BMX7: Decentralized Routing Security for Community Mesh Networks

Axel Neumann
axel@ac.upc.edu

May 5, 2016
Wireless Battle Mesh v9 @ Porto, Portugal

Outline

1 Introduction

2 Protocol Overview

3 Protocol Messages

4 Integration and Validation

5 Conclusion and Appendix

Outline

1 Introduction

2 Protocol Overview

3 Protocol Messages

4 Integration and Validation

5 Conclusion and Appendix

BMX7! What? Why?

- BATMAN $\rightarrow$ BatMan eXperimental $\rightarrow$ BMX6 $\rightarrow$ BMX7
- BMX6
  - Isolate node properties into single node description (e.g. addresses, name, networks)
  - Propagate node description once and reference it via its hash (e.g. from routing updates)
- BMX7
  - Signed node descriptions (RSA2048)
  - Authenticated node IDentities
  - Ownership proving (crypto-generated) IPv6 addresses
  - Secure routing against untrusted nodes
  - Capacity and interference aware routing metric

Private vers. open networks

Small mesh(es) among friends

logical link (encrypted)

Group of friends

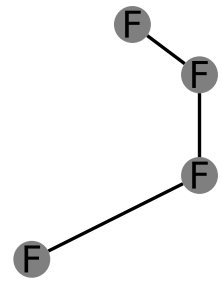
- **Priority: Functioning network!**
- Run any routing protocol
- No doubt about attacks from friends
- Excluding all potential attackers via full encryption

Other groups of friends...

- Same priority: Functioning network!
- Using different encryption key
  $\Rightarrow$ Logically disconnected networks

Result: Bunch of closed networks...

- No collaboration, no benefits!
- Individual nodes are just isolated



Private vers. open networks

Small mesh(es) among friends

logical link (encrypted)

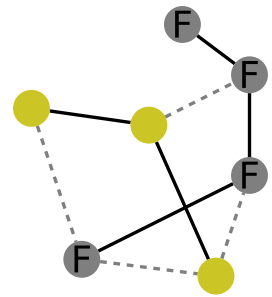
physical link

Group of friends

- **Priority: Functioning network!**
- Run any routing protocol
- No doubt about attacks from friends
- Excluding all potential attackers via full encryption

Other groups of friends...

- Same priority: Functioning network!
- Using different encryption key
  $\Rightarrow$ Logically disconnected networks



Result: Bunch of closed networks...

- No collaboration, no benefits!
- Individual nodes are just isolated

Private vers. open networks

Group of friends

- **Priority: Functioning network!**
- Run any routing protocol
- No doubt about attacks from friends
- Excluding all potential attackers via full encryption

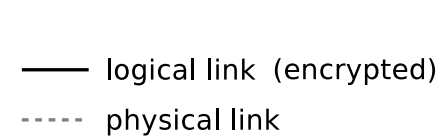
Other groups of friends...

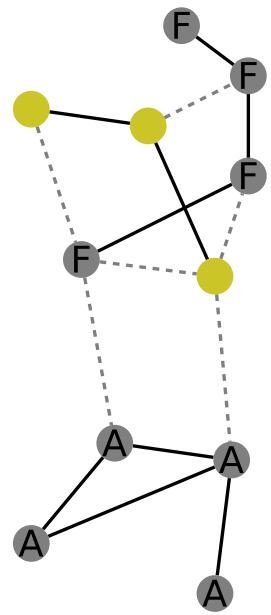
- Same priority: Functioning network!
- Using different encryption key
  $\Rightarrow$ Logically disconnected networks

Result: Bunch of closed networks...

- No collaboration, no benefits!
- Individual nodes are just isolated

Small mesh(es) among friends





Private vers. open networks

Small mesh(es) among friends

logical link (encrypted)

physical link

Group of friends

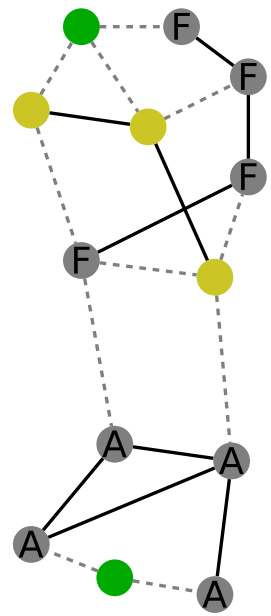
- **Priority: Functioning network!**
- Run any routing protocol
- No doubt about attacks from friends
- Excluding all potential attackers via full encryption

Other groups of friends...

- Same priority: Functioning network!
- Using different encryption key
  $\Rightarrow$ Logically disconnected networks

Result: Bunch of closed networks...

- No collaboration, no benefits!
- Individual nodes are just isolated



Private vers. open networks

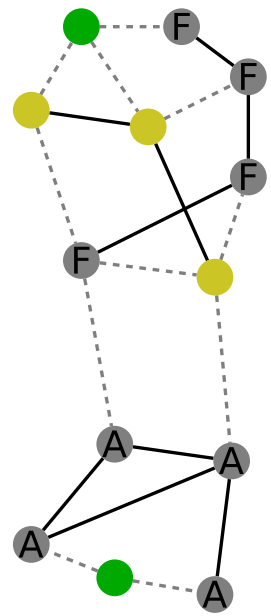
Small mesh(es) among friends

logical link (encrypted)

physical link

BMX7 can provide

- Allow individuals to use existing infrastructure
- Secure routing among trusted friends!
  Ensuring that unknown nodes can not mess with other node's routes



Securing an open and decentralized network!!??

Common problem: Single node can attack

- control plane (route establishment)
- data plane (traffic forwarding)

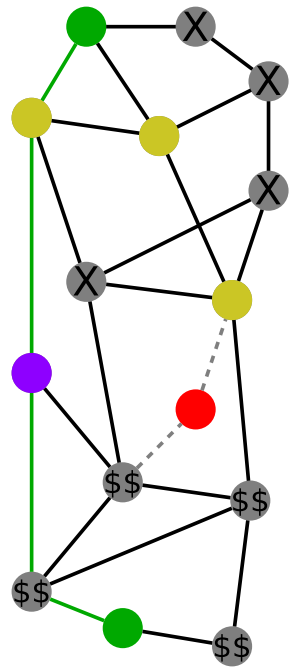
Common Solution: Access control, **exclude unreliable nodes**

- Easy for traditional ISP
  - Centralized administration of own routers
  - Supported via: Authenticated OSPF, SOLSR, Babel HMAC
- Trust & reliability assessment in CNs is hard
  - **Distributed administration**, partially **unknown nodes**
  - **Subversive attacks**: selective dropping, DPI & eavesdropping
  - **Trust is NOT a binary but a controversial policy decision**

Trust challenge for CNs: Reach consensus on set of reliable nodes

Open network, prone to subversive attacks

logical link
physical link
logical route
innocent node, trusting X & $$ nodes
eavesdropping node
selective dropper
excluded attacker



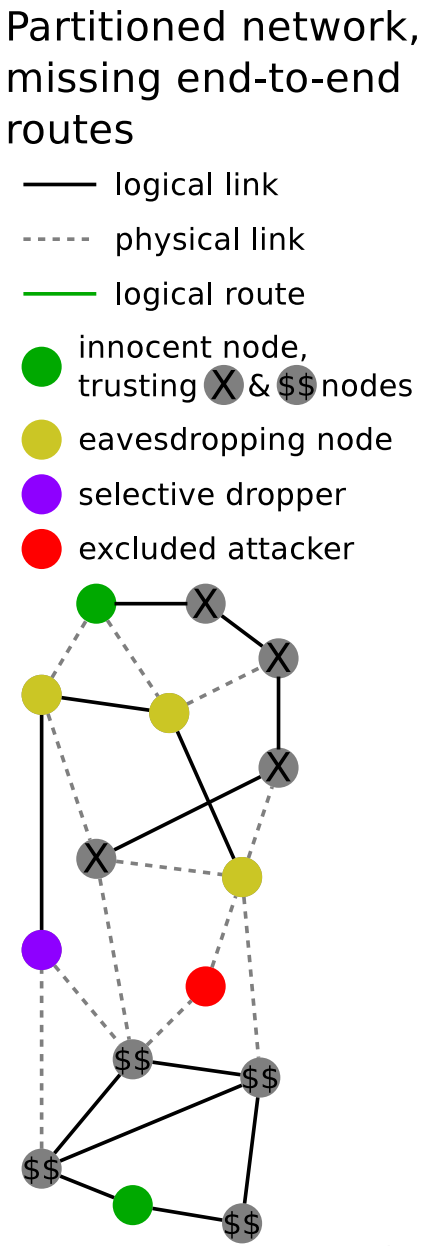
Exclusive trust set: Balancing...

- **Openness:** Exclude only malicious-proven nodes
  - How prove selective dropping or eavesdropping?
    ⇒ Few excluded. Potential attackers remain!
    ⇒ No more security :-(
- **Security:** Exclude all questionable nodes
  - e.g. anonymous, enthusiasts, kids, companies (competing), political, ...
    ⇒ No more openness :-(
    ⇒ Abandoned create own network ⇒ Partitioning!
- **Complexity to find consensus:** Hardly scales with increasing size!

Trust challenge for CNs: Reach consensus on set of reliable nodes

Exclusive trust set: Balancing...

- **Openness:** Exclude only malicious-proven nodes
  - How prove selective dropping or eavesdropping?
    ⇒ Few excluded. Potential attackers remain!
    ⇒ No more security :-(
- **Security:** Exclude all questionable nodes
  - e.g. anonymous, enthusiasts, kids, companies (competing), political, ...
    ⇒ No more openness :-(
    ⇒ Abandoned create own network ⇒ Partitioning!
- **Complexity to find consensus:** Hardly scales with increasing size!



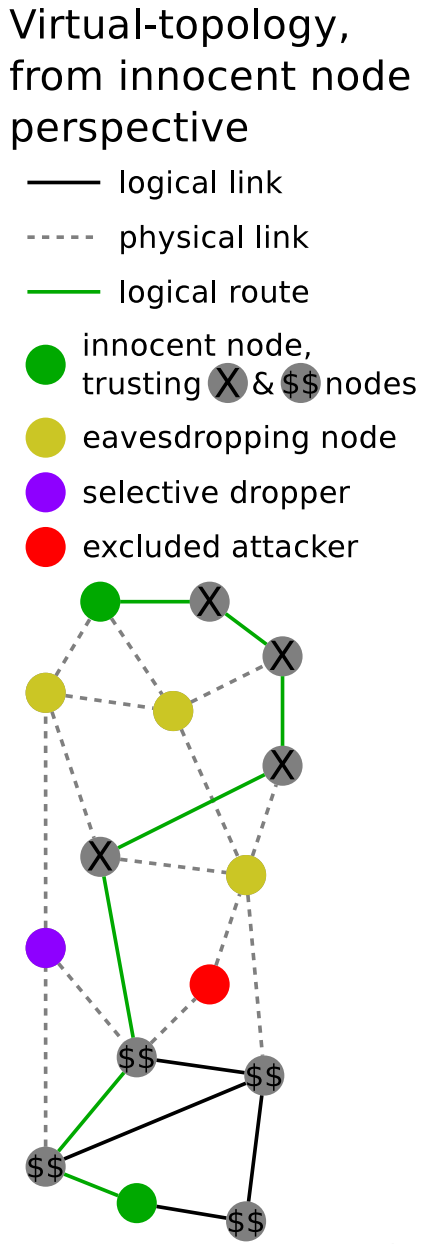
Trust challenge for CNs: Reach consensus on set of reliable nodes

Exclusive trust set: Balancing...

- **Openness:** Exclude only malicious-proven nodes
- **Security:** Exclude all questionable nodes
- **Complexity to find consensus:** Hardly scales!

Multiple trust sets -> parallel (virtual) topologies

- How many?
- Who decides?
- Consensus?
- Security?
- Overhead?



Trust challenge for CNs: Reach consensus on set of reliable nodes

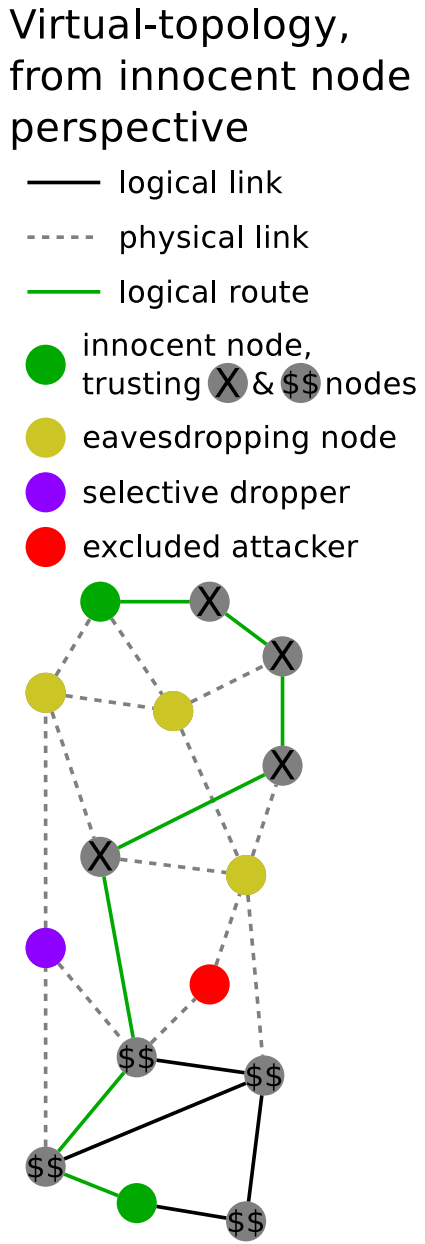
Exclusive trust set: Balancing...

- **Openness:** Exclude only malicious-proven nodes
- **Security:** Exclude all questionable nodes
- **Complexity to find consensus:** Hardly scales!

Multiple trust sets -> parallel (virtual) topologies

- How many? **One for each (admin)**
- Who decides? **Each on his own!**
- Consensus? **Not needed!**
- Security? **User (node admin) tailored!**
- Overhead? **Lets see...**

Freedom of choice is natural in public transport!
Why not also for public community networks?



Outline

1 Introduction

2 Protocol Overview

3 Protocol Messages

4 Integration and Validation

5 Conclusion and Appendix

SEMTOR Protocol Objectives

Securely-**E**ntrusted **M**ulti-**To**pology **R**outing

- **Secure against non-trusted nodes** by logic exclusion
  - Mutually-trusted and cooperative nodes can not be attacked by external
  - No defense against attacks from trusted nodes!
- **Openness & Decentralization**
  - Support new and unknown but identifiable nodes
  - Support user-individual sets of trusted nodes, defining each user's trusted virtual topology.
    - Allows unrestricted combination trust groups (overlapping and excluding group membership)
  - No central registry or orchestration
- **Scalability:** Keep protocol overhead within capacities of common CN router hardware

Basic approach and assumptions

- Basic idea: Let each node **dictate its trusted nodes to discard topology-sensitive information from non-trusted nodes**
  $\Rightarrow$ Routes establish only along trusted nodes
  $\Rightarrow$ Own traffic forwarded only along trusted nodes
- **Traffic owner given by packet's destination address**
  - Using identity-proving cryptographically-generated addresses (CGAs) for collision avoidance
- **Trust assessment out of scope!** Considerable options
  - Real-life community
  - Social networks
  - Public-key server (network of trust)
  - Reputation system (individually tuned)
- **Virtual topology** of node *X* given by verified links between trusted nodes of *X*.

Outline

1 Introduction

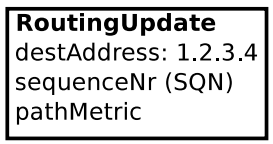
2 Protocol Overview

3 Protocol Messages

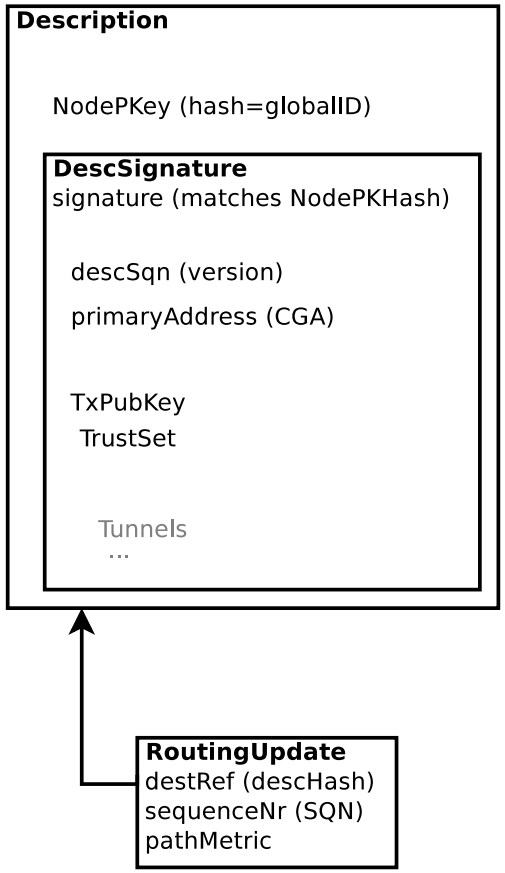
4 Integration and Validation

5 Conclusion and Appendix

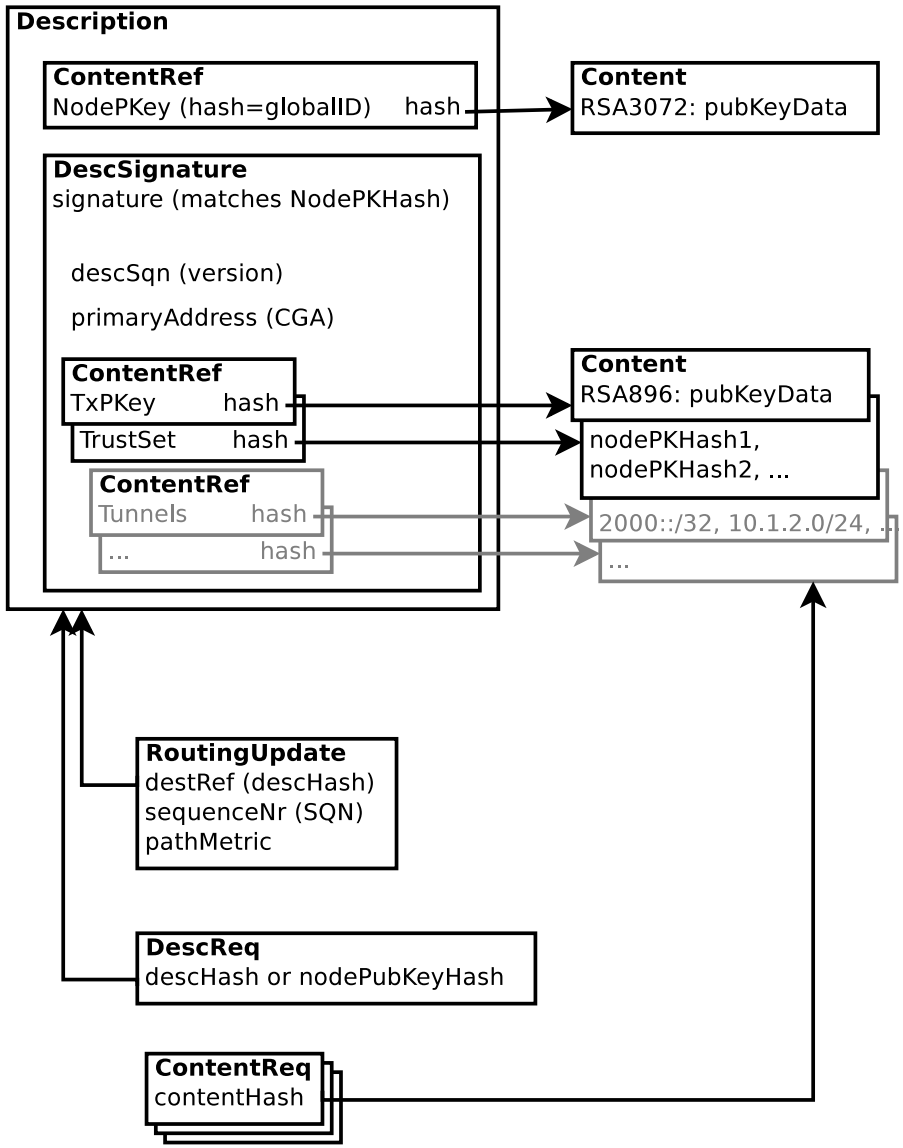
- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



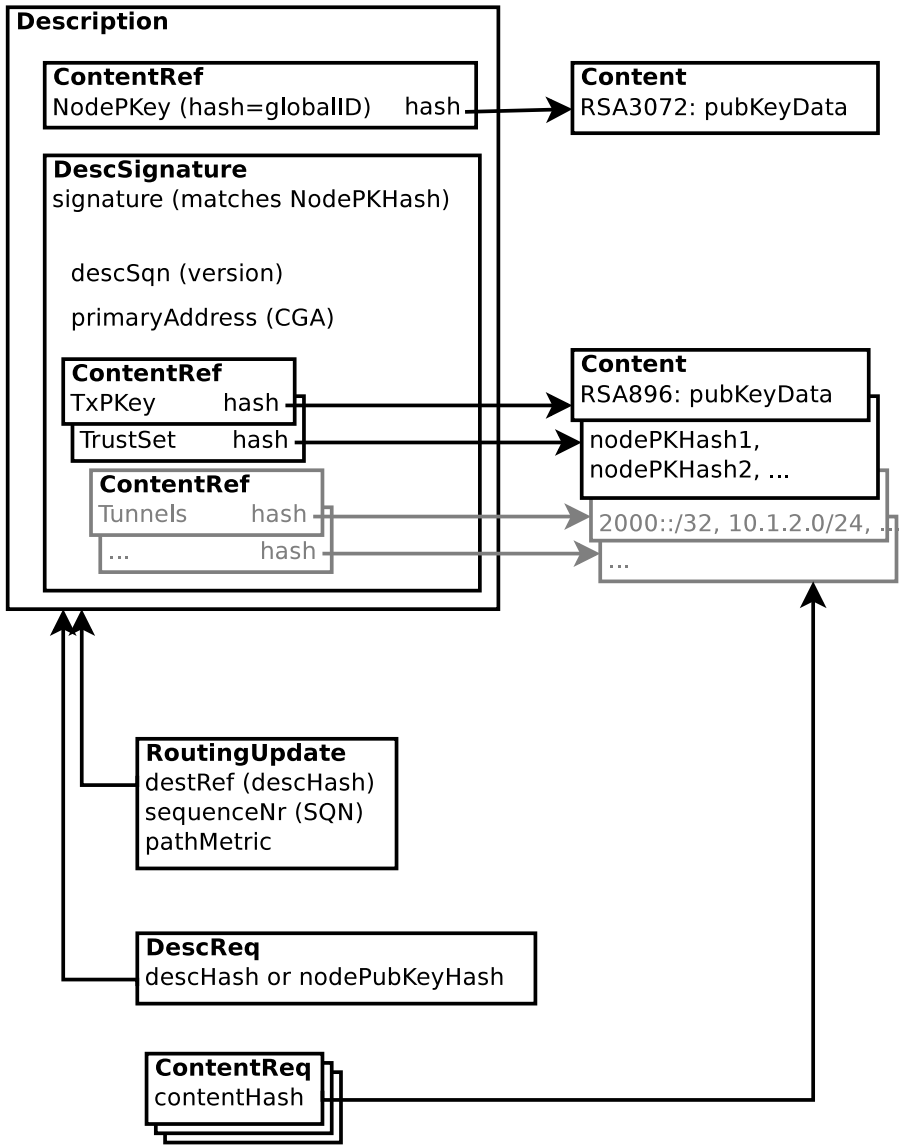
- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



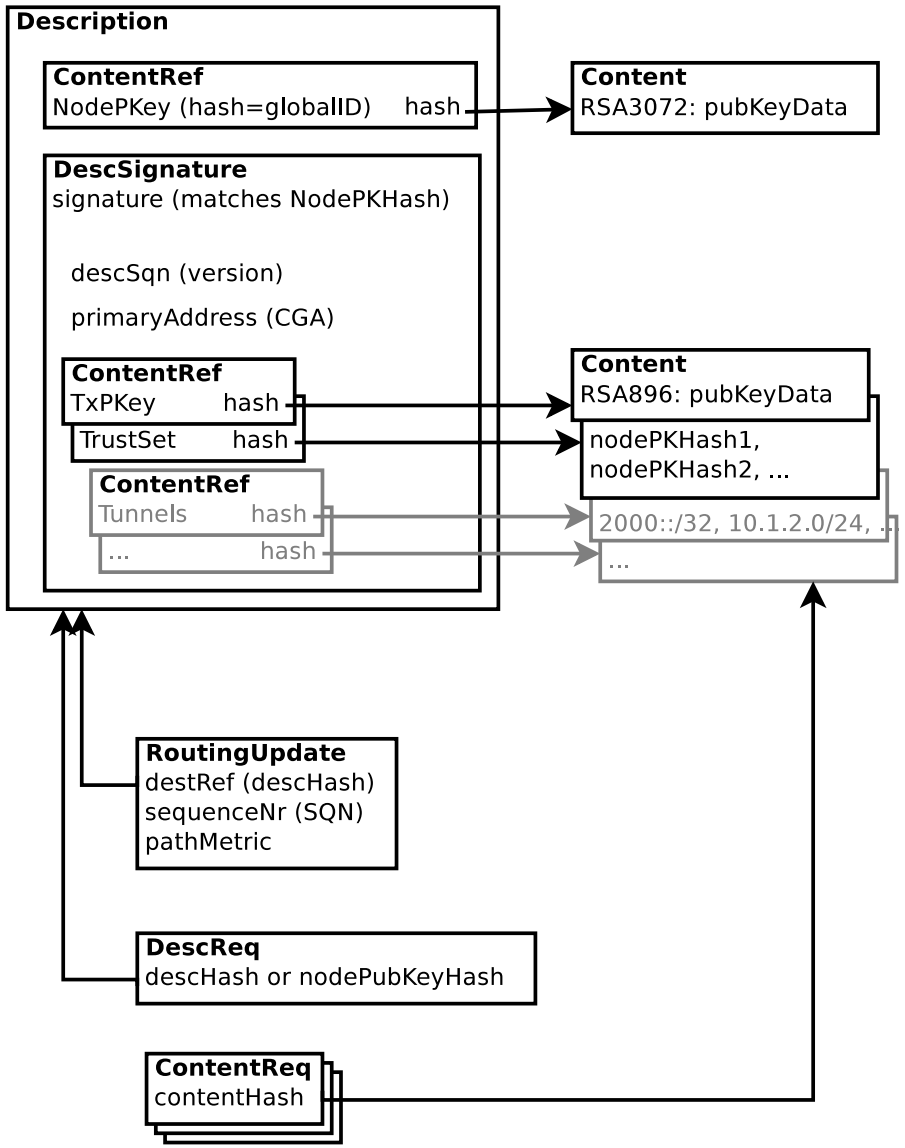
- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



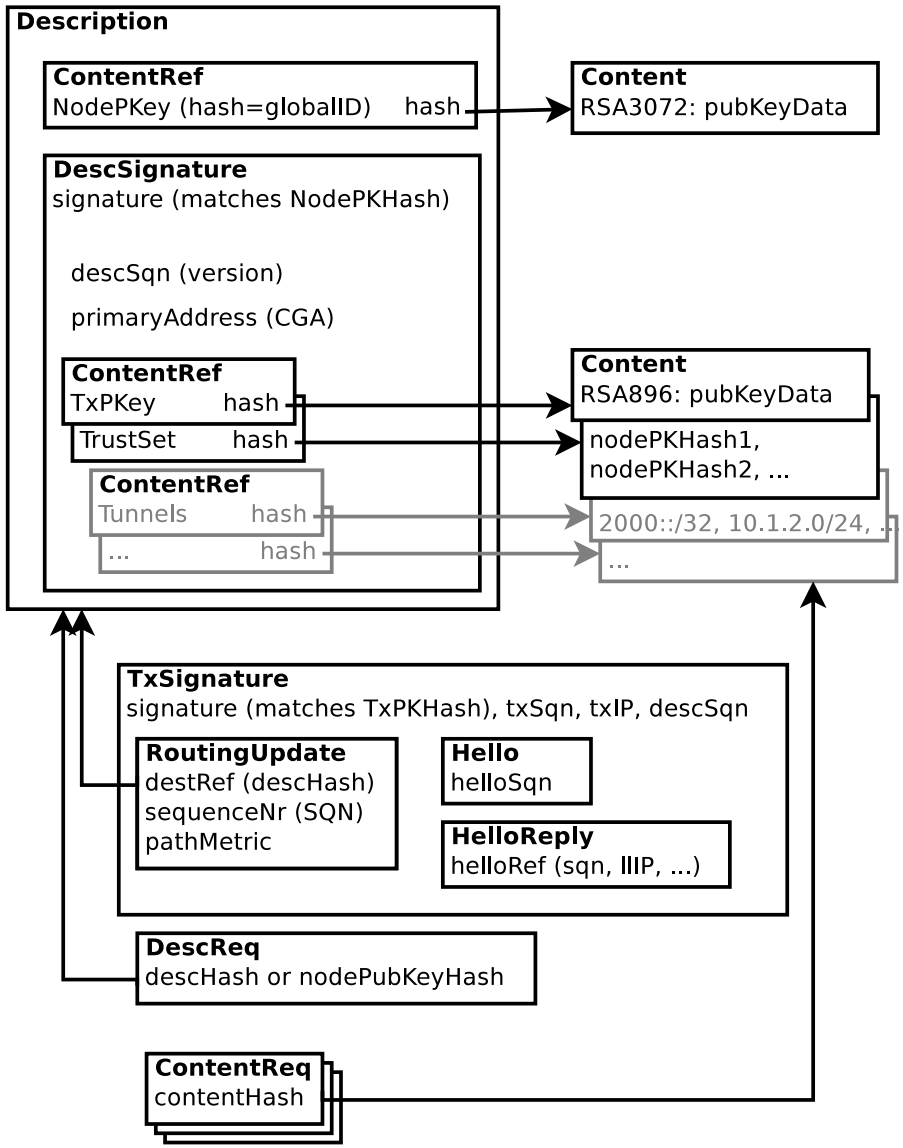
- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



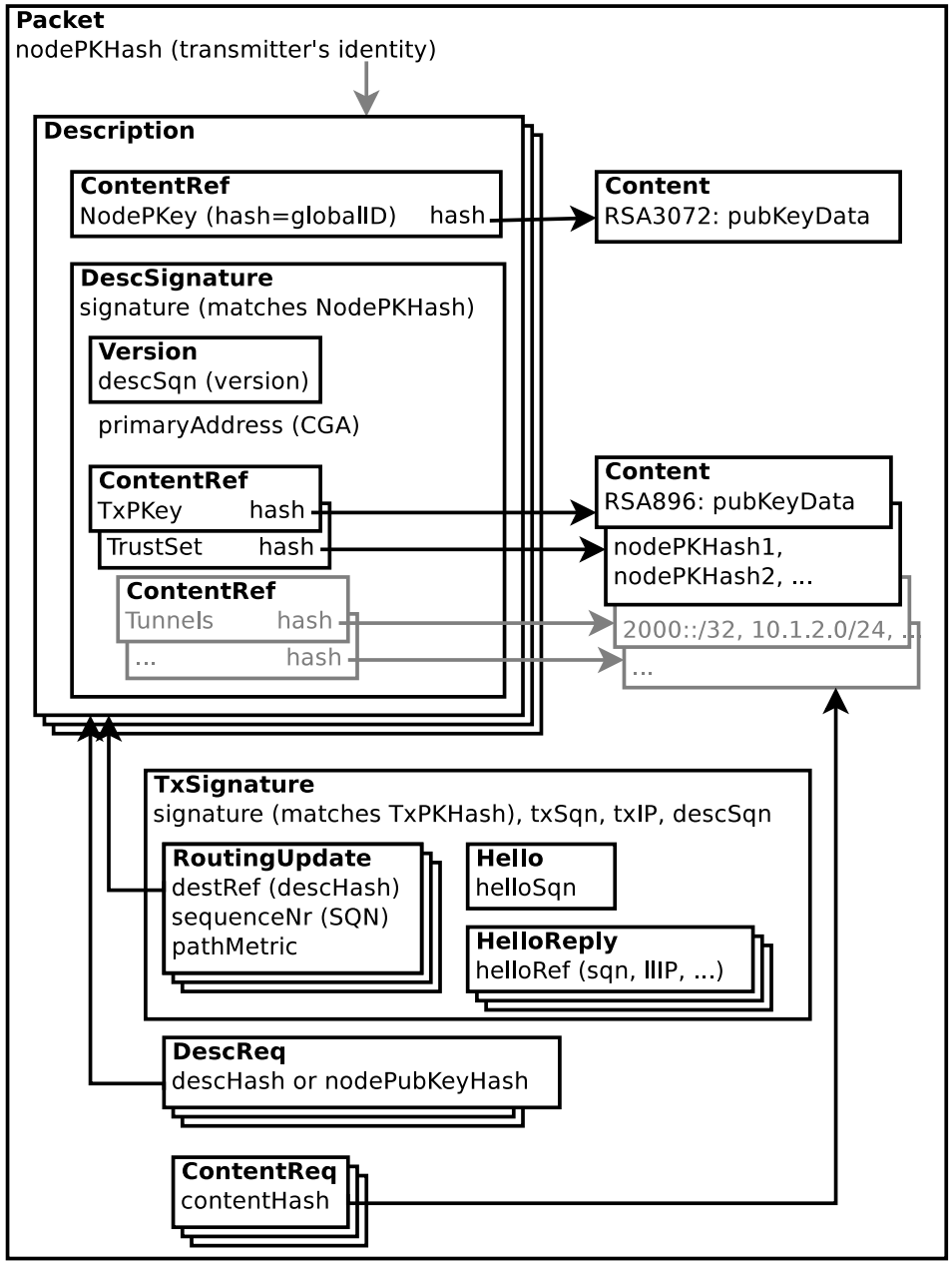
- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



- Basis: Destination-sequenced distance-vector routing
- RoutingUpdate references **node description** (via descHash)
- Description and heavy content requested on demand
  - **node ID** (hash of nodePKey)
  - **Permanent public key** (nodePKey)
  - **Signature** (self-signed)
  - **Address** (identity proving CGA)
  - Description **version**
  - List of **trusted nodes**, indicating eligible neighbors for propagating routing updates
  - Replaceable, weak, public key (TxPKey)
- TX signature for continuous link verification, using lightweight TxPKey



Outline

1 Introduction

2 Protocol Overview

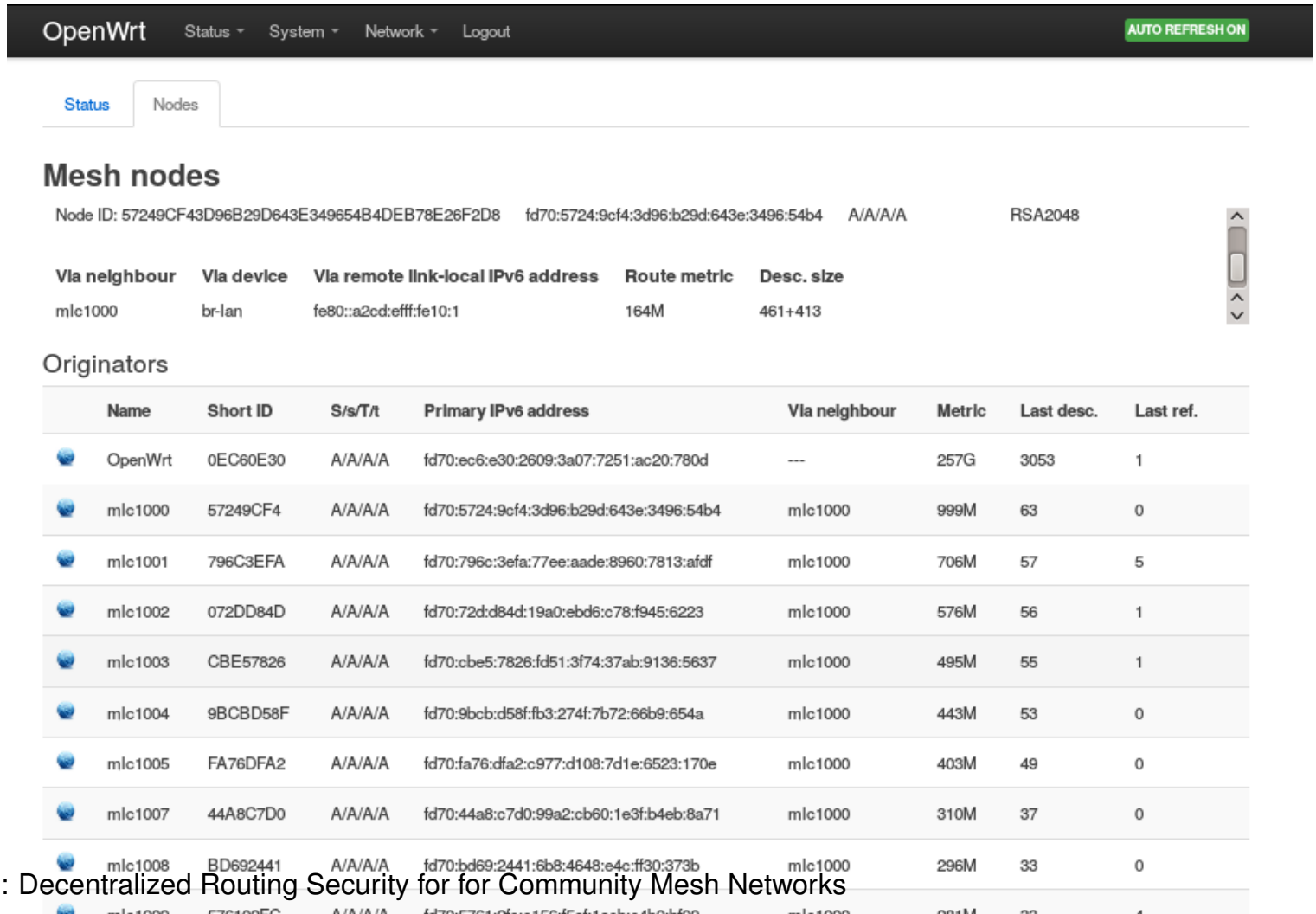
3 Protocol Messages

4 Integration and Validation

5 Conclusion and Appendix

First Integrations in existing firmwares

- qMp
- Libre mesh



Validation

- **Open-source implementation** (bmx6-based)
- Real **embedded target device** (typical hardware for CNs)
- Stressed with **protocol traffic generated by emulated network**
- SEMTOR implementation running in real and virtual nodes

Table 1 : HW and OS characteristics of used target device

Characteristic	Details
Type / CPU	TP-Link TL-WR703N, Atheros AR7240@400 MHz
Wireless	AR9331, 802.11bgn 150 Mbps @100 mW
Flash / Memory	4 MB / 32 MB
Ports	100 MBit Ethernet, USB 2.0
Power supply	5 V, 100 mA, 0.5 W
Cost	approx 10 Euro
OS and distro	Linux OpenWrt (v15.05, r46943)
Further reading	http://wiki.openwrt.org/toh/tp-link/tl-wr703n
Routing	BMX6 semtor branch, git rev 2fb169f
Libraries	PolarSSL version 1.3.4

Validation

- **Open-source implementation** (bmx6-based)
- Real **embedded target device** (typical hardware for CNs)
- Stressed with **protocol traffic generated by emulated network**
- SEMTOR implementation running in real and virtual nodes

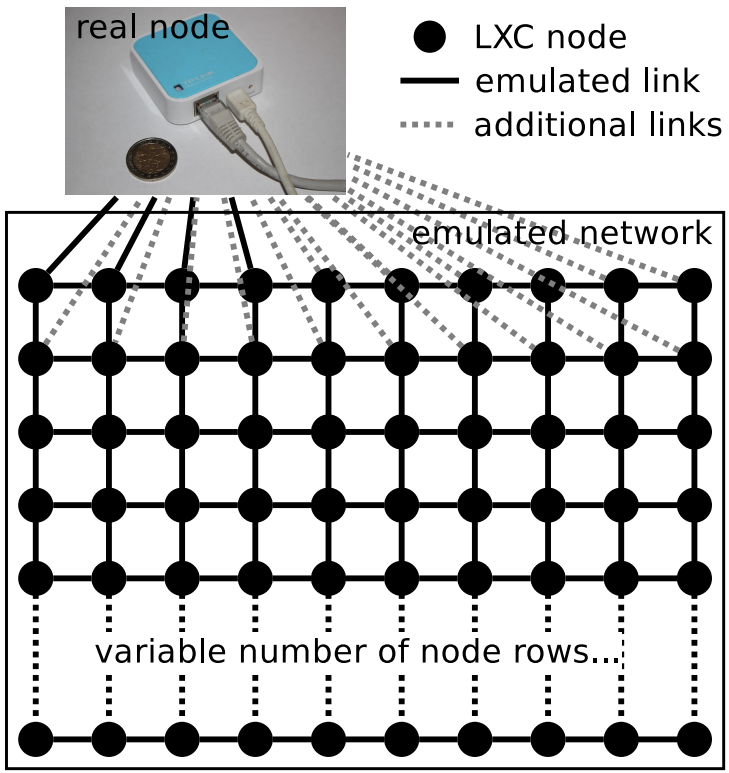


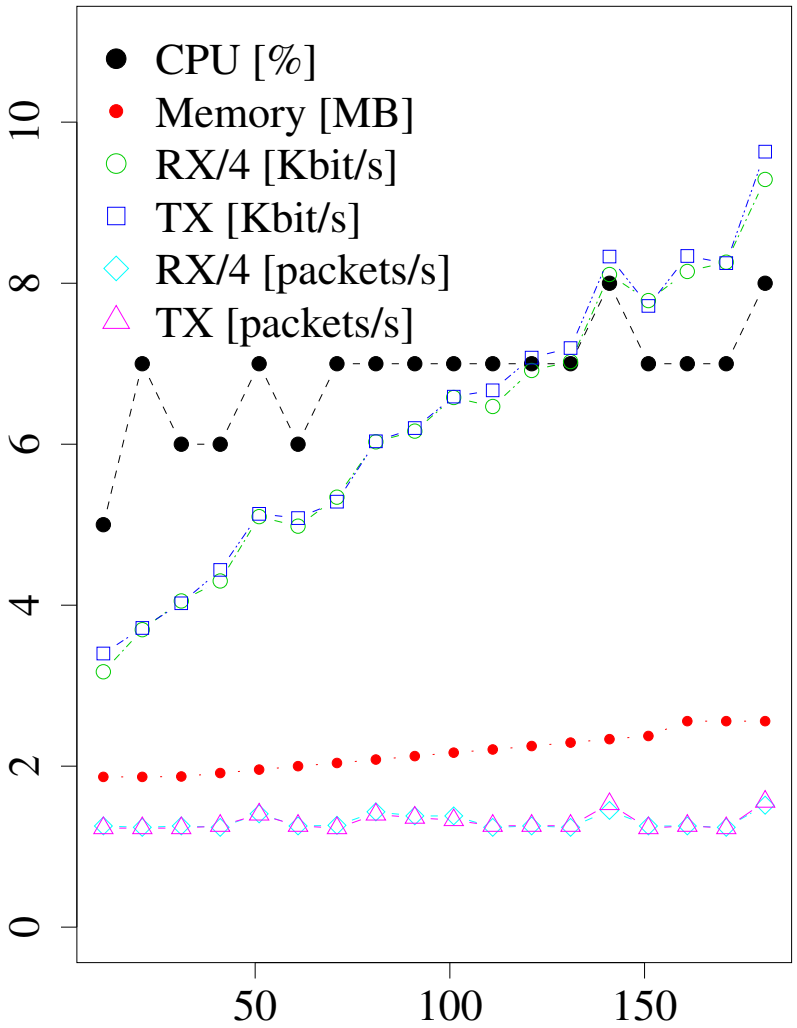
Table 2 : Default parametrization of emulation and protocol

Parameter	Default [range]
Network size (number of nodes)	100 [10..180]
Density (number of links)	4 [4..20]
Node interfaces	1
Grid network structure	10x10 [10x1..10x18]
Link dynamics and loss	static @ zero loss
Primary key strength	RSA3072
TxKey strength	RSA896 [512..1536]
Description-update interval	36000 s [100..4 s]
Routing updates interval	6 s
Link-probing interval	0.8 s
Max message aggregation (TX) interval	0.8 s

Impact of network size

Varying number of nodes

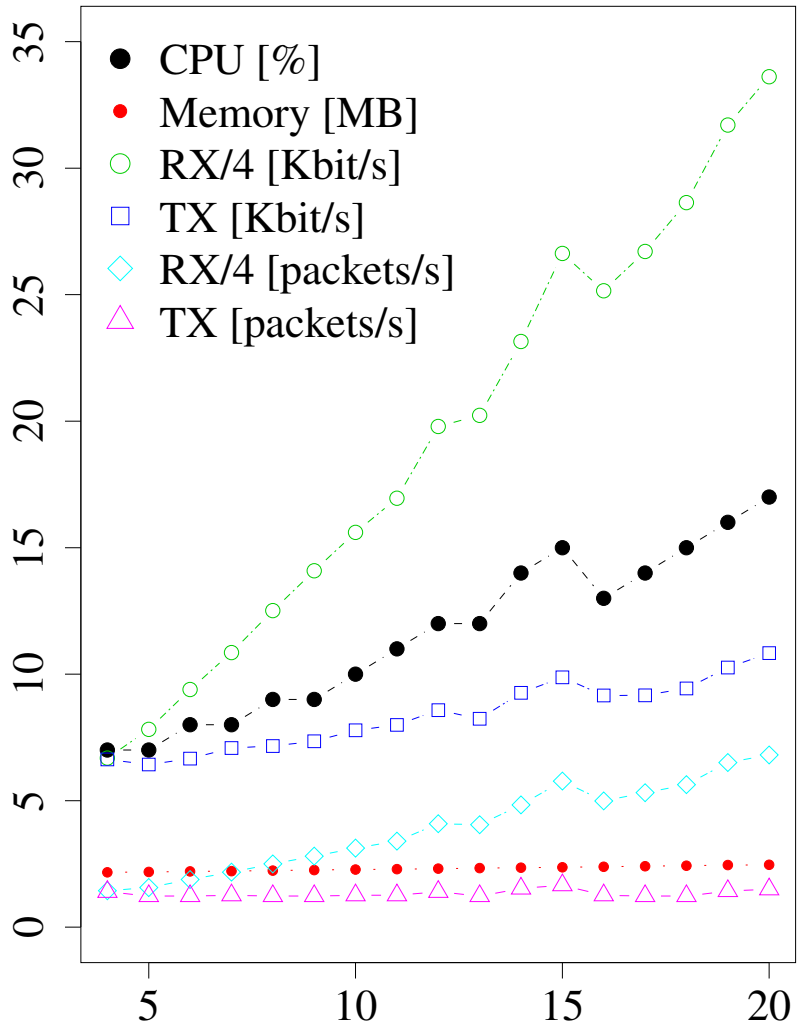
- Linearly increasing CPU, memory, data overhead
- Message aggregation achieves constant packet rate



Impact of network density

Varying number of links with target device

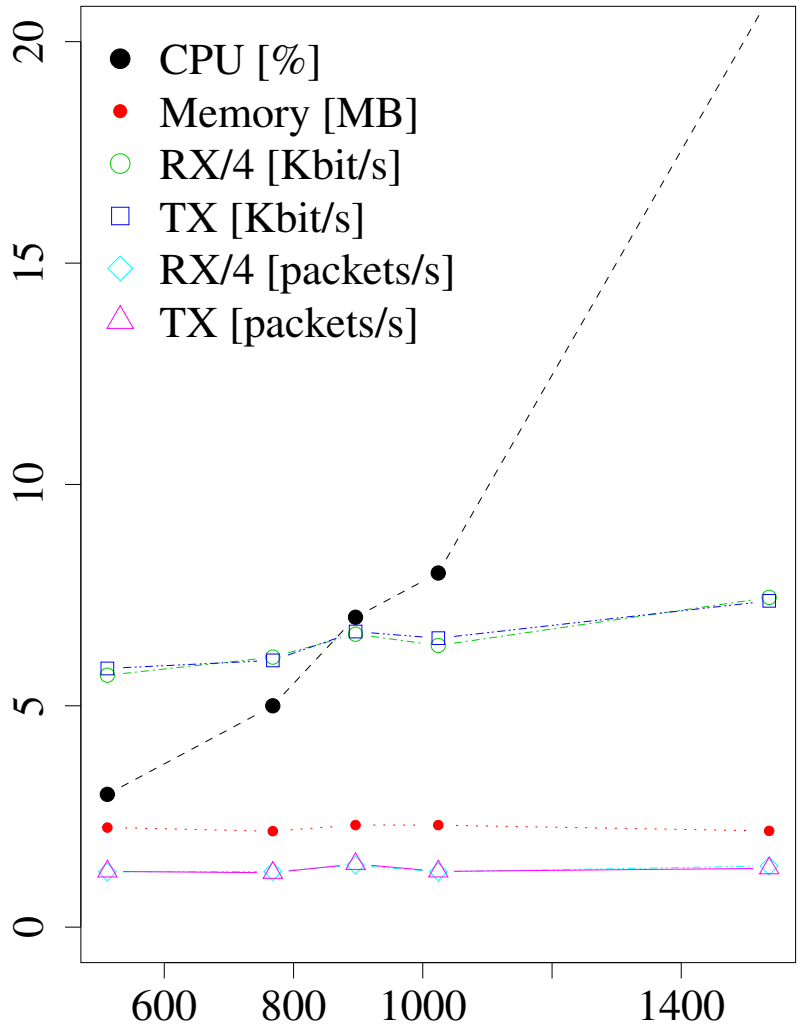
- Linearly increasing CPU and data overhead
- Unaffected memory consumption (memory for description content allocated anyway)



Impact of asymmetric key strength

Varying RSA key length used for link verification

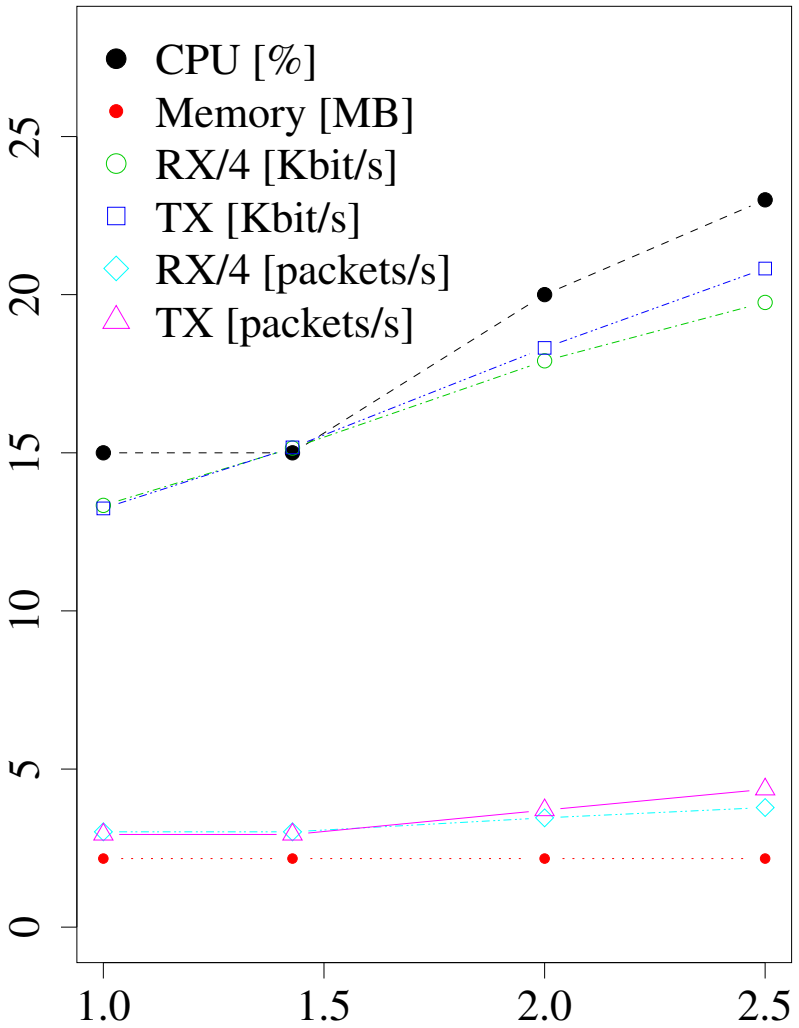
- Linearly increasing data overhead
- Unaffected memory consumption and TX rate
- Exponentially increasing CPU overhead (typical for RSA cryptography)



Impact of description update frequency

Varying total number of updates over time

- Linearly increasing CPU and protocol data overhead
  $\Rightarrow$ Potential bottleneck as node-reconfiguration rate can not be controlled



Outline

1 Introduction

2 Protocol Overview

3 Protocol Messages

4 Integration and Validation

5 Conclusion and Appendix

Conclusion and Outlook

- Findings:
  - Pointed **requirements** for open and decentralized CNs
  - Described mechanisms for **user-individual trusted routing**
  - Validated our approach via **implementation & testing on real embedded hardware**
  - Showed feasibility of **strong asymmetric cryptography** for securing routing-topology while satisfying scalability requirements for typical sized **CN clouds** with 100+ nodes.
  - Identified (based on benchmarking results) **scalability limits** and network-characteristics with significant impact.
- Next:
  - Allow trust import from particular (highly-trusted) nodes
  - Denial of Service attacks... (there are some ideas)

Thank you!

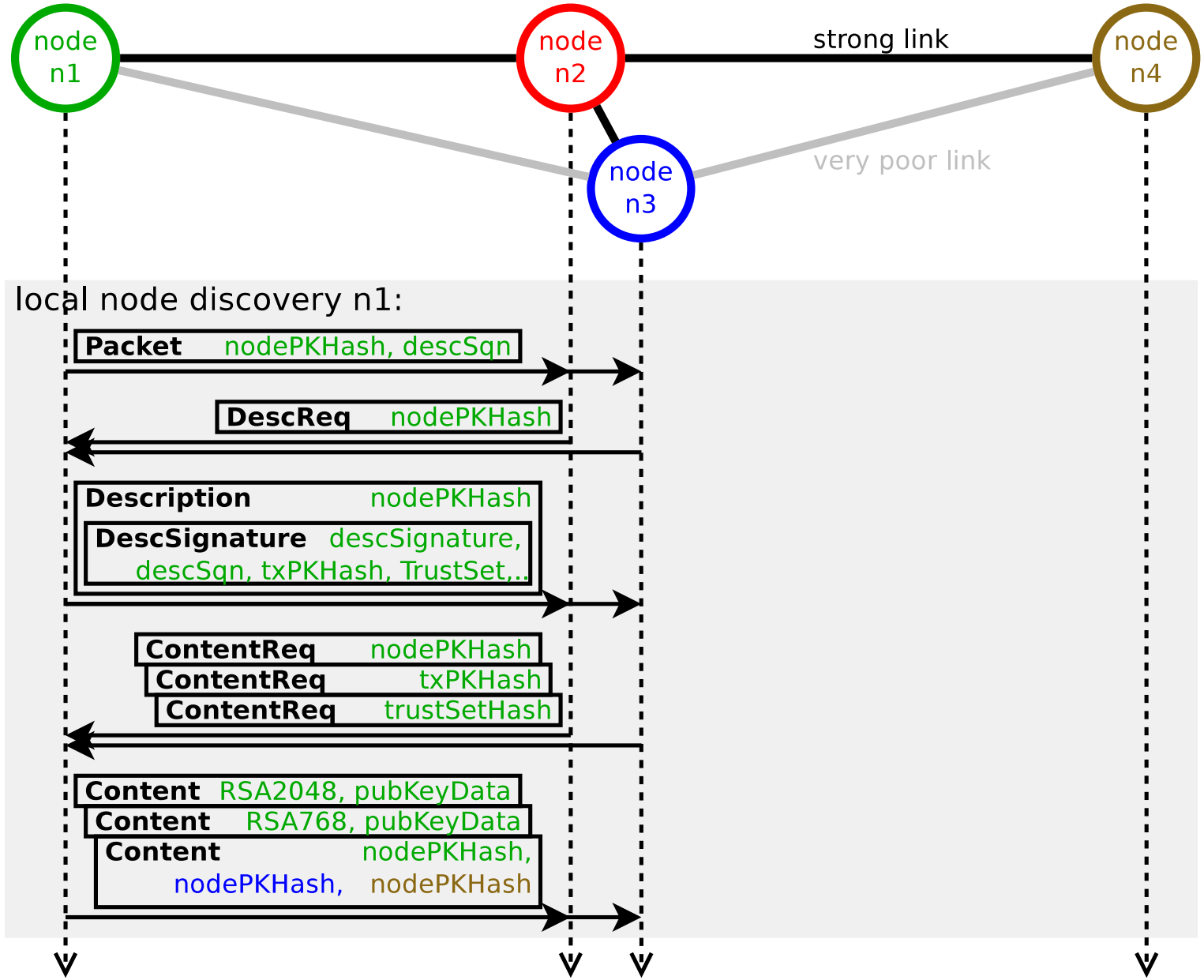
Questions?

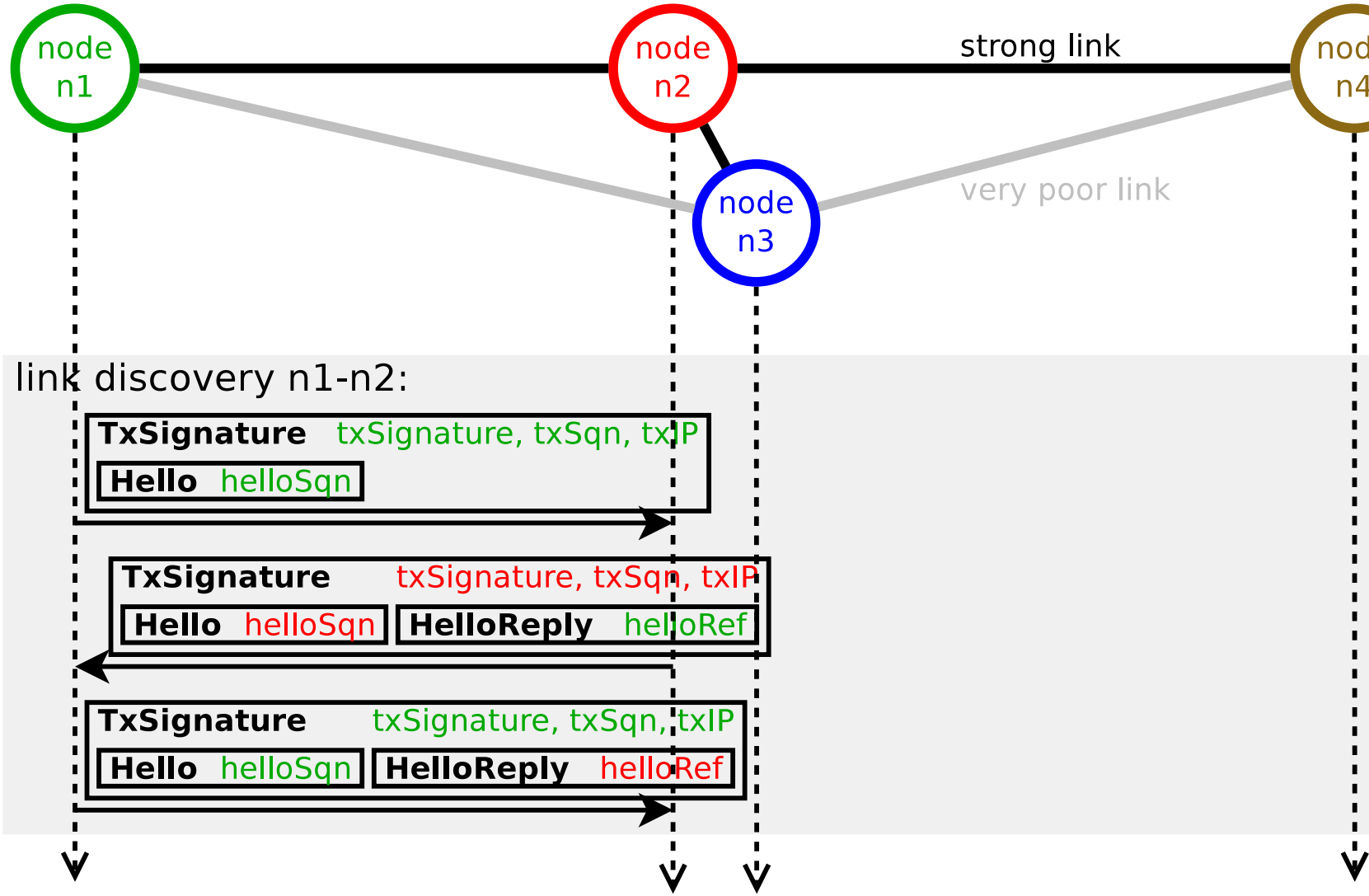
- `http://bmx6.net`
- `https://lists.bmx6.net/cgi-bin/mailman/listinfo/bmxd`

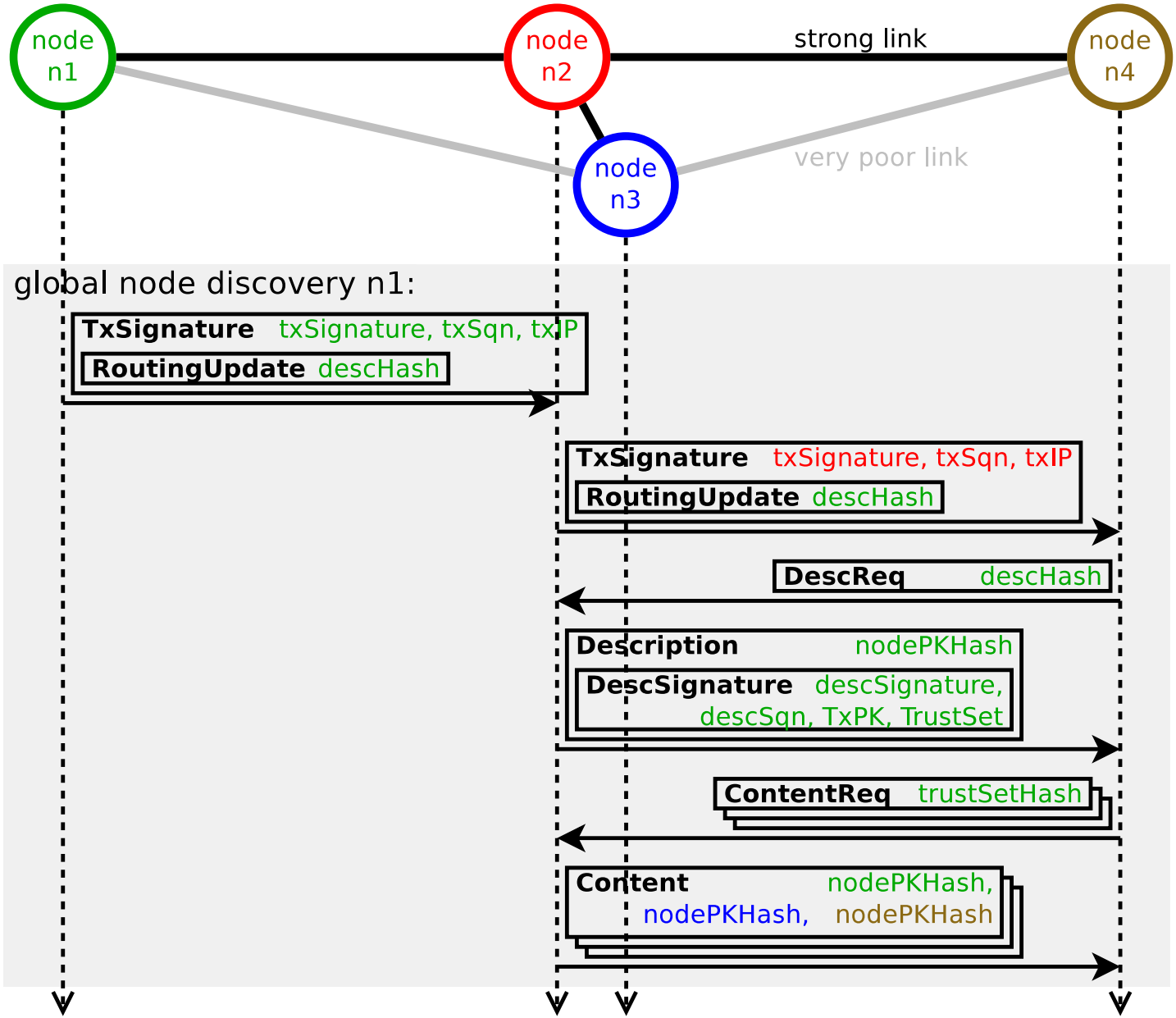
Appendix

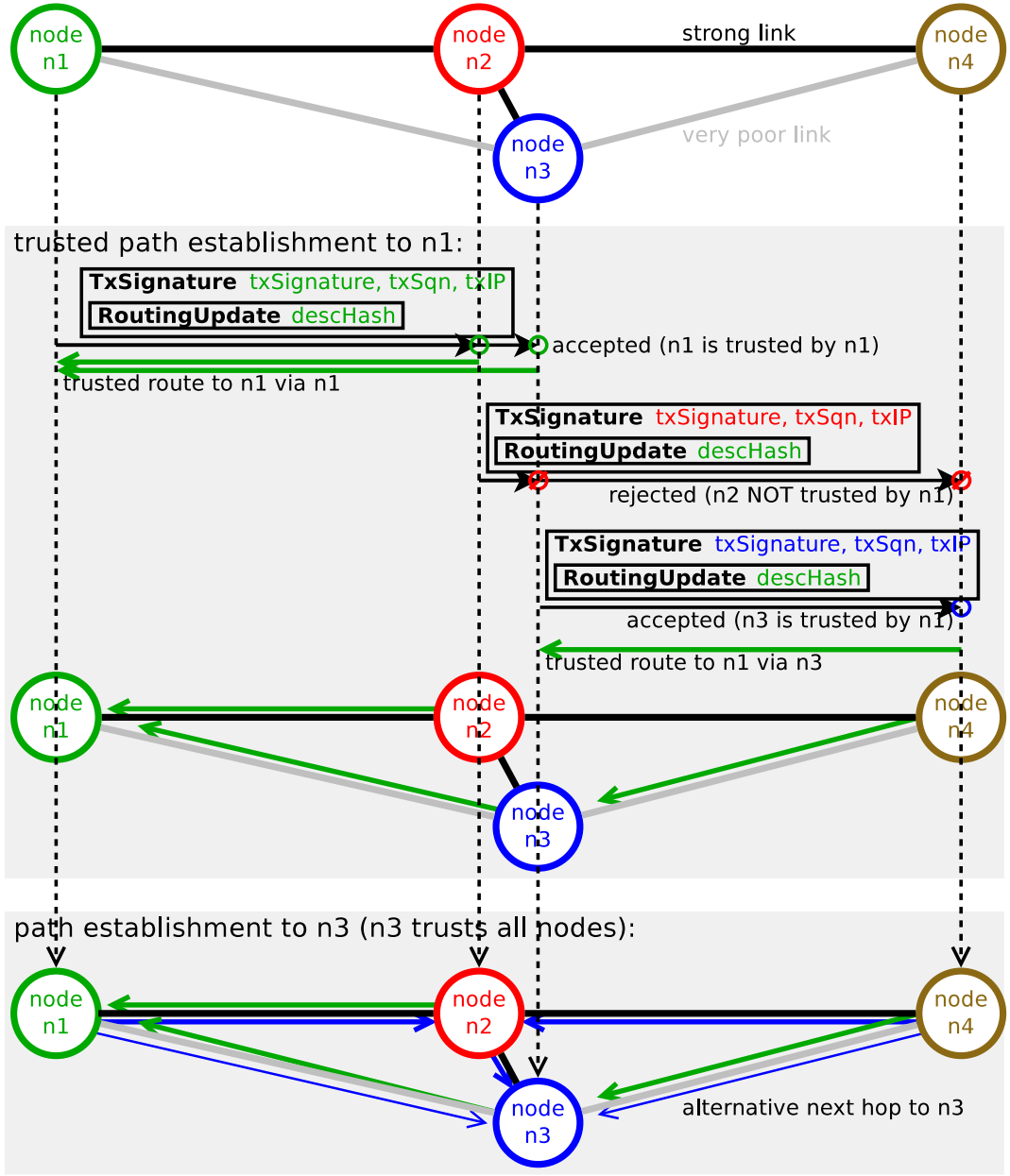
Bibliography

- A. Neumann, E. López, L. Cerdà-Alabern, and L. Navarro, “Securely-entrusted multi-topology routing for community networks”. In: 2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS). Jan. 2016, pp. 1-8.
- L. Cerdà-Alabern, A. Neumann, and L. Maccari. “Experimental Evaluation of BMX6 Routing Metrics in a 802.11an Wireless-Community Mesh Network”. In: 4th International Workshop on Community Networks and Bottom-up- Broadband (CNBuB’2015). Rome, Italy, Aug. 2015.
- A. Neumann, E. López, and L. Navarro. “Evaluation of mesh routing protocols for wireless community networks”. In: Computer Networks 93, Part2 (2015), pp. 308-323, Community Networks, ISSN: 1389-1286
- R. Pueyo, V. Oncins, and A. Neumann. “Enhancing reflection and self-determination in a real-life community mesh network”. In: Computer Networks 93, Part 2 (2015), Community Networks, pp. 297–307. ISSN: 1389-1286.


node n1
node n2
node n3
node n4
strong link
very poor link
local node discovery n1:
Packet nodePKHash, descSqn
DescReq nodePKHash
Description nodePKHash
DescSignature descSignature, descSqn, txPKHash, TrustSet,..
ContentReq nodePKHash
ContentReq txPKHash
ContentReq trustSetHash
Content RSA2048, pubKeyData
Content RSA768, pubKeyData
Content nodePKHash, nodePKHash, nodePKHash


node n1
node n2
strong link
nod n4
node n3
very poor link
link discovery n1-n2:
TxSignature txSignature, txSqn, txIP
Hello helloSqn
TxSignature txSignature, txSqn, txIP
Hello helloSqn
HelloReply helloRef
TxSignature txSignature, txSqn, txIP
Hello helloSqn
HelloReply helloRef


node n1
node n2
node n3
node n4
strong link
very poor link
global node discovery n1:
TxSignature txSignature, txSqn, txIP
RoutingUpdate descHash
TxSignature txSignature, txSqn, txIP
RoutingUpdate descHash
DescReq descHash
Description nodePKHash
DescSignature descSignature, descSqn, TxPK, TrustSet
ContentReq trustSetHash
Content nodePKHash, nodePKHash, nodePKHash


node n1
node n2
node n3
node n4
strong link
very poor link
trusted path establishment to n1:
TxSignature txSignature, txSqn, txIP
RoutingUpdate descHash
accepted (n1 is trusted by n1)
trusted route to n1 via n1
TxSignature txSignature, txSqn, txIP
RoutingUpdate descHash
rejected (n2 NOT trusted by n1)
TxSignature txSignature, txSqn, txIP
RoutingUpdate descHash
accepted (n3 is trusted by n1)
trusted route to n1 via n3
path establishment to n3 (n3 trusts all nodes):
alternative next hop to n3