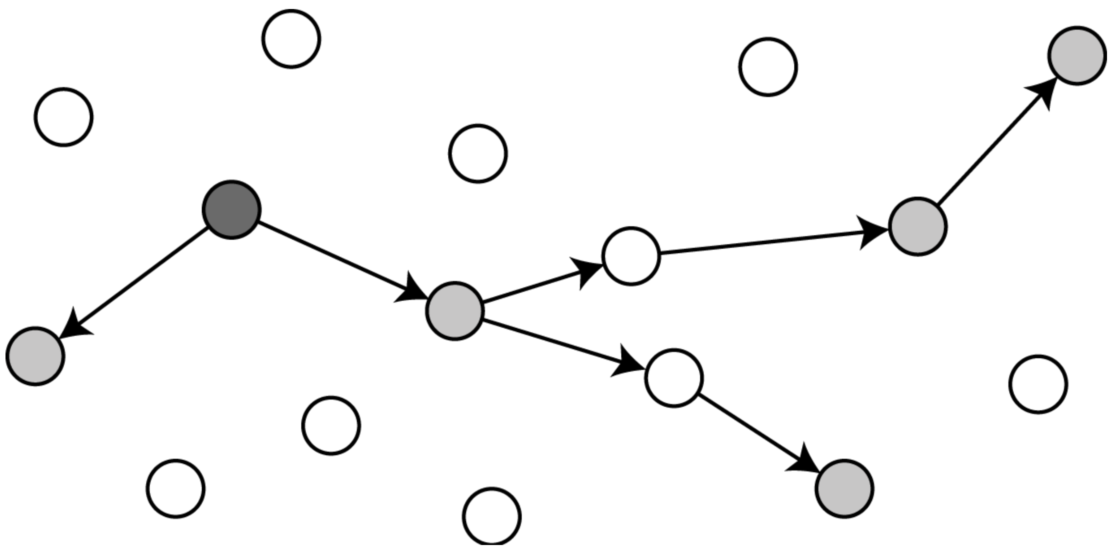
Castor and Xcastor

Secure and Efficient Unicast and Multicast Routing for Mobile Ad Hoc Networks



Milan Schmittner
mschmittner@seemoo.tu-darmstadt.de
https://seemoo.de/mschmittner (PGP key)

SECURE MOBILE NETWORKING

Secure Routing in Mesh Networks

SECURE MOBILE NETWORKING

Challenges:

- **Openness**. Community (Freifunk) or disast…
  (Serval) networks: everyone should b…
  but how to cope with „bad“ guy…
- **Distributed system**. No…

Selected D… …s:

- **PHY/**…
- …
  B… (packet dropping), wormhole, rushing,
  co… …age spoofing (zero distance), …, or a
  com…ation of the above.

tl;dr

Wireless mesh networks are easy targets for DoS attacks

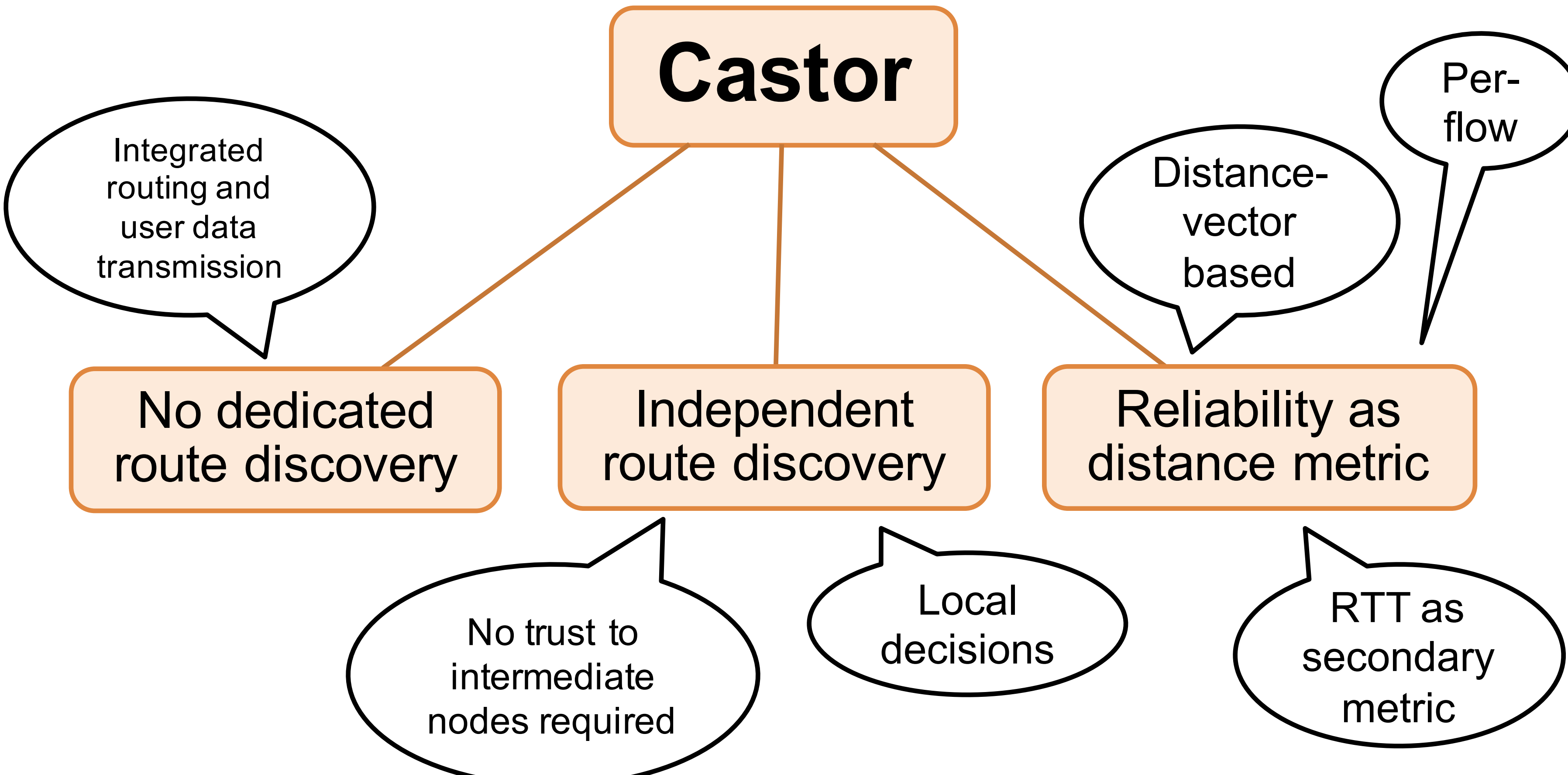
Outline

1. Motivation
2. Castor [1]
    1. Core concepts
    2. Castor Routing by Example
3. Xcastor [2]
4. Conclusion

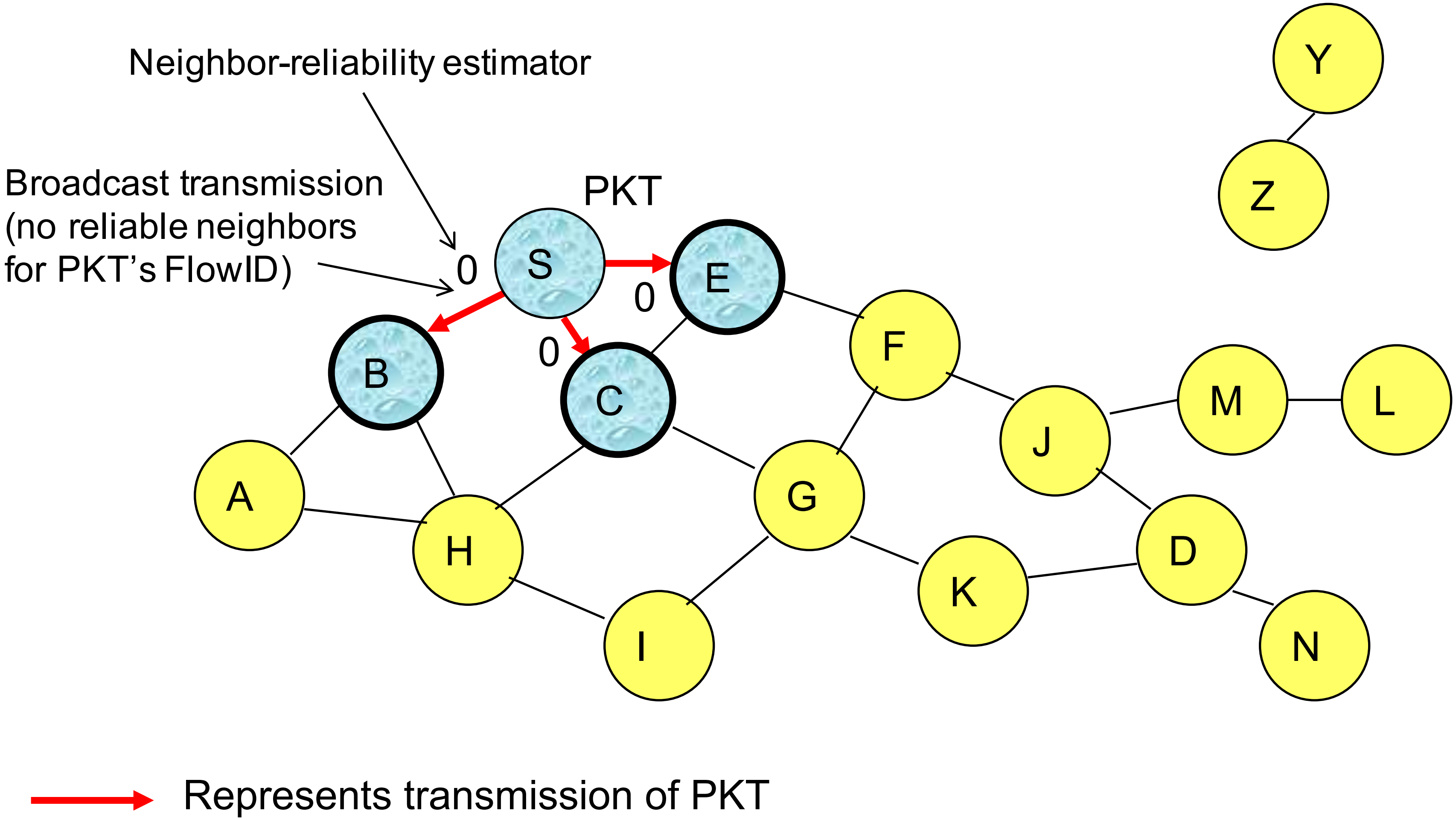
[1] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer, "Castor: Scalable Secure Routing for Ad Hoc Networks," in *Proceedings of the IEEE Conference on Computer Communications*, 2010, pp. 1–9.

[2] M. Schmittner, "Scalable and Secure Multicast for Mobile Ad-hoc Networks," *Master thesis*, Technische Universität Darmstadt, 2014.

Castor: Core Concepts



Castor: PKT Delivery



(s) (d) (H)

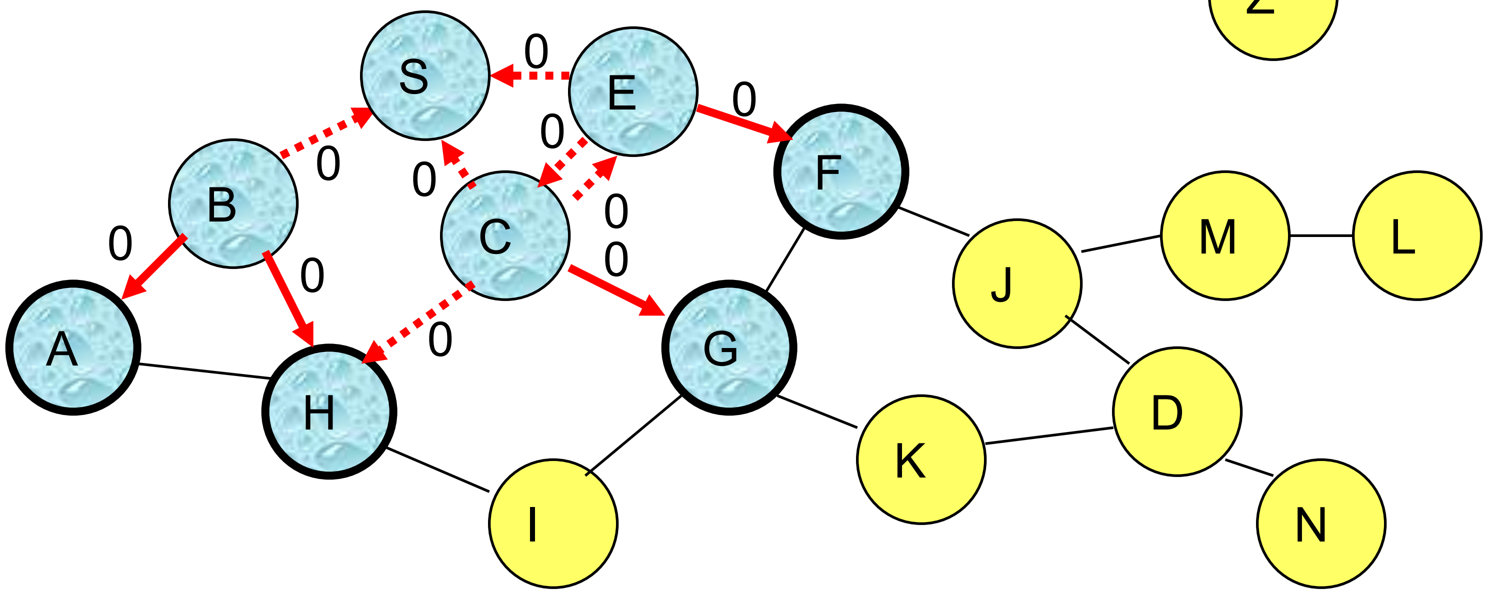
PKT = (S, D, FlowID, PktID, FlowAuth, PktAuth, Msg)

Slide by Michael Noisternig

Castor: PKT Delivery

SEMG SECURE MOBILE NETWORKING

Each receiving intermediate node:
- verifies PktID belongs to FlowID (using FlowAuth)
- caches PKT (simplified) and forwards it



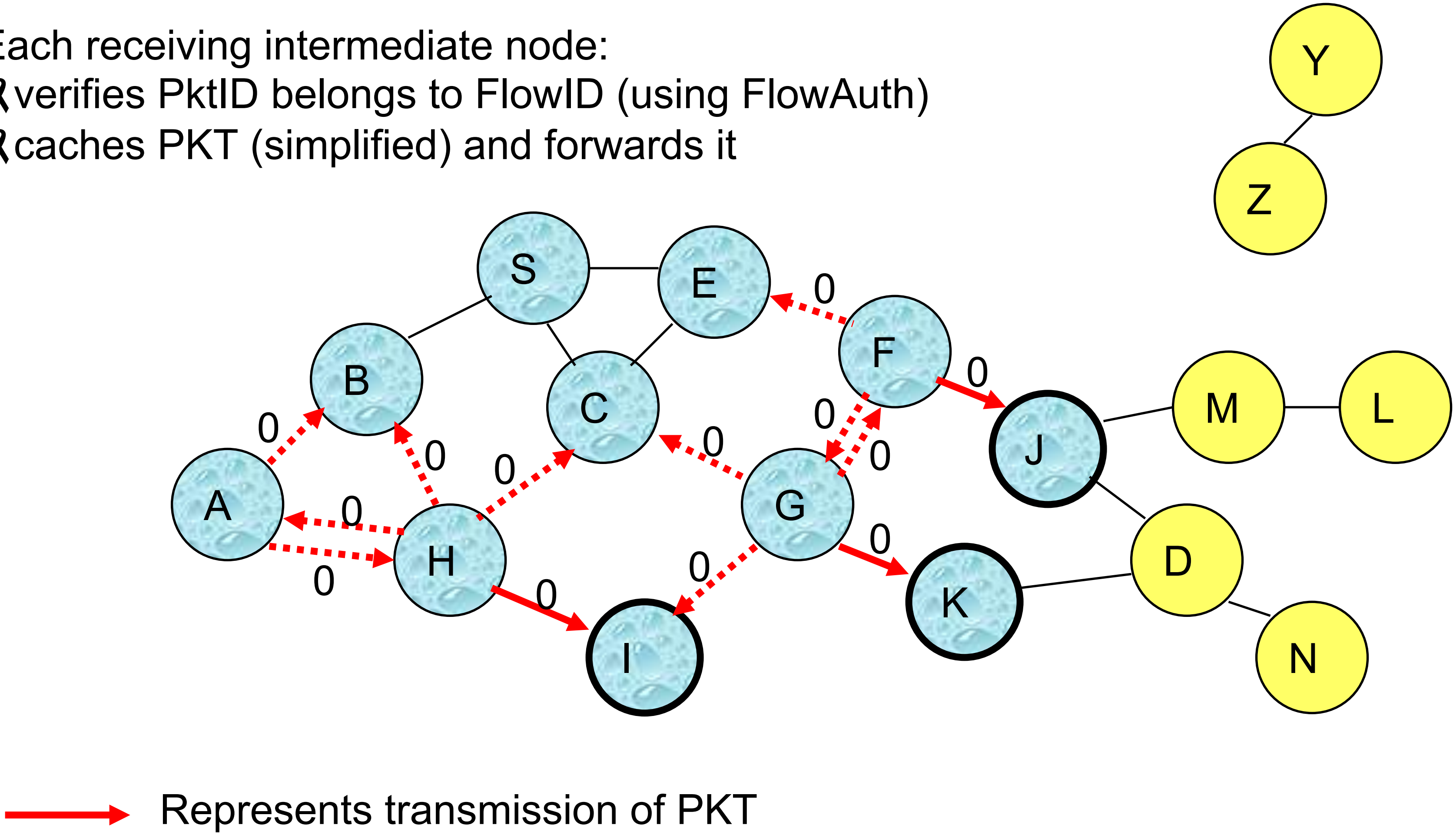
→ Represents transmission of PKT

PKT = (S, D, FlowID, PktID (b_k), FlowAuth (f_k), PktAuth, Msg)

Slide by Michael Noisternig

Castor: PKT Delivery

Each receiving intermediate node:
- verifies PktID belongs to FlowID (using FlowAuth)
- caches PKT (simplified) and forwards it

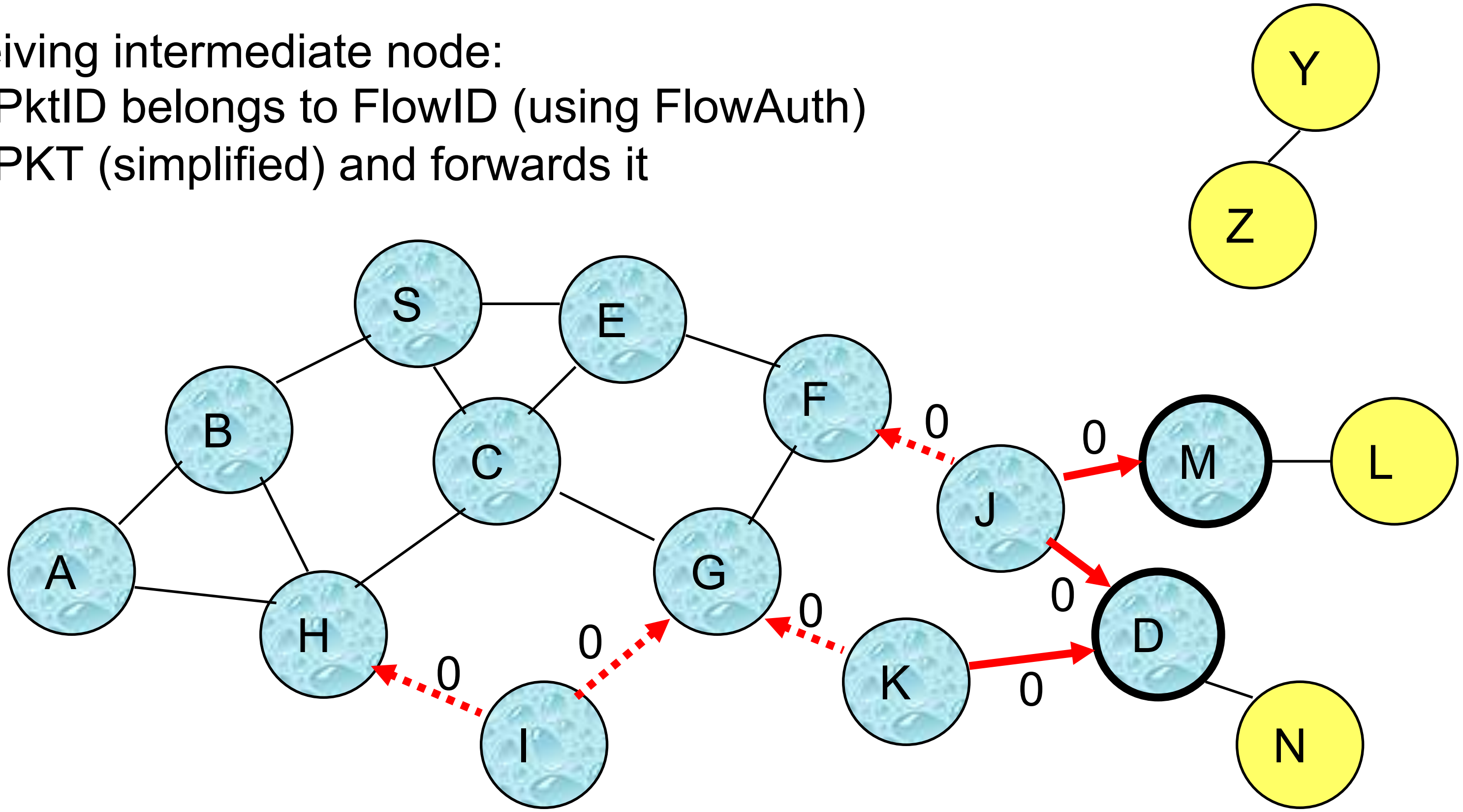


PKT = (S, D, FlowID, PktID, FlowAuth, PktAuth, Msg)

Slide by Michael Noisternig

Castor: PKT Delivery

Each receiving intermediate node:
- verifies PktID belongs to FlowID (using FlowAuth)
- caches PKT (simplified) and forwards it



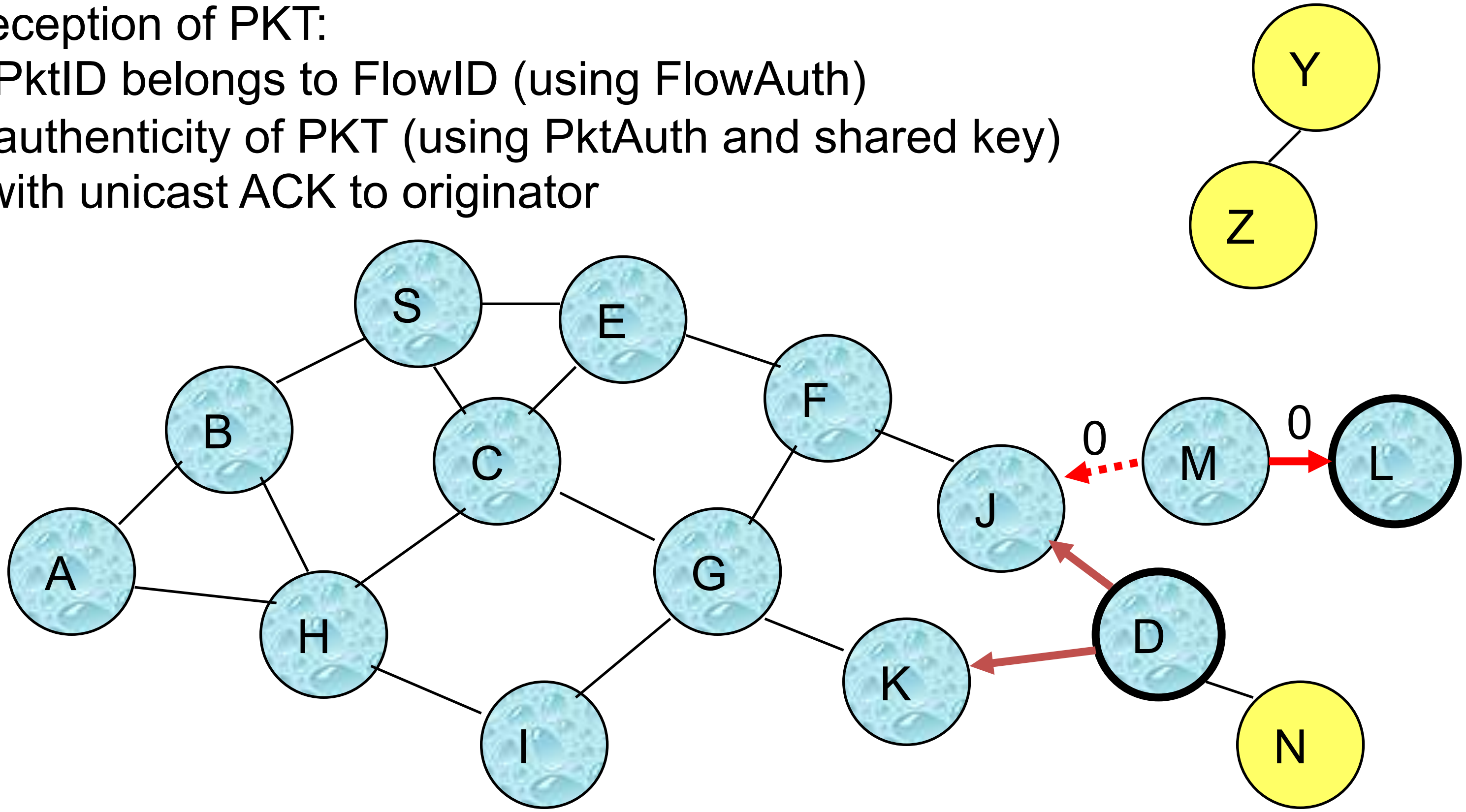
Represents transmission of PKT

PKT = (S, D, FlowID, PktID, FlowAuth, PktAuth, Msg)

Slide by Michael Noisternig

Castor: ACK Delivery

D, upon reception of PKT:
- verifies PktID belongs to FlowID (using FlowAuth)
- verifies authenticity of PKT (using PktAuth and shared key)
- replies with unicast ACK to originator



Represents transmission of ACK

ACK = (AckAuth) ← (a_k)

(e_k)

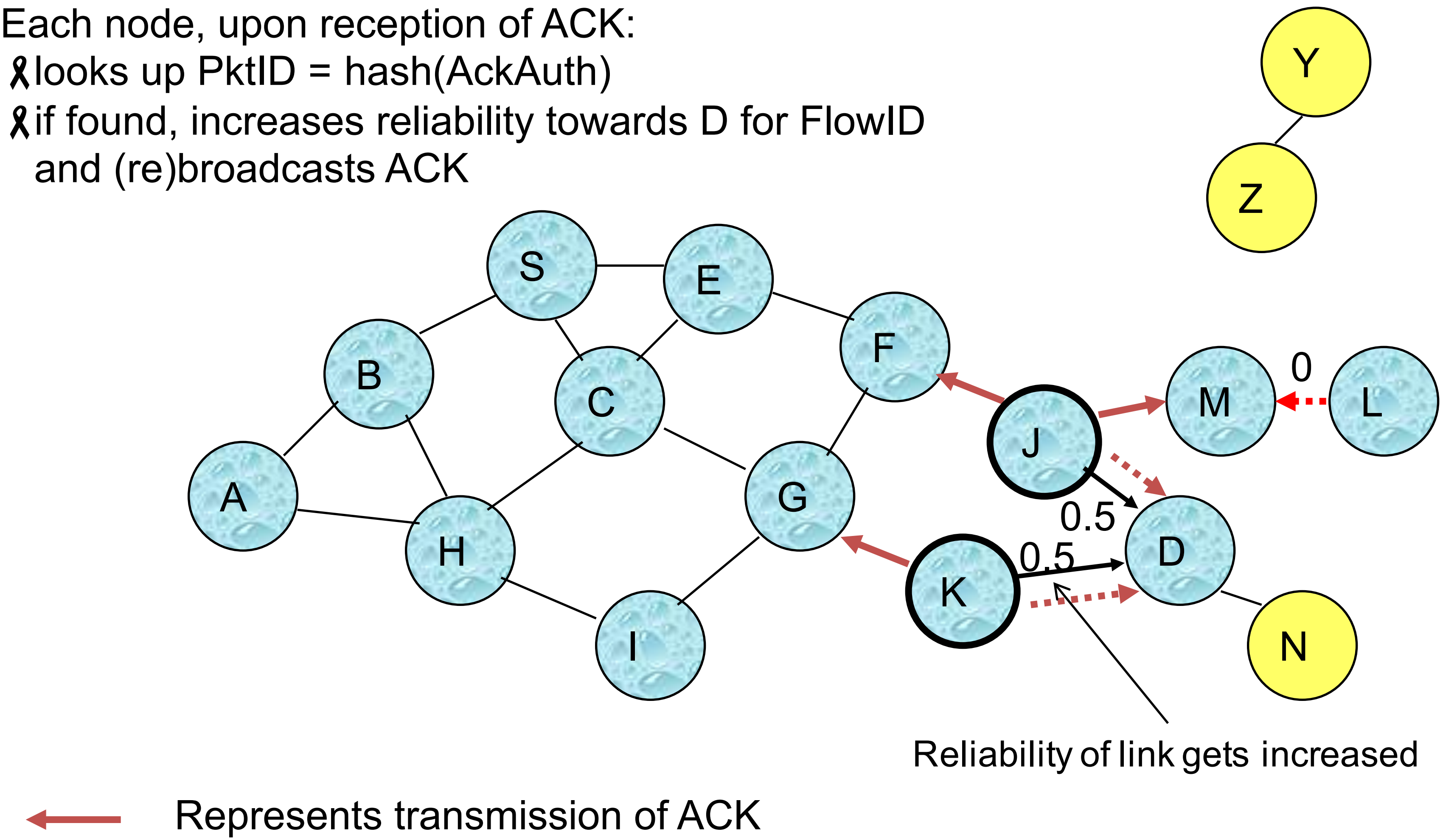
PKT = (S, D, FlowID, PktID, FlowAuth, PktAuth, Msg)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:

- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK

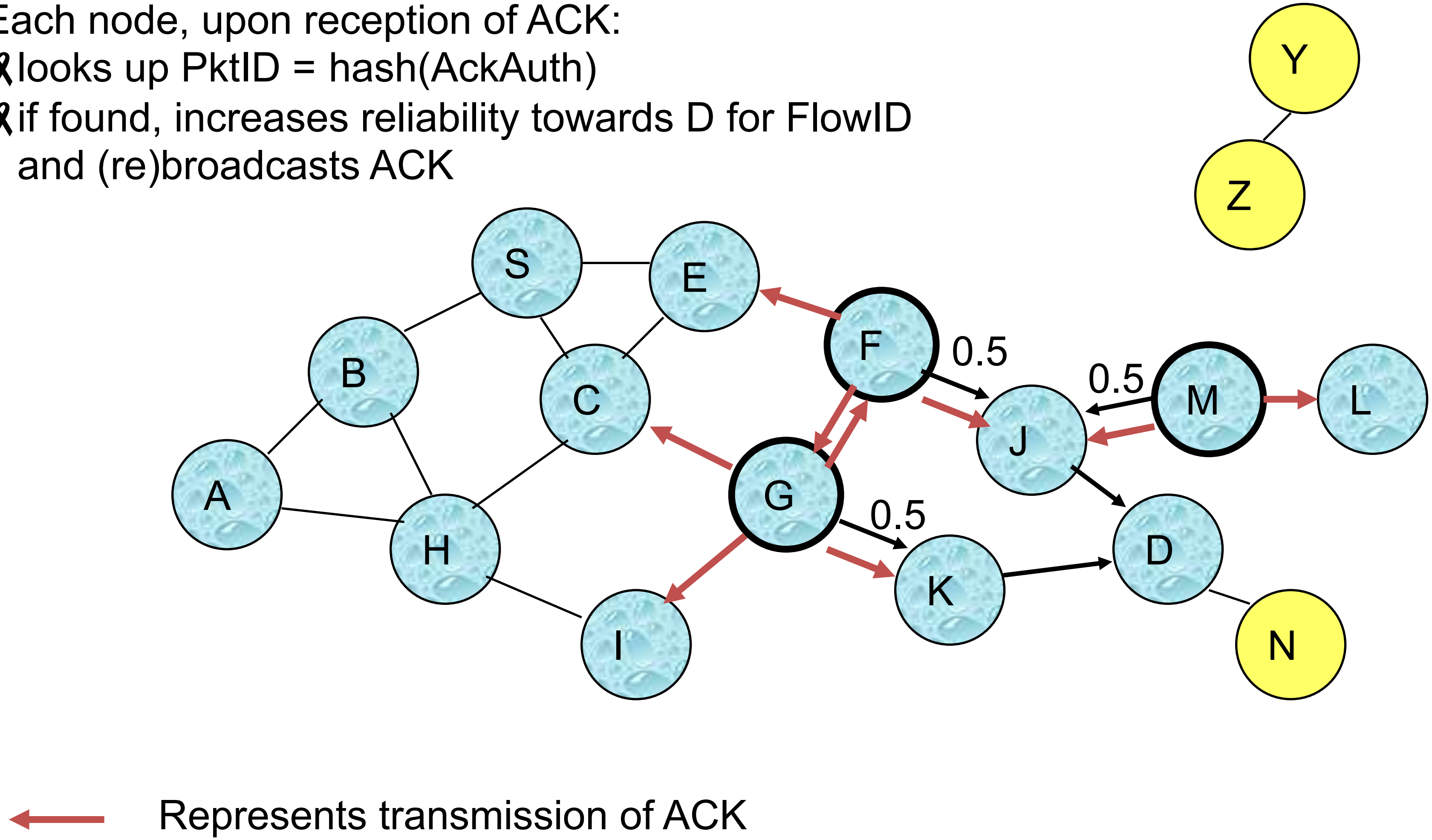


ACK = (AckAuth)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:
- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK

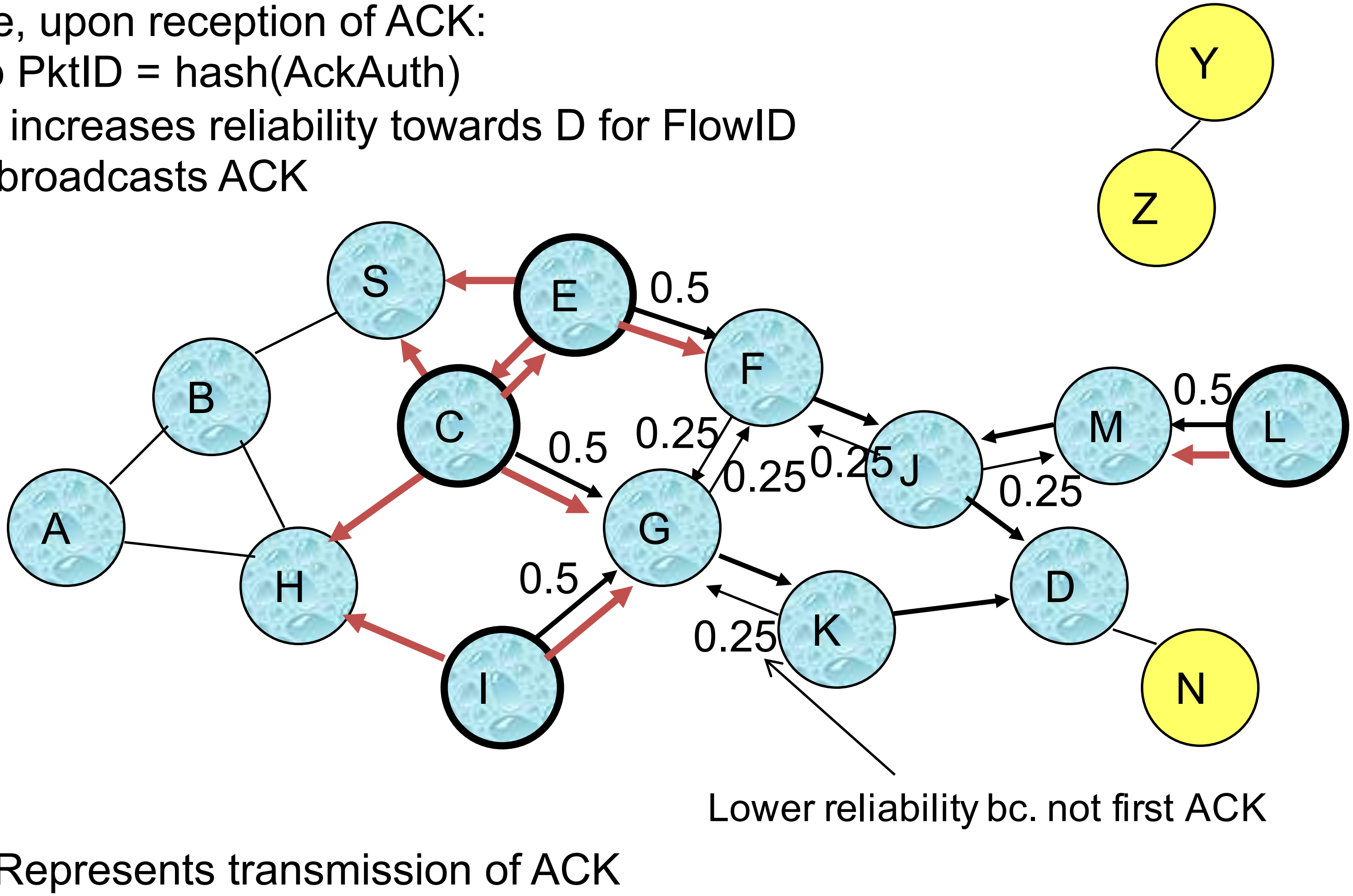


ACK = (AckAuth)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:
- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK



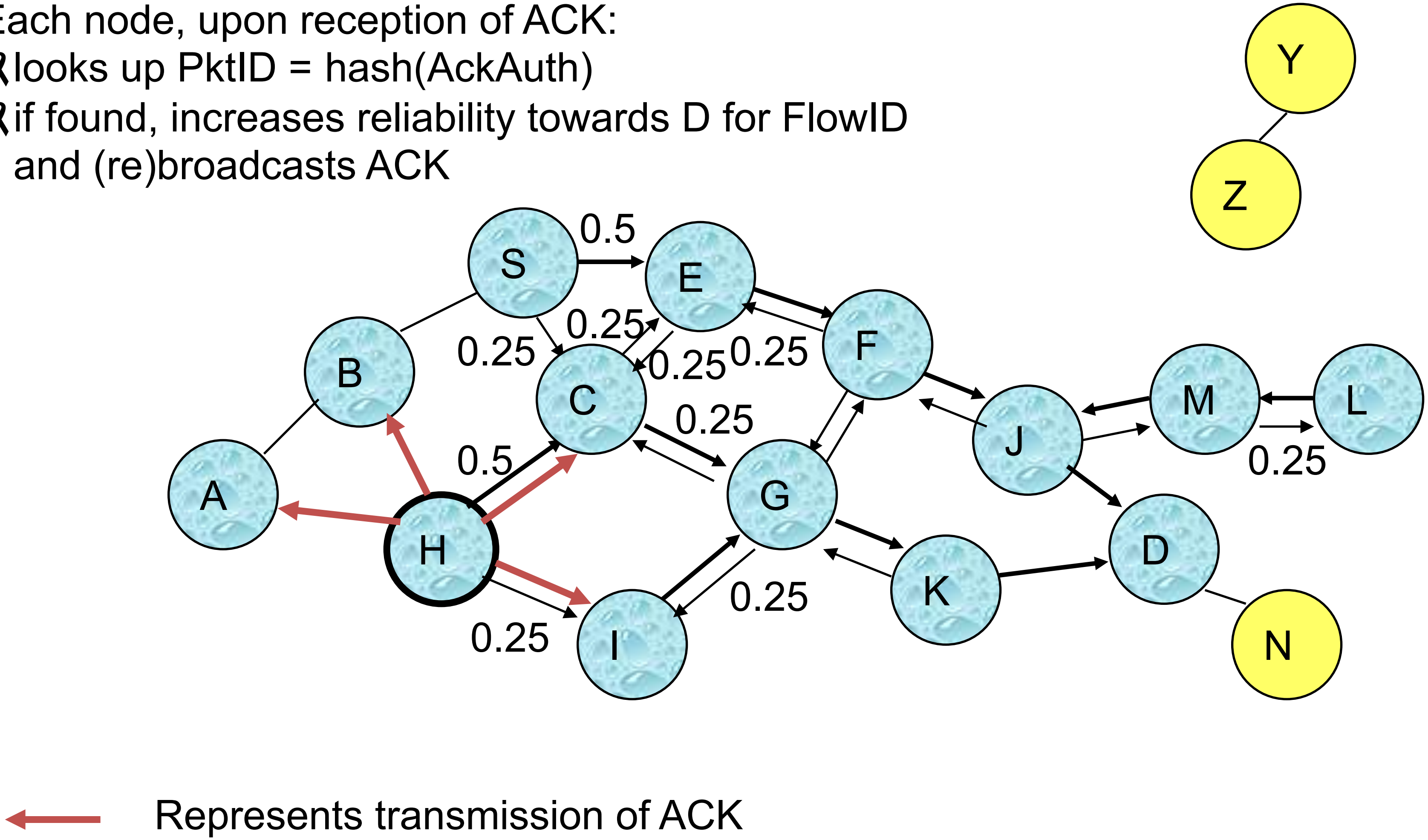
ACK = (AckAuth)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:

- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK

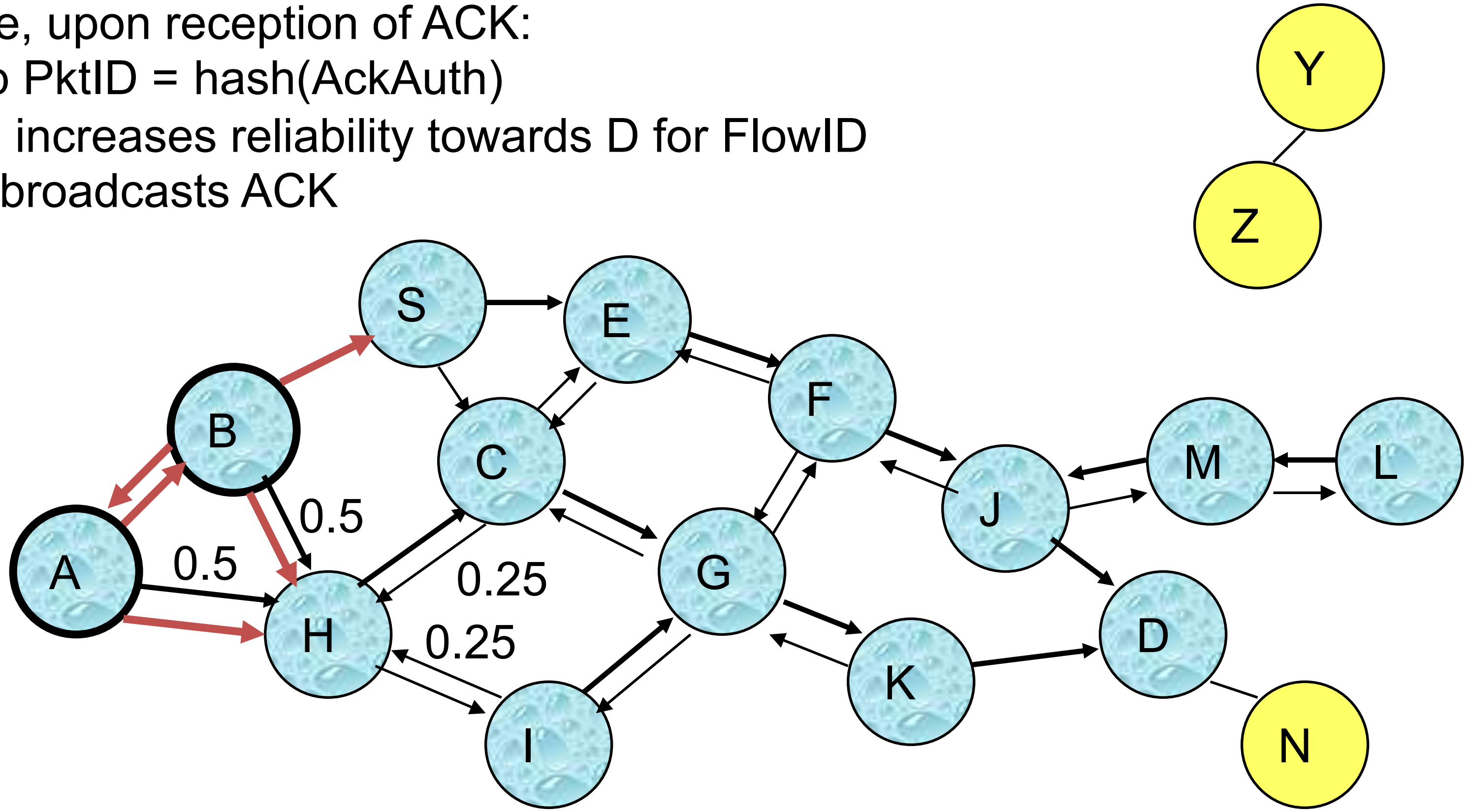


ACK = (AckAuth)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:
- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK




Represents transmission of ACK

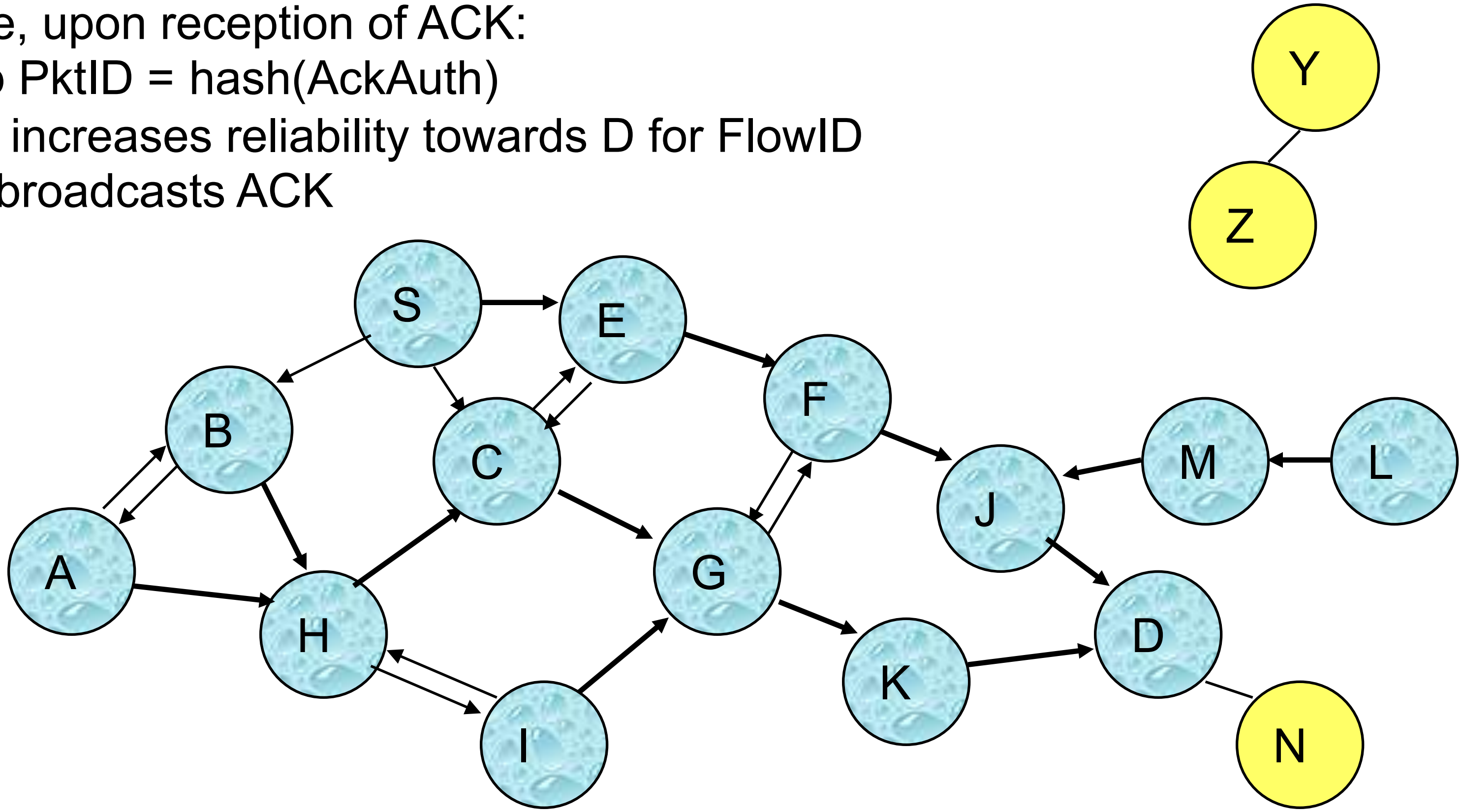
ACK = (AckAuth)

Slide by Michael Noisternig

Castor: ACK Delivery

Each node, upon reception of ACK:

- looks up PktID = hash(AckAuth)
- if found, increases reliability towards D for FlowID and (re)broadcasts ACK



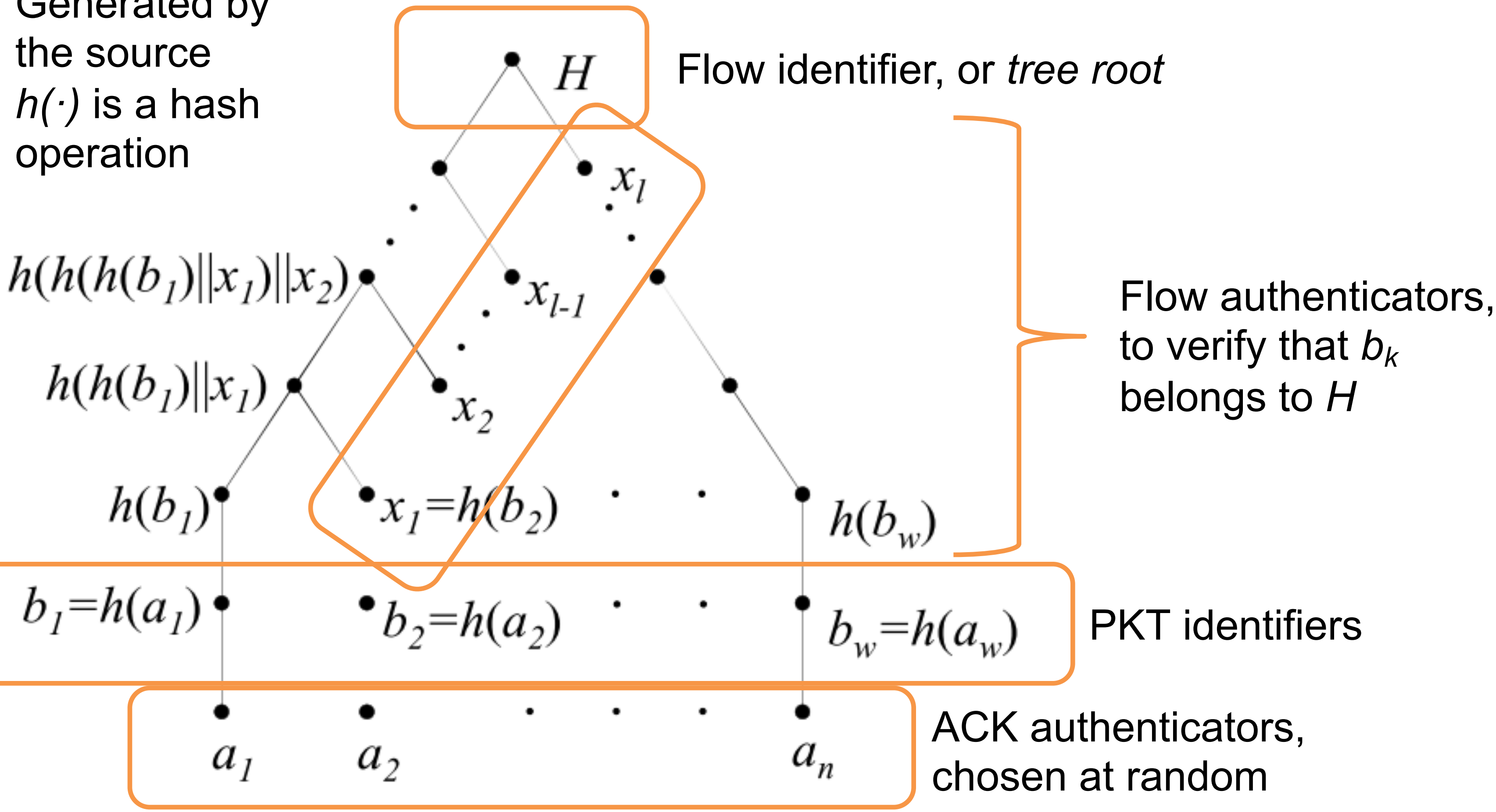
Represents transmission of ACK

ACK = (AckAuth)

Slide by Michael Noisternig

Castor: Merkle Hash Tree

- Generated by the source
- *h(·)* is a hash operation



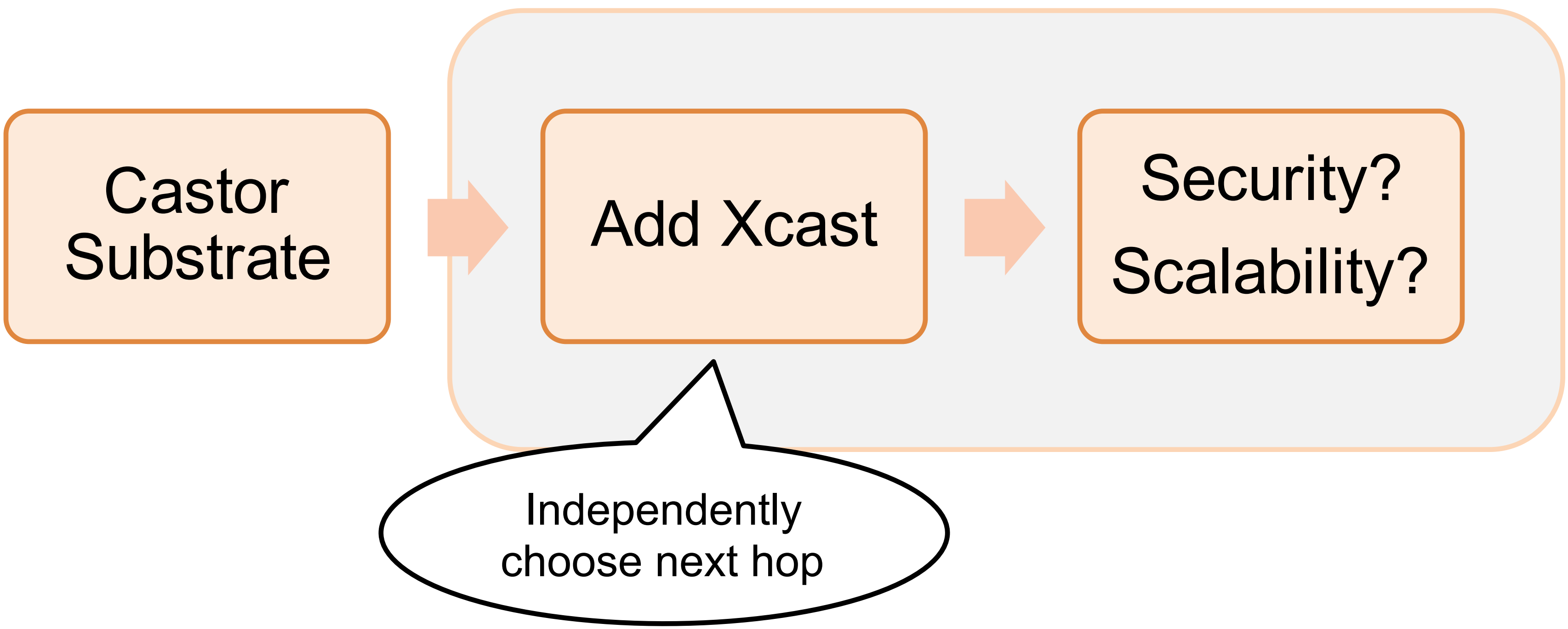
Outline

1. Motivation
2. Castor [1]
3. Xcastor [2]
   1. Design
   2. Short evaluation
4. Conclusion

[1] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer, "Castor: Scalable Secure Routing for Ad Hoc Networks," in *Proceedings of the IEEE Conference on Computer Communications*, 2010, pp. 1–9.

[2] M. Schmittner, "Scalable and Secure Multicast for Mobile Ad-hoc Networks," *Master thesis*, Technische Universität Darmstadt, 2014.

Protocol Design: Choices & Goals



Protocol Design: Xcast with Castor

$$PKT = (s, d, H, b_k, f_k, e_k)$$

$$ACK = (a_k)$$

Protocol Design: Xcast with Castor

SECURE MOBILE NETWORKING

$$PKT = (s, \underbrace{d_1, d_2, \dots, d_n}_{\text{Include all destinations}}, H, b_k, f_k, \underbrace{e_k}_{e_k = Enc_{K_{s,d}}(a_k)})$$

$$ACK = (a_k)$$

CASED

TECHNISCHE UNIVERSITÄT DARMSTADT

Protocol Design: Xcast with Castor

$$PKT = \left(s, d_1, d_2, \dots, d_n, H, b_k, f_k, \underbrace{e_{k,G}}\right)$$

Encrypt for group

$$e_{k,G} = Enc_{K_G}(a_k)$$

$$ACK = (a_k)$$

Protocol Design: Xcast with Castor

$$PKT = (s, d_1, d_2, \dots, d_n, H, b_k, f_k, e_{k,G})$$

$$ACK = (a_k)$$

They all look the same!

Protocol Design: Xcast with Castor

$$PKT = (s, d_1, d_2, \dots, d_n, H, b_k, f_k, e_{k,G})$$

$$b_k = Hash(a_k)$$

?

$$ACK = (e_{k,i})$$

Encrypt individually

$$\mathrm{e}_{k,i} = \mathrm{Enc}_{K_{s,d_i}}(a_k)$$

Protocol Design: Xcast with Castor

$$PKT = \left(s, d_1, d_2, \dots, d_n, H, b_k, \underbrace{b_{k,1}, b_{k,2}, \dots, b_{k,n}}_{\text{Individual PKT ids}}, f_k, e_{k,G}\right)$$

$$b_{k,i} = Hash\left(e_{k,i}\right)$$

$$ACK = (\underbrace{e_{k,i}}_{\text{Encrypt individually}})$$

$$\mathrm{e}_{k,i} = \mathrm{Enc}_{K_{s,d_i}}(a_k)$$

Protocol Design: Xcast with Castor

SECURE MOBILE NETWORKING

$$PKT = \left(s, d_1, d_2, \ldots, d_n, H, b_k, b_{k,1}, b_{k,2}, \ldots, b_{k,n}, f_k, e_{k,G}\right)$$

Why encrypt?

$$ACK = (e_{k,i})$$

Protocol Design: Xcast with Castor

$$PKT = (s, d_1, d_2, \dots, d_n, H, b_k, b_{k,1}, b_{k,2}, \dots, b_{k,n}, f_k, \underbrace{a_k}_{\text{Plaintext}})$$

$$ACK = (e_{k,i})$$

Still secure!
(attacker needs shared key)

Protocol Design: Xcast with Castor

$$PKT = \left(s, d_1, d_2, \dots, d_n, H, b_k, b_{k,1}, b_{k,2}, \dots, b_{k,n}, f_k, a_k\right)$$

$$ACK = (e_{k,i})$$

Protocol Design: Xcast with Castor

$$PKT = \left(s, d_1, d_2, \ldots, d_n, H, b_k, b_{k,1}, b_{k,2}, \ldots, b_{k,n}, f_k, a_k\right)$$

$$b_k = \mathrm{Hash}(a_k)$$

$$ACK = (e_{k,i})$$

Protocol Design: Xcast with Castor

$$PKT = \left(s, d_1, d_2, \dots, d_n, H, b_{k,1}, b_{k,2}, \dots, b_{k,n}, f_k, a_k\right)$$

Drop b_k

$$ACK = (e_{k,i})$$

Protocol Design: Xcast with Castor

$$PKT = (s, d_1, d_2, \dots, d_n, H, b_{k,1}, b_{k,2}, \dots, b_{k,n}, f_k, a_k)$$

Drop b_k

$$ACK = (e_{k,i})$$

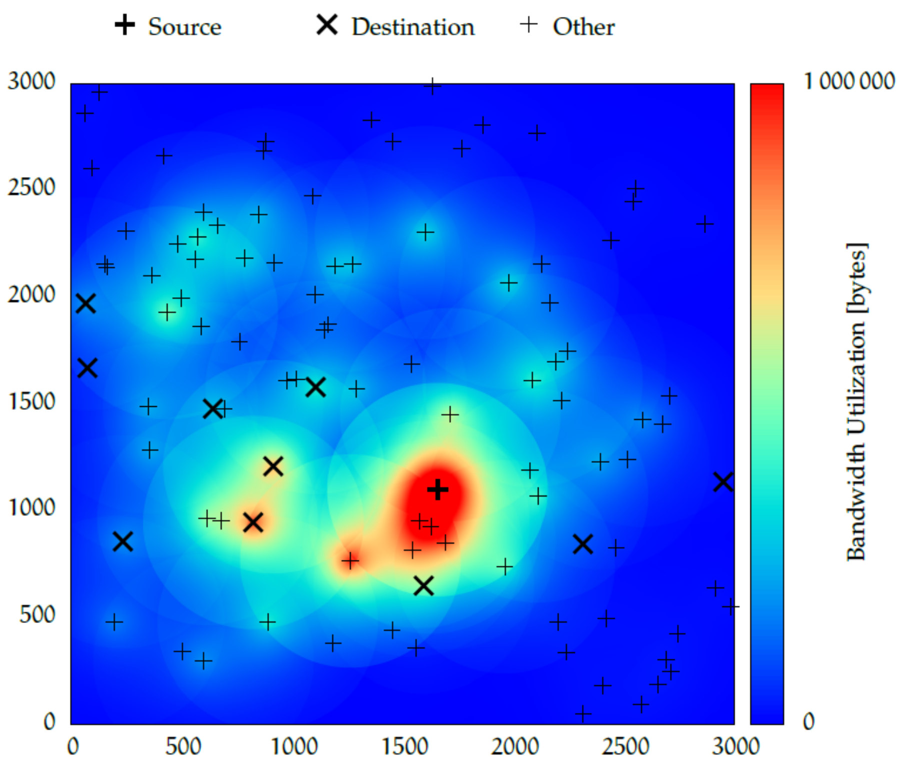
Xcastor: Summary

- **Design** in a nutshell
  - Explicit multicast
    - source node includes each destination explicitly in every PKT header
    - Each destination receives its unique PKT identifer
  - Intermediate nodes keep routing state per *subflow* (= Castor flow + destination)
    - Routing decisions are made independently for each destination
- Same **security** features as Castor
- **Scalable** for many small groups
    - Additional per-PKT overhead: $(n-1)\times\left(size(d_i)+size(b_{k,i})\right)$

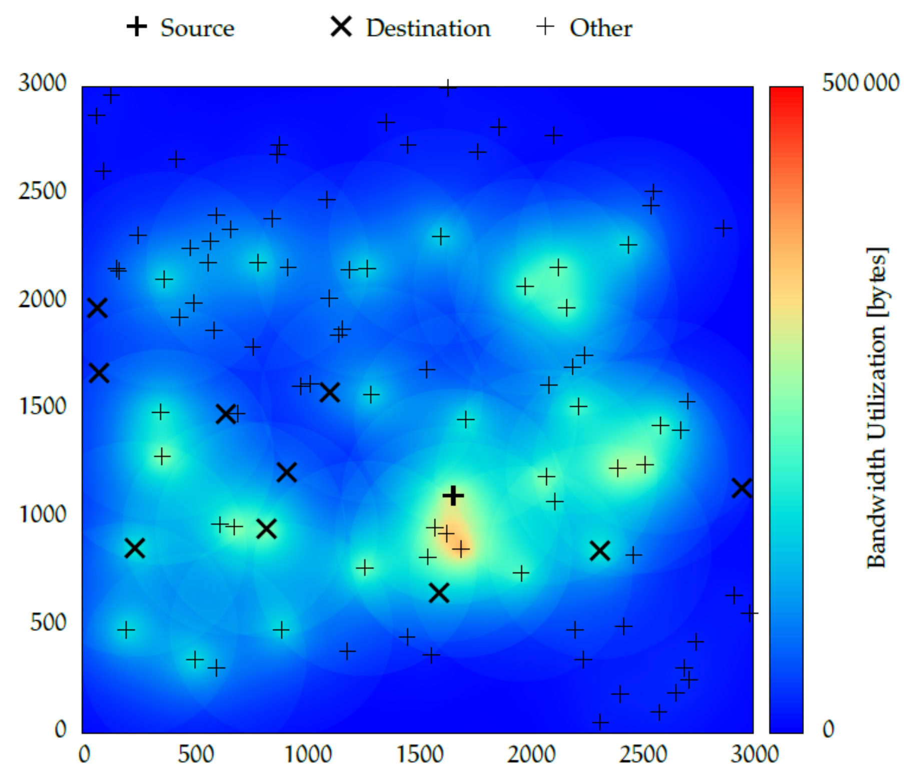
Xcastor: Evaluation

SECURE MOBILE NETWORKING

Castor



Xcastor

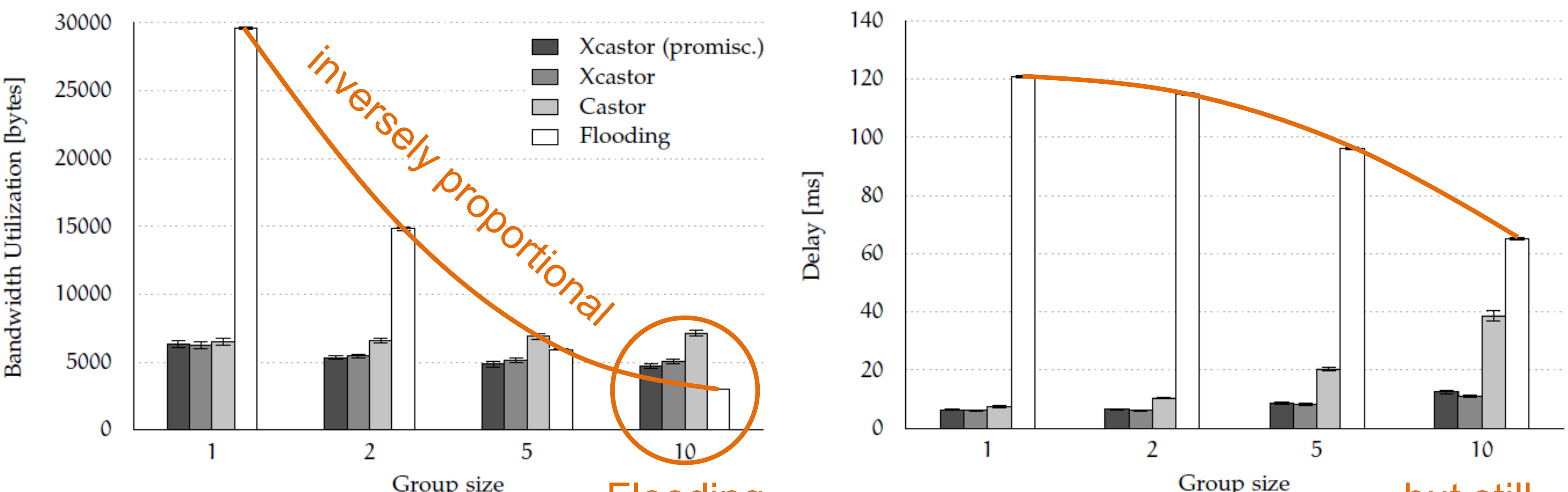


- Congestion around the source (increases with group size)

- Less traffic
- Traffic more evenly spread
- But paths may not be optimal

Xcastor: Group Size Scalability

SECURE MOBILE NETWORKING



Flooding becomes more efficient…

… but still slower than the others

Outline

1. Motivation
2. Castor [1]
3. Xcastor [2]
4. Conclusion
    1. Practical Considerations
    2. Future Work

[1] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer, "Castor: Scalable Secure Routing for Ad Hoc Networks," in *Proceedings of the IEEE Conference on Computer Communications*, 2010, pp. 1–9.

[2] M. Schmittner, "Scalable and Secure Multicast for Mobile Ad-hoc Networks," *Master thesis*, Technische Universität Darmstadt, 2014.

Practical Considerations

- Group Management (Xcastor)
  - Currently uses an static mapping of IP multicast address → list of IP unicast addresses
  - How to dynamically (un)subscribe to certain groups?
  - If we want not only one-to-many but many-to-many communication, we need to keep consistent group state on all group members (overhead?)
- Key Management (Castor and Xcastor)
  - Need shared secret between source and destination
  - Group key between all group members if we want confidentiality

Future Work

Completed

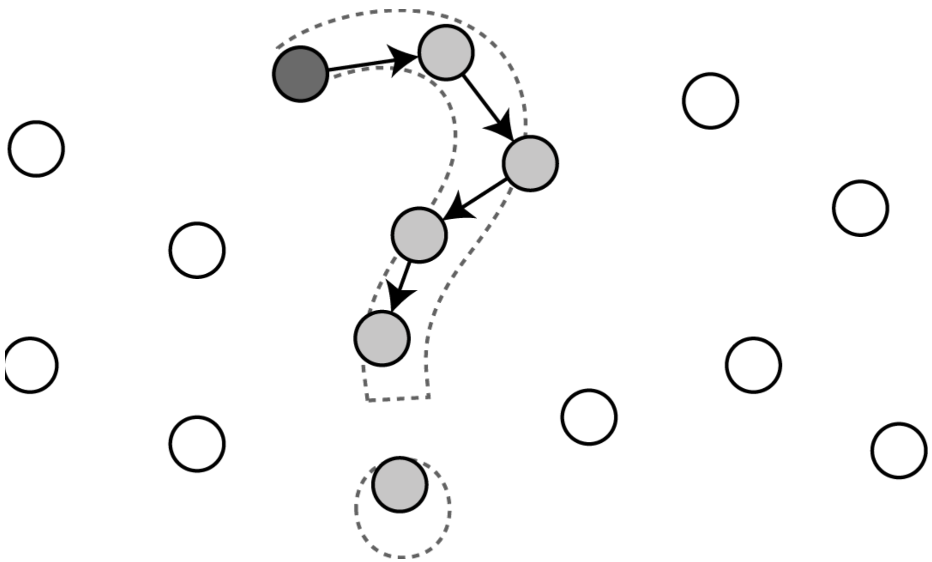
- ✓ Xcastor: Secure Explicit Multicast Routing as an extension to Castor
- ✓ Castor (and Xcastor) **implementations** in Click
- ✓ **Evaluation** in ns-3 and result analysis

ToDo

- Evaluation on real testbed
- Reimplementation of Xcastor (current version is broken due to Castor v2 enhancements)

Castor and Xcastor

Secure and Efficient Unicast and Multicast Routing for Mobile Ad Hoc Networks



Milan Schmittner
mschmittner@seemoo.tu-darmstadt.de
https://seemoo.de/mschmittner (PGP key)

SECURE MOBILE NETWORKING

CASED

TECHNISCHE UNIVERSITÄT DARMSTADT