule 2
          - don’t know
          - rule N
            - don’t know
            - other, smarter rules
            - ignore
  - Yes
    - Receive and forget
    - too many hops?
      - ignore
      - ignore
      - Ignore

Am I the recipient?

Yes

Receive and forget

No

Ignore
One of the essential rules: SPD

K has been seeing some packets sent by A and B. It keeps track of:

- the smallest recently noted number of hops from A ($h_A$)
- the smallest recently noted number of hops from B ($h_B$)

When B receives a packet from A, it notes the total number of hops $h_{AB}$ made by it. This number will be conveyed towards A in the header of a next packet going in the opposite direction, i.e., from B to A.

And, of course, it works the same way for A receiving a packet from B.
One of the essential rules: SPD

Suppose $K$ receives a packet sent by $A$ and going to $B$. $K$ sees that the packet has made $h$ hops so far. The rule has to decide between *ignore* and *don’t know*.

$K$ compares $h + h_B$ to $h_{BA}$. Note that if $h + h_B > h_{BA}$, the node has grounds to believe that there are better forwarders than itself (so the rule may say *ignore*).

A dampening parameter (slack) is usually applied; the rule says *ignore* if $h + h_B > h_{BA} + \text{slack}$. The role of slack is to provide for controllable intentional redundancy.
All rules of TARP follow a certain important philosophy

They are driven by cached information collected and stored by the node.

If the node has no room to store all the information, the rule cannot tell what to do, so it says *don’t know*.

This means that the packet will not be ignored; *it will be rebroadcast*!

This way we smoothly trade the node’s footprint for the redundancy of routes *(try this with a P-P scheme 😛)*.
No hiccups, no critical nodes

slack = 1

primary helpers

secondary helpers (also forwarding)

eliminates duplicates
No hiccups, no critical nodes

Suppose one of the primary helpers disappears
No hiccups, no critical nodes

what was previously removed as a duplicate, has no competition now

New secondary helpers automatically kick in
Why does it always have to be about routing!!??

A wireless network may want to solve a specific problem, rather than providing (complete) connectivity among its nodes; that may not even be a good prerequisite for solving the actual problem.

What we need is a meta-protocol (a way to implement mesh-ed interactions), rather than a general-purpose routing scheme.
Example

Find the maximum of all values (indicated by sensors) among all nodes
Example (traditional solution)

- The sink sends queries to all nodes and receives replies.
- Those queries propagate via P-P paths allowing the sink to reach explicit destinations.
- Replies travel from destinations to the sink via reverse P-P paths.
- You can optimize (e.g., merge/piggyback reports of intermediate nodes).
A TARP-like solution

A “partial”

{\mathcal{T}, \mathcal{M}, \Phi}

collection ID

Max

Contributors (bitmap?)

Remember:

maximum outgoing partial

Cache:

identities of neighbors: set \mathcal{N}

solved \mathcal{T}'s
A TARP-like solution

Received partial: \( \{T_I, M_I, \Phi_I\} \)

Yes

Solved already?

\( \Phi_I \leftarrow \Phi_I \cup \{\text{self}\} \)
\( M_I \leftarrow \max\{M_I, V\} \)

Yes

\( \Phi_I \subseteq \Phi_C \) ?

\( \Phi_O \leftarrow \Phi_C \)
\( \Phi_C = \Phi_I \cup \Phi_C \)
\( M_C = \max\{M_I, M_C\} \)

Yes

Solved

No

Broadcast:
\( \{T_C, M_C, \Phi_C\} \)

\( \Phi_C = \text{All} ? \)

All \( \subseteq \Phi_I \land N \subseteq \Phi_O \) ?

Yes

No
Natural ways to add fuzziness:

1. \( \Phi_1 \subseteq \Phi_c \)
   - Fail with some probability (e.g., depending on connectivity feedback)

2. \( N \subseteq \Phi_i \cap N \subseteq \Phi_o \)
   - Fail with some probability (depending on how well you think your neighborhood info is perceived and cached)
Customized TARP is part of the solution:

- Solved
- Forward the solution in a TARP-routed fashion to the sink

All routes lead to the sink: **customize the SDP cache** to never rid itself of the sink entry

May want to use a largish slack to tell everybody that the game is over
Two golden principles:

Collaboration (non contrivance):
No elaborate intrigues where the integrity of a complex action would depend on the well-being of a single member of a large cohort

Participation (scalability of effort):
Small (or less fortunate) should be able to help in proportion to their abilities