Adding Disruption Tolerant Networking to UnetStack

Arnav Dhamija

Acoustic Research Laboratory, National University of Singapore

arnav.dhamija@gmail.com

April 25, 2019

Agenda

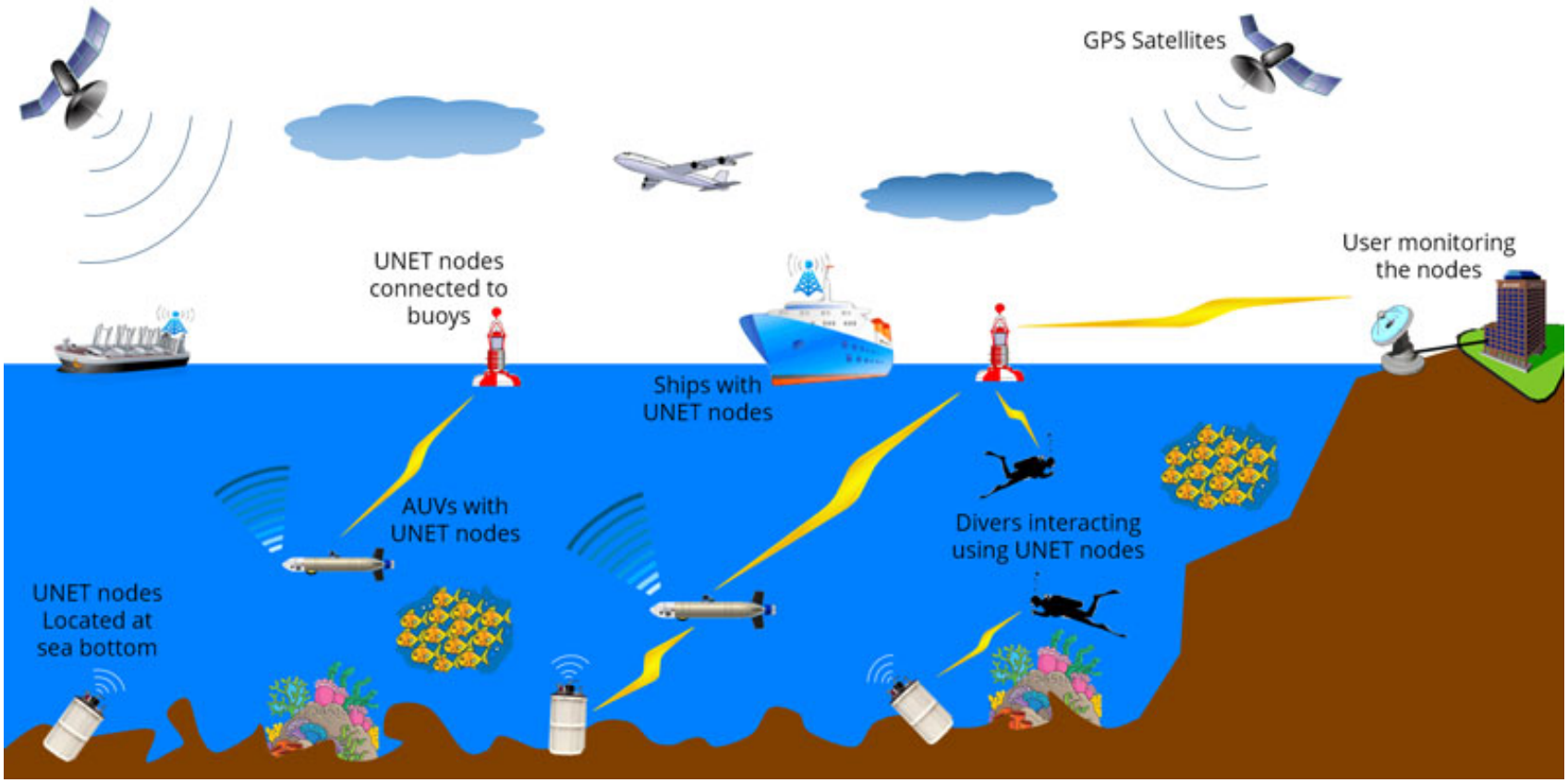
Challenges in Underwater Networks

- Underwater networks typically use acoustic waves

Challenges in Underwater Networks

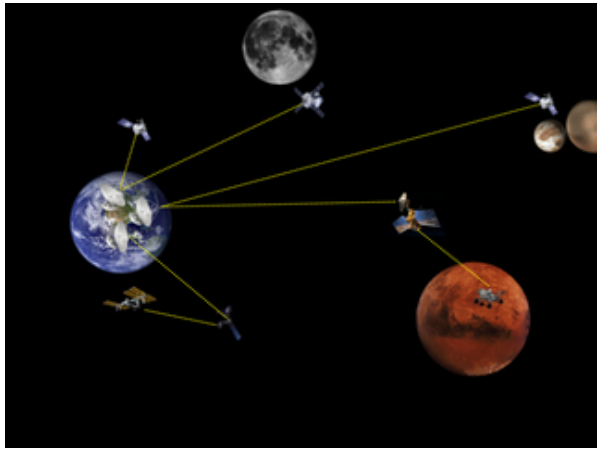
- Underwater networks typically use acoustic waves
- Challenges:
  - Noise from ships, shrimp, bubbles
  - Surface characteristics
  - Interference from other transmissions
  - Link availability
  - High energy consumption
- Disruptions can be high and reliability can be low, hence an ideal place for using *DTN* protocols

Example


GPS Satellites
UNET nodes
connected to
buoys
User monitoring
the nodes
Ships with
UNET nodes
AUVs with
UNET nodes
Divers interacting
using UNET nodes
UNET nodes
Located at
sea bottom

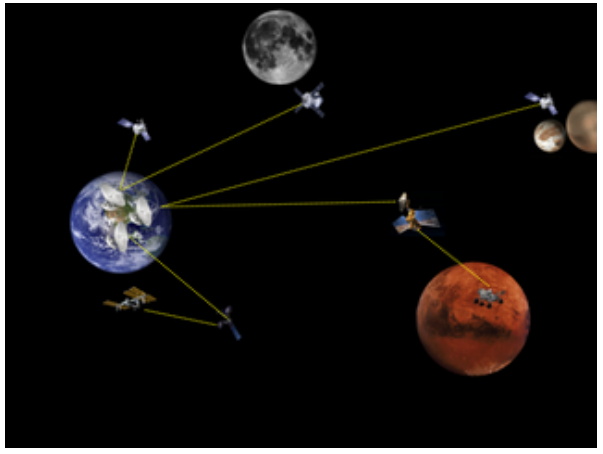
Disruption Tolerant Networks

- Used where the communication network is likely to be disrupted due to:
  - Network Topology (Deep Space Networks / VANETs)
  - Environmental Conditions (Underwater Networks)



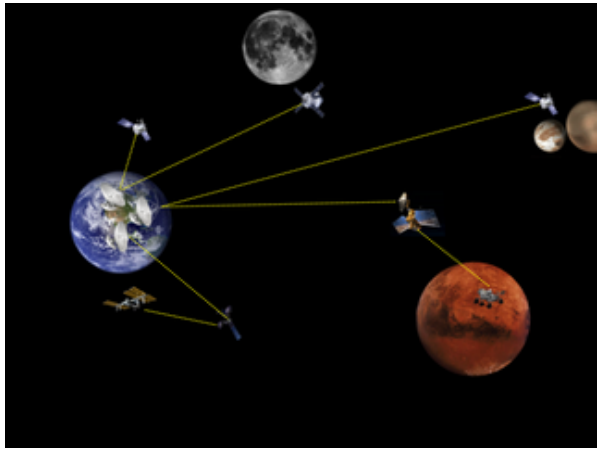
Disruption Tolerant Networks

- Used where the communication network is likely to be disrupted due to:
  - Network Topology (Deep Space Networks / VANETs)
  - Environmental Conditions (Underwater Networks)
- Prioritises successful message delivery over network throughput



Disruption Tolerant Networks

- Used where the communication network is likely to be disrupted due to:
  - Network Topology (Deep Space Networks / VANETs)
  - Environmental Conditions (Underwater Networks)
- Prioritises successful message delivery over network throughput
- Very different from typical Internet protocols!



Key Features

- For making a network tolerant to delays and disruptions, DTNs typically have:
  - Store and Forward

Key Features

- For making a network tolerant to delays and disruptions, DTNs typically have:
  - Store and Forward
  - TTLs for messages

Key Features

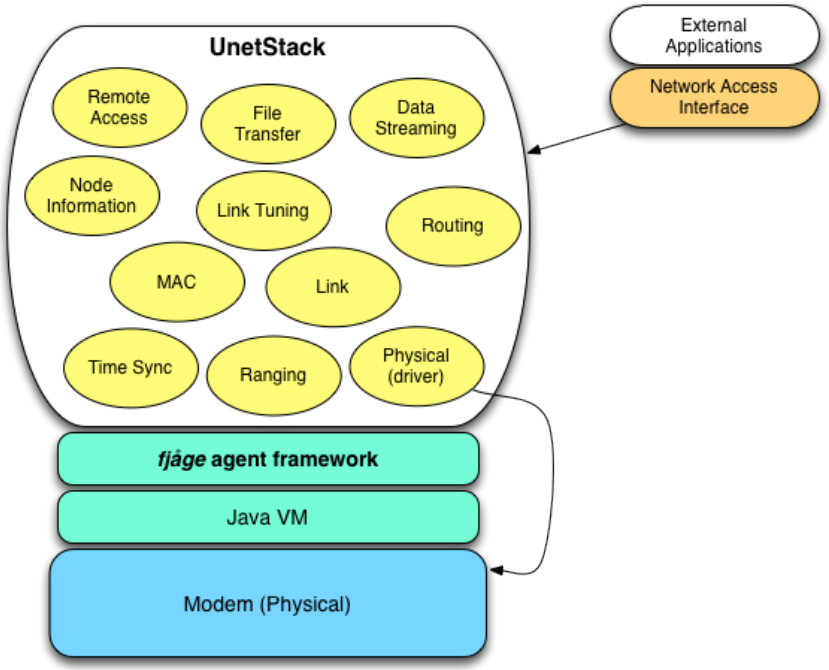
- For making a network tolerant to delays and disruptions, DTNs typically have:
  - Store and Forward
  - TTLs for messages
  - Dedicated routing algorithms (SNW, PRoPHET, MaxProp, etc)

Key Features

- For making a network tolerant to delays and disruptions, DTNs typically have:
  - Store and Forward
  - TTLs for messages
  - Dedicated routing algorithms (SNW, PRoPHET, MaxProp, etc)
- We are *not* going to focus on Routing and multi-copy
- Emphasis on efficiency!

UnetStack

- Underwater Network Simulator built on top of fjåge, written in Java and Groovy
- Agent based design
- Cross-layer optimisation, unlike layered network stack
- Any layer can talk to other layers!



DtnLink

Putting it all together...

- DTNs can be useful in underwater networks
- We need a new UnetAgent to implement all this functionality
- A `LINK` agent, leveraging capabilities of existing agents

DtnLink

Putting it all together...

- DTNs can be useful in underwater networks
- We need a new UnetAgent to implement all this functionality
- A `LINK` agent, leveraging capabilities of existing agents
- Let's call it `DtnLink`

Agenda

What can DtnLink be used for?

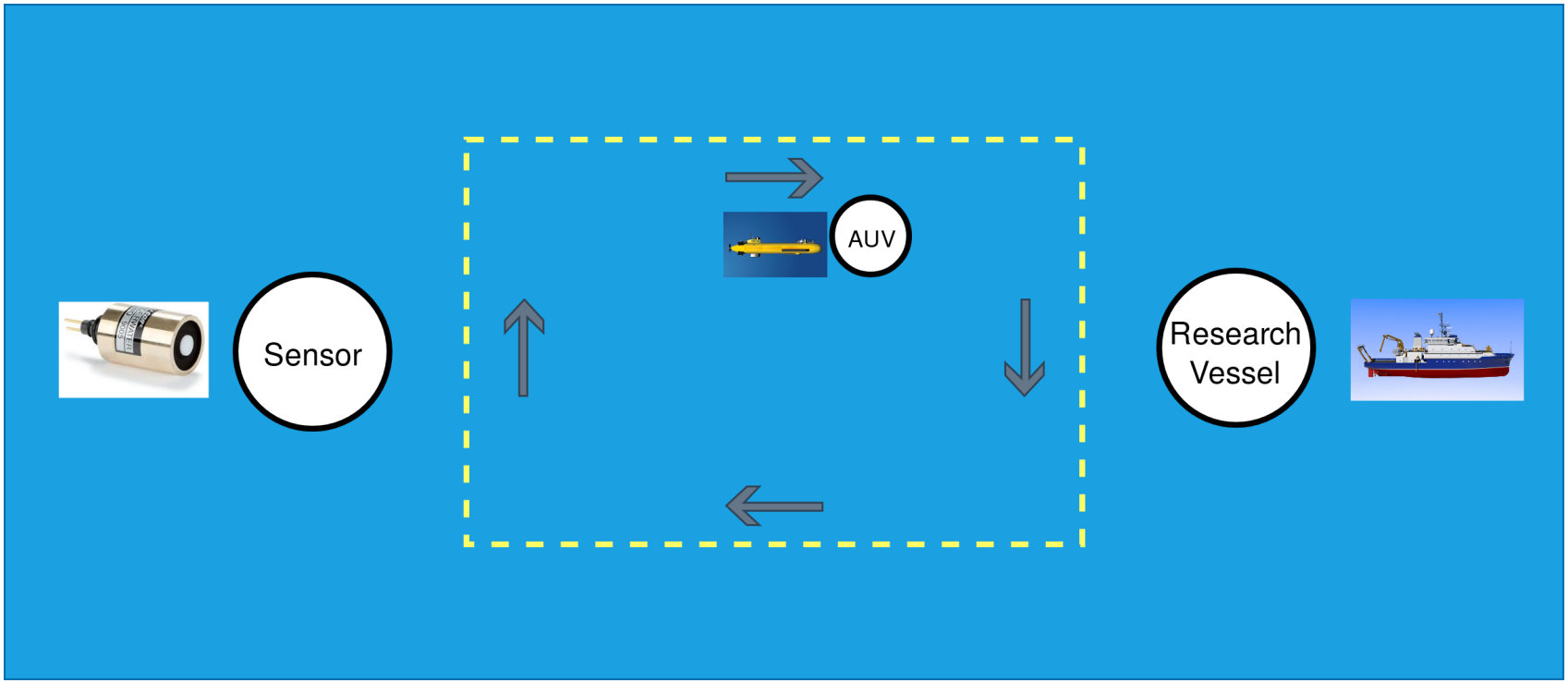
- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:

What can DtnLink be used for?

- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:
  - Data Muling

Data Muling

Using mobile nodes (e.g. AUVs) for relaying messages



What can DtnLink be used for?

- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:
  - Data Muling
  - Time Varying Links

Time Varying Links

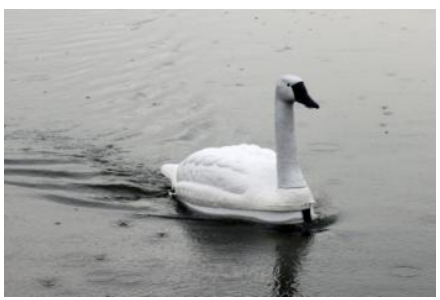
- In underwater networks, not all links may be available at all time
  - Acoustic links
  - Optical links
  - WiFi/LTE links
- How do we choose the link to use?

Time Varying Links

- In underwater networks, not all links may be available at all time
  - Acoustic links
  - Optical links
  - WiFi/LTE links
- How do we choose the link to use?
- `DtnLink` can choose the link based on which it `SNOOP`s a message
- Each node periodically sends *Beacons* for advertisement

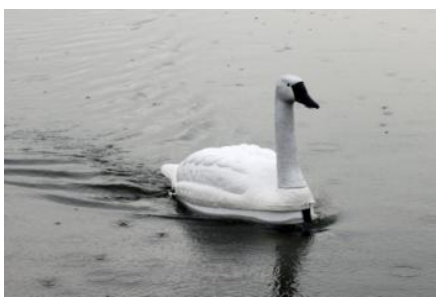
What can DtnLink be used for?

- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:
  - Data Muling
  - Time Varying Links
  - NUSwan



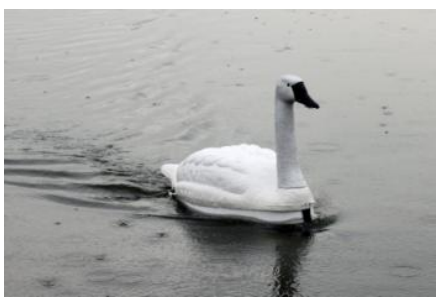
What can DtnLink be used for?

- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:
  - Data Muling
  - Time Varying Links
  - NUSwan



What can DtnLink be used for?

- Useful where the channel medium is lossy, delivery times are not a priority
- Some ideas:
  - Data Muling
  - Time Varying Links
  - NUSwan
- *...and many more*



Agenda

Features

- `DtnLink` is a new UnetAgent

Features

- `DtnLink` is a new UnetAgent
- Features
  - Fragmentation of large messages
  - Detection of duplicate messages
  - Stop-And-Wait sending
  - Support for multiple links

Features

- `DtnLink` is a new UnetAgent
- Features
  - Fragmentation of large messages
  - Detection of duplicate messages
  - Stop-And-Wait sending
  - Support for multiple links
- Configurable Options
  - Link Priorities
  - Order of sending messages (`ARRIVAL`, `EXPIRY`, `RANDOM`)
  - Short-circuiting (send messages to destination without DTN headers)

PDU Structure

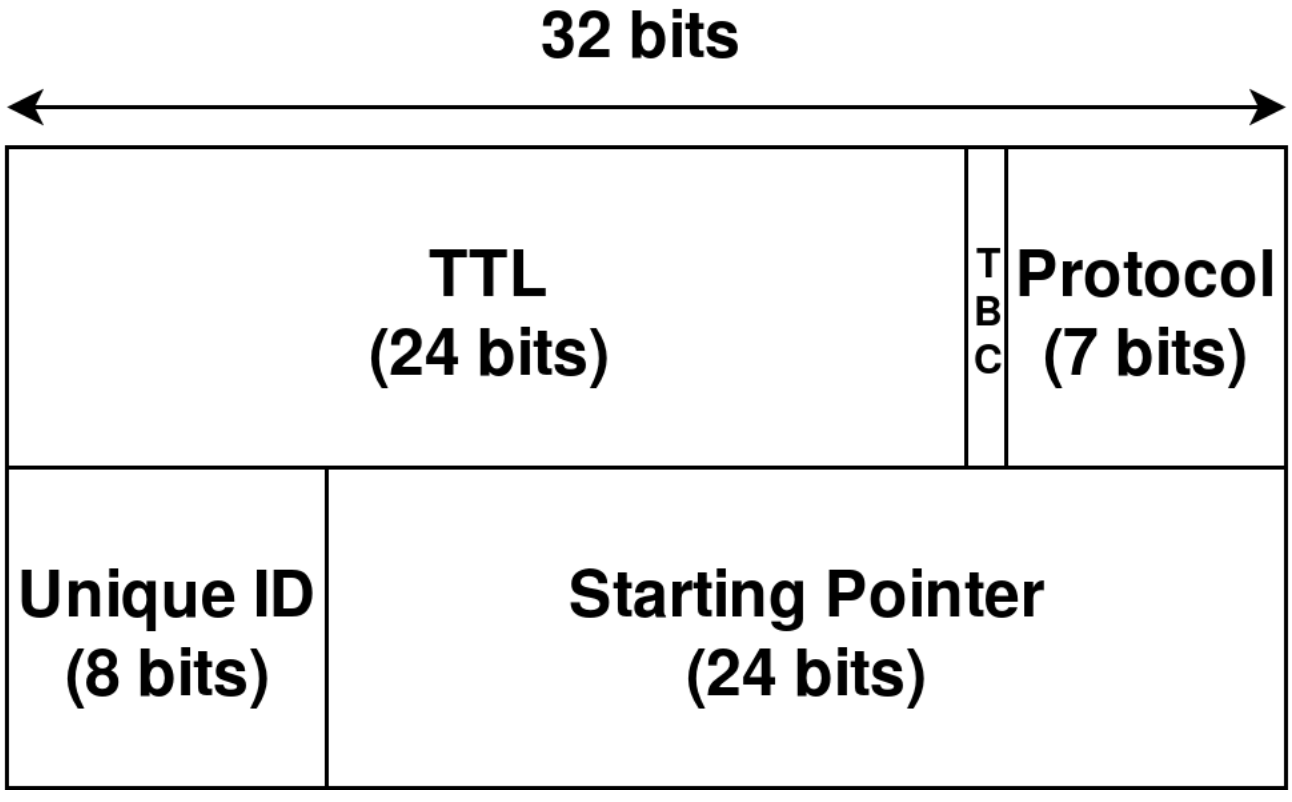
- **P**rotocol **D**ata **U**nit
- Consists of Headers + Data
- Added before the first byte of the data

PDU Structure

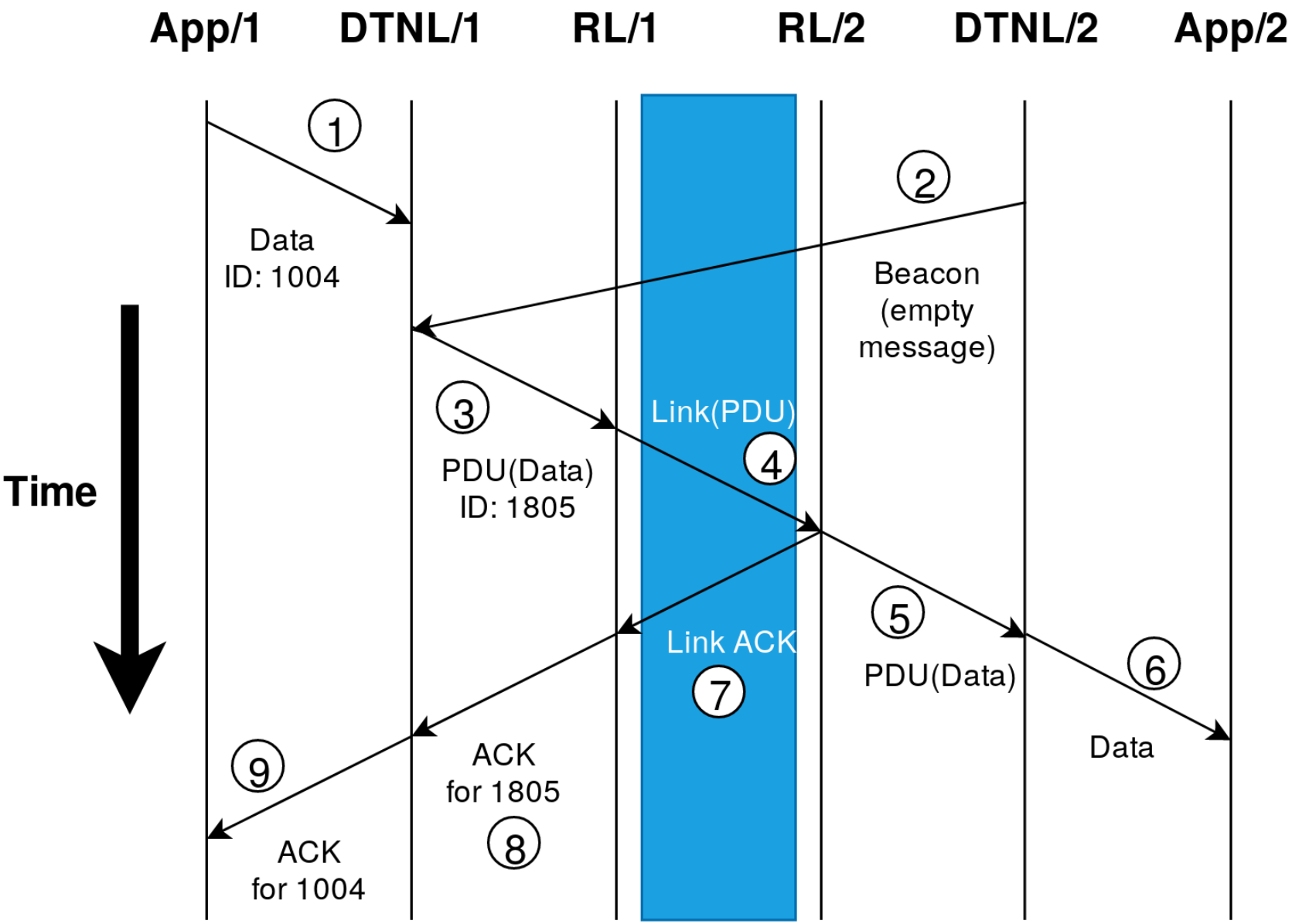
- **P**rotocol **D**ata **U**nit
- Consists of Headers + Data
- Added before the first byte of the data
- `DtnLink` uses a 64 bit header
- PDU size *must* be less than the MTU[1] of the Link used

[1] MTU = Maximum Transmission Unit

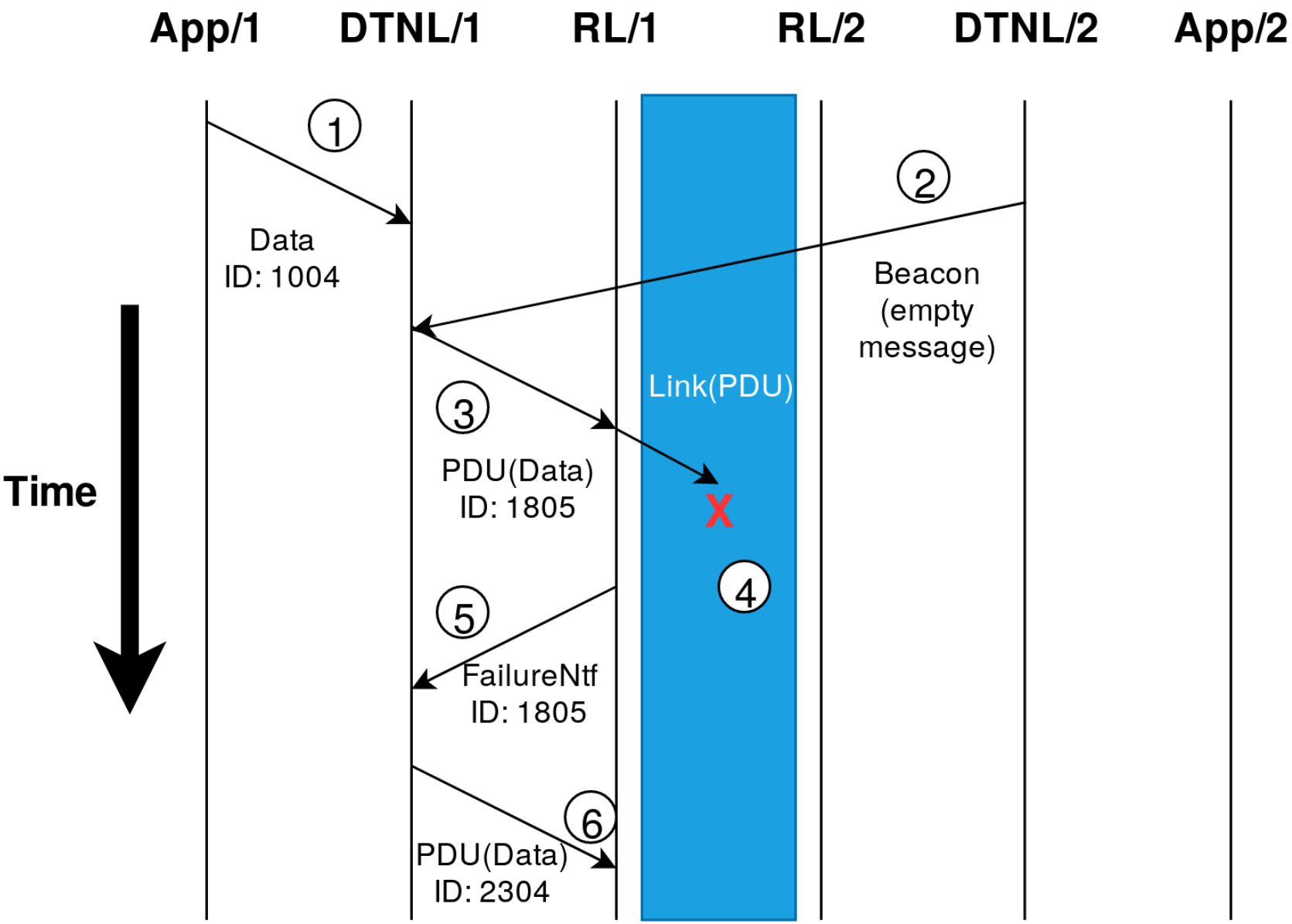
PDU Structure



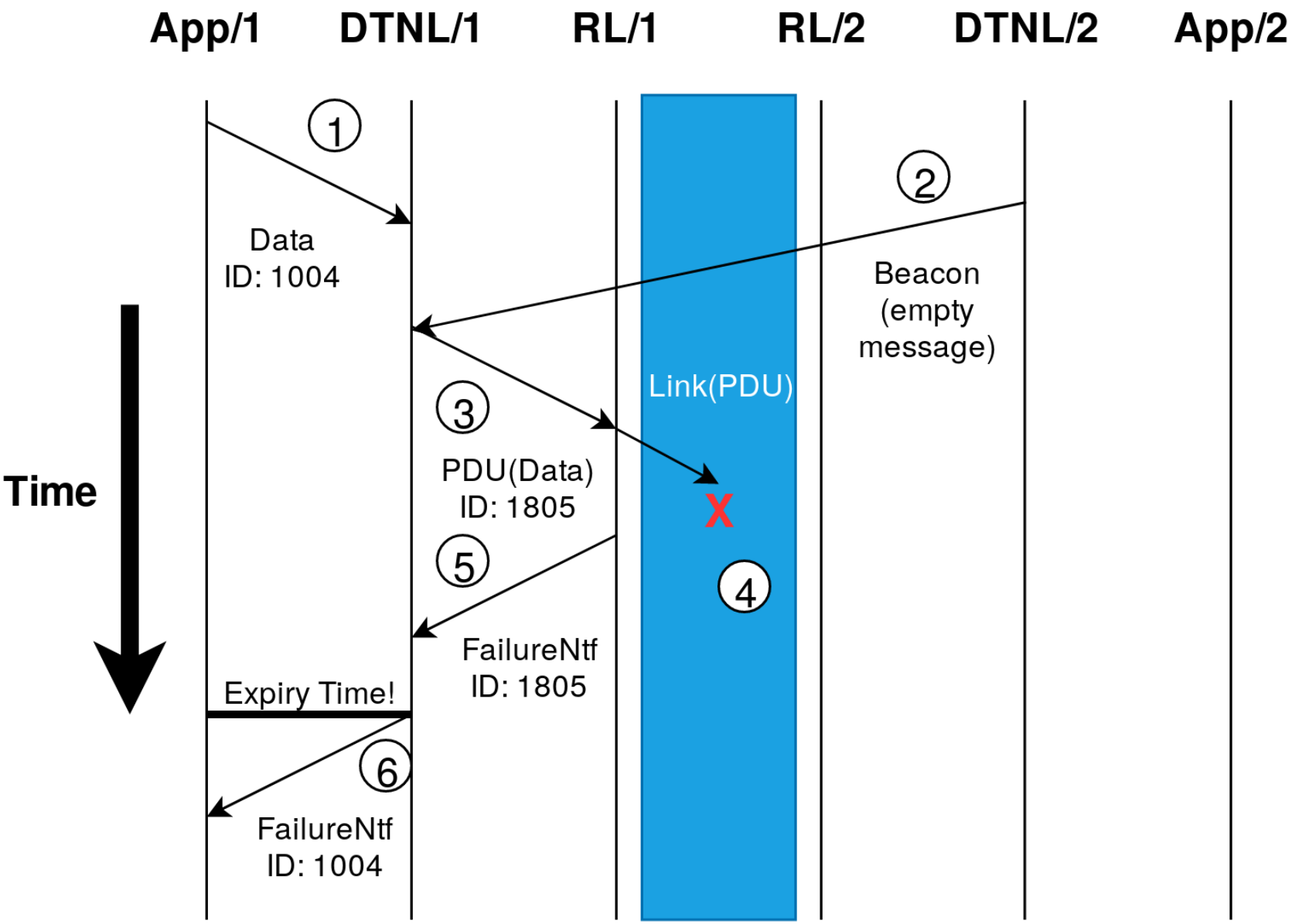
Successful Delivery



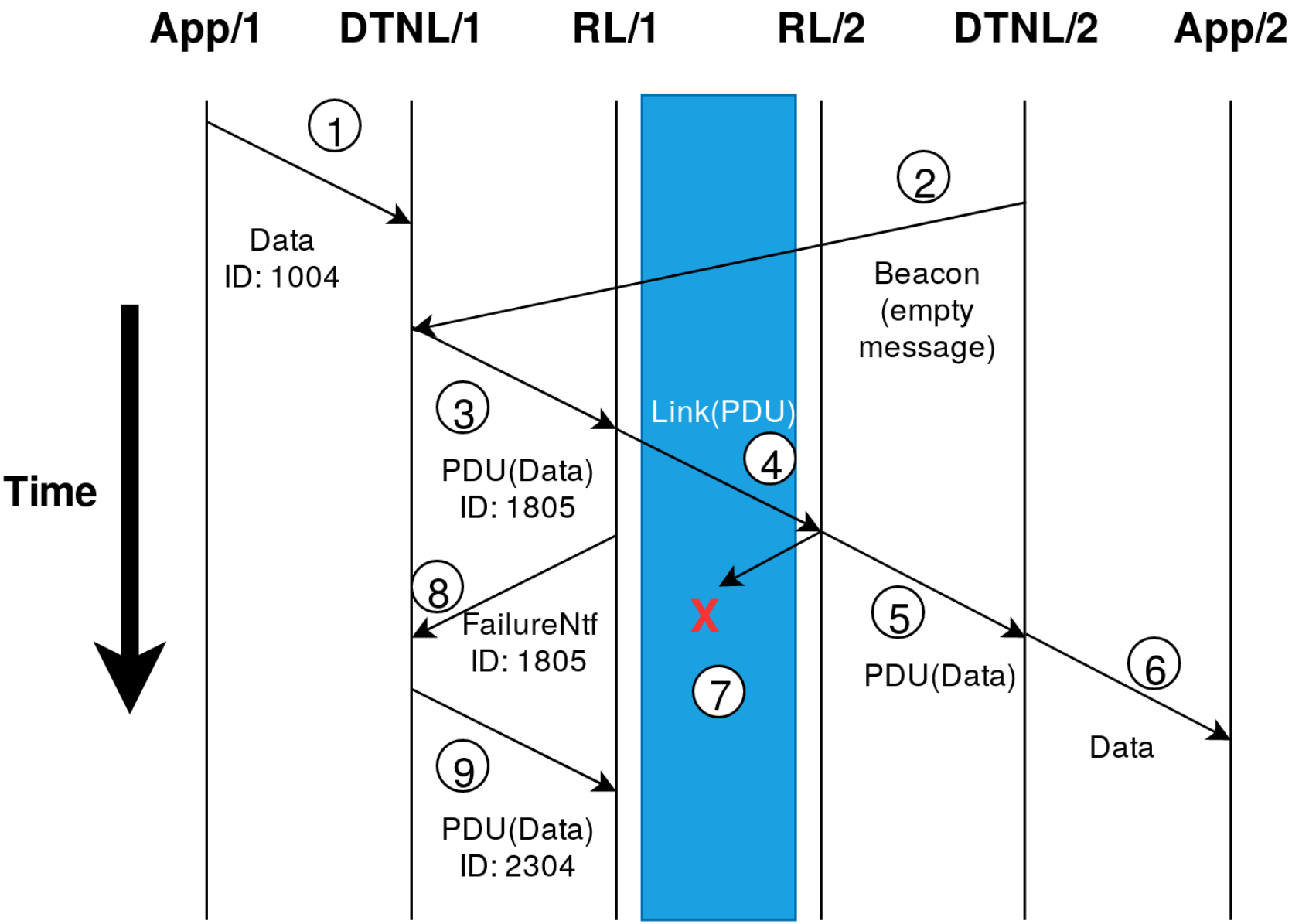
Delivery Failure



TTL Expiry


App/1
DTNL/1
RL/1
RL/2
DTNL/2
App/2
1
Data
ID: 1004
2
Beacon
(empty
message)
Link(PDU)
3
PDU(Data)
ID: 1805
X
4
5
FailureNtf
ID: 1805
Time
Expiry Time!
6
FailureNtf
ID: 1004

ACK Failure


App/1
DTNL/1
RL/1
RL/2
DTNL/2
App/2
Time
1
Data
ID: 1004
2
Beacon
(empty
message)
3
PDU(Data)
ID: 1805
Link(PDU)
4
X
5
PDU(Data)
6
Data
7
8
FailureNtf
ID: 1805
9
PDU(Data)
ID: 2304

Duplicate Messages

- ACK Fails can lead to duplicate messages
- We need a way to identify duplicate messages

Duplicate Messages

- ACK Fails can lead to duplicate messages
- We need a way to identify duplicate messages
- Idea: use a nonce for each message

Detecting Duplicate Messages with Hashing

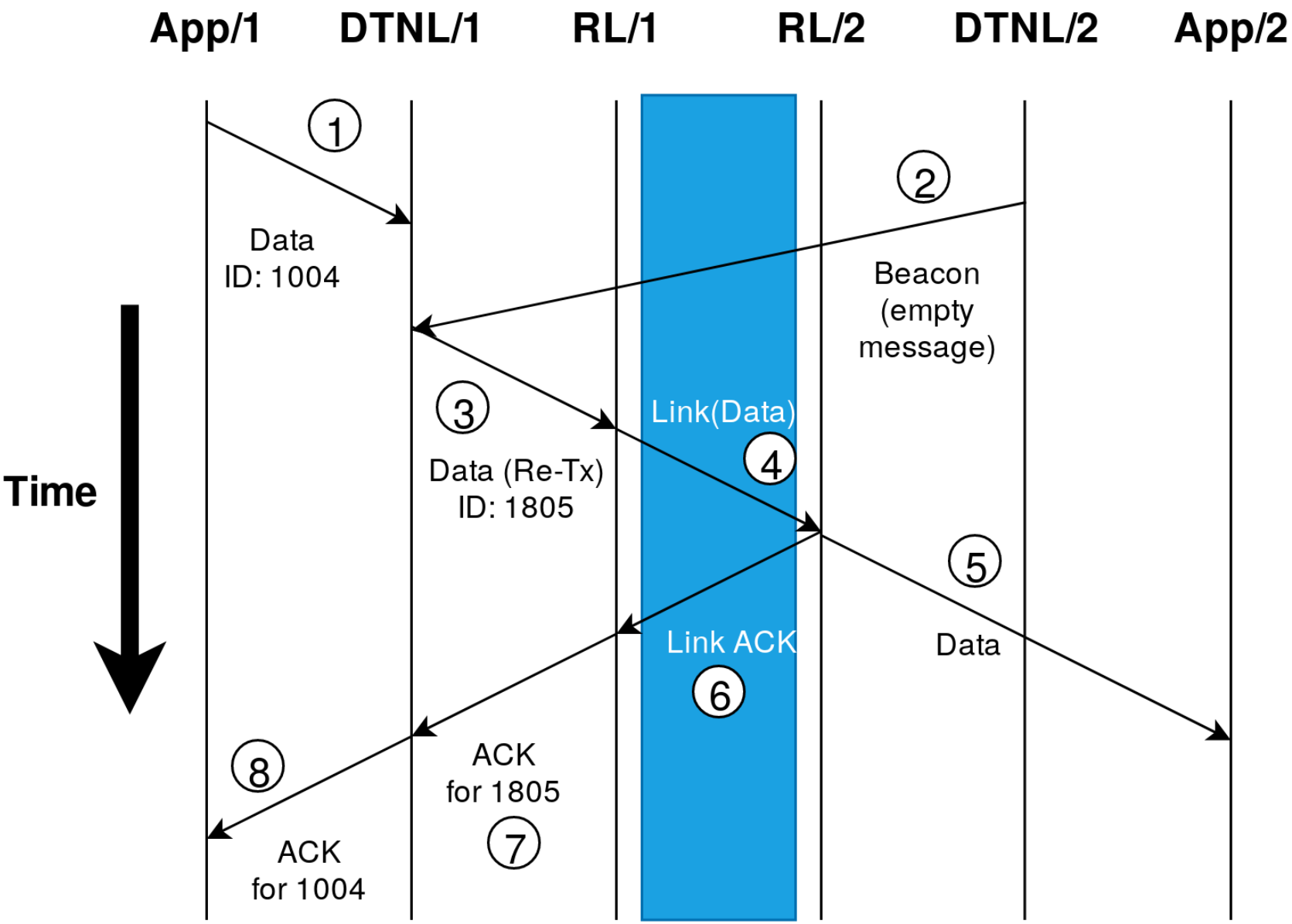
Sender

- Sender encodes a nonce in the PDU for each message

Receiver

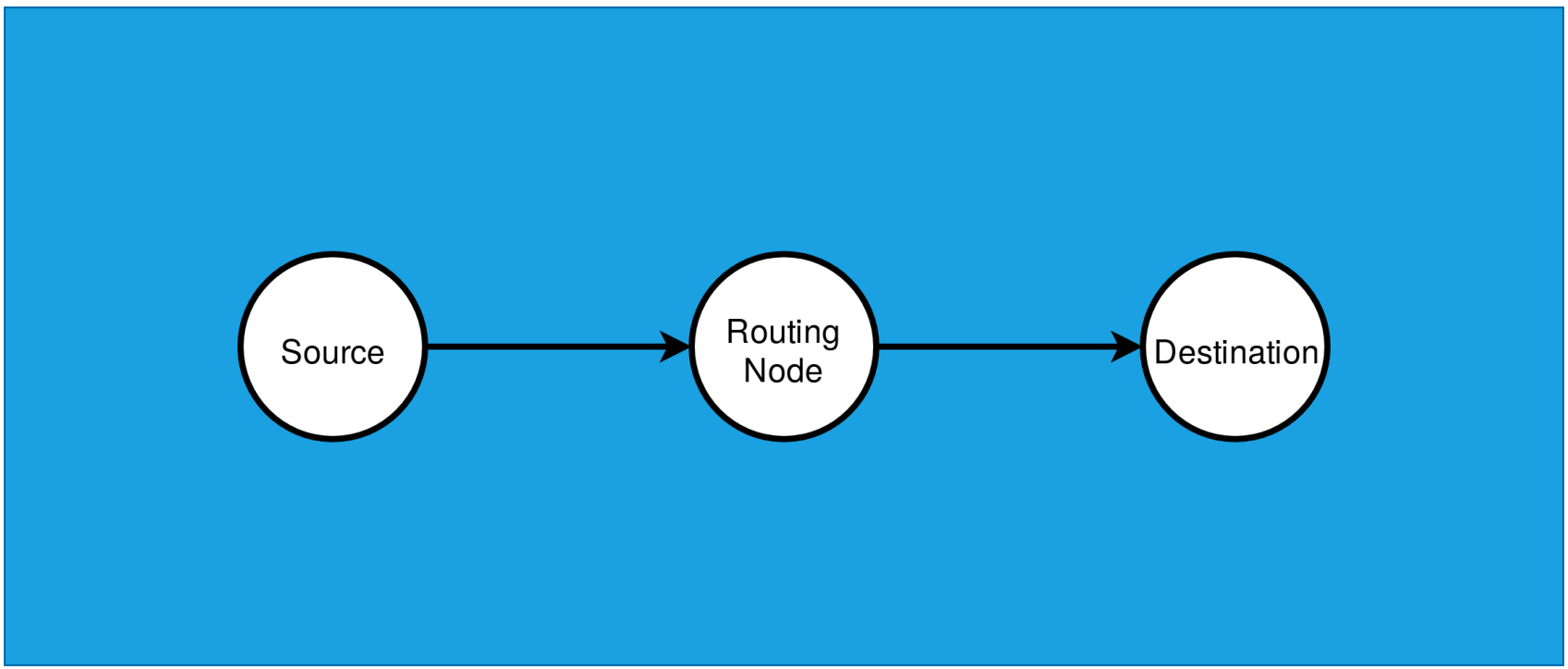
- Receiver computes `hashCode` of message data and the nonce
- Stores this value in a `Set`
- If current message's `hashCode` exists in the `Set`, discard the message
- Otherwise, send it up to the application

Short circuit

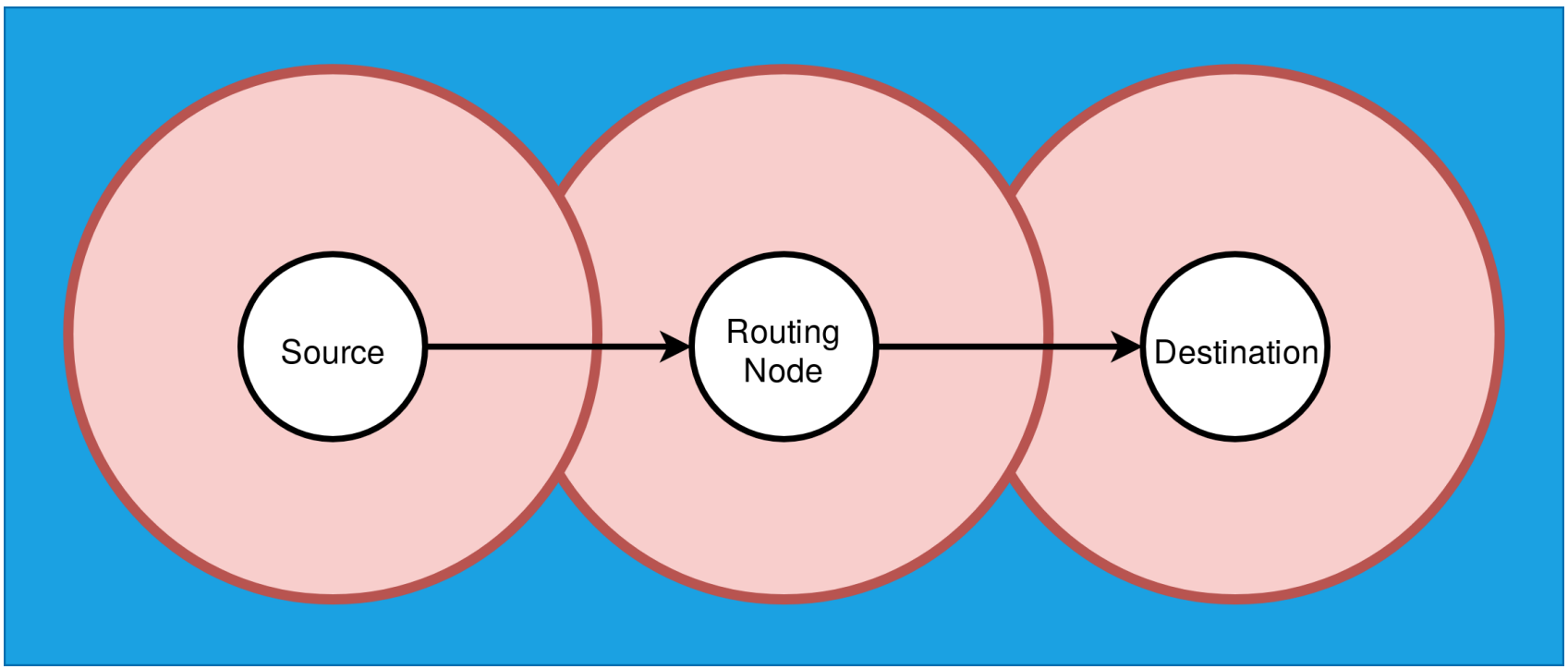


Agenda

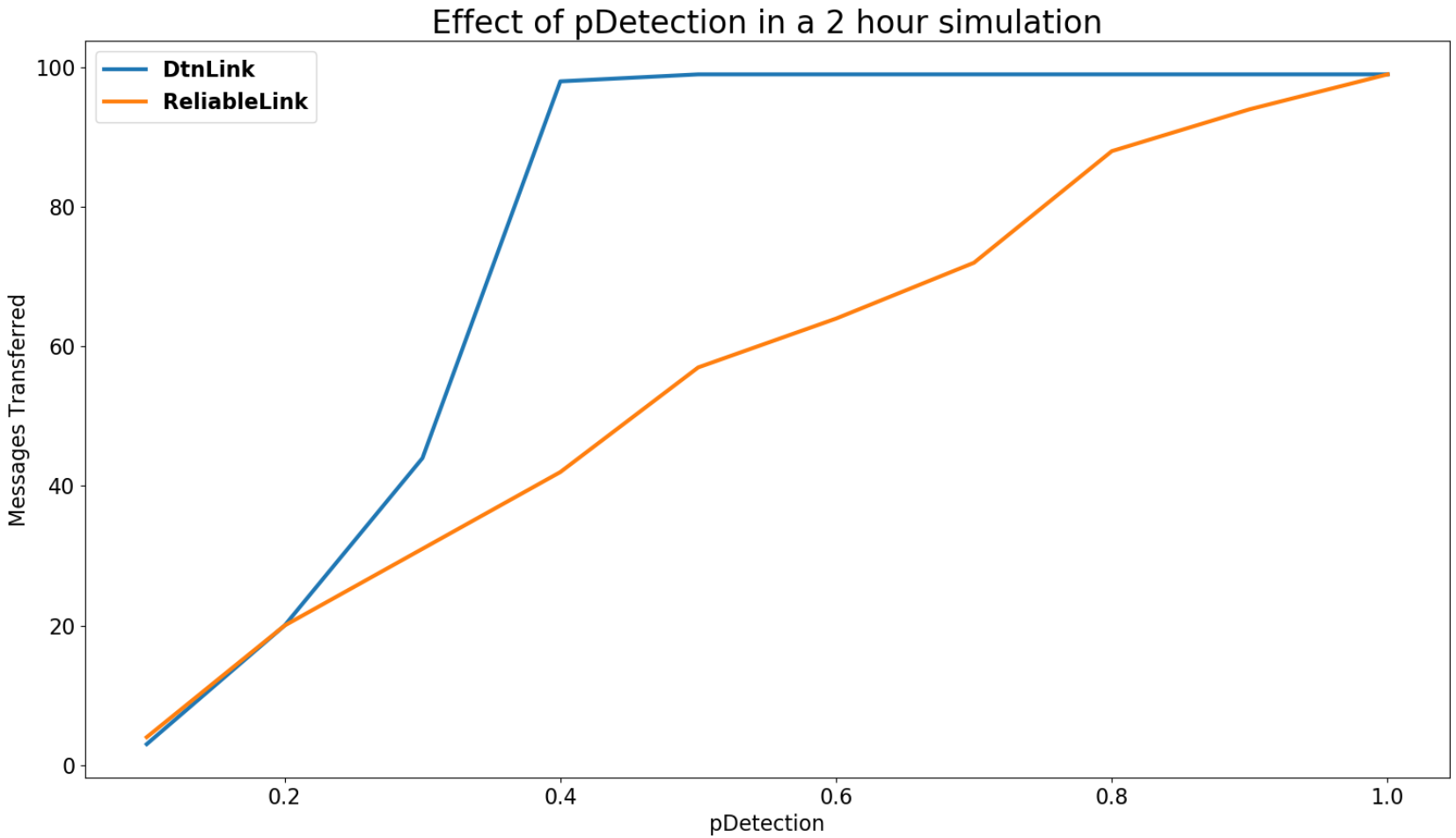
Multihop simulation



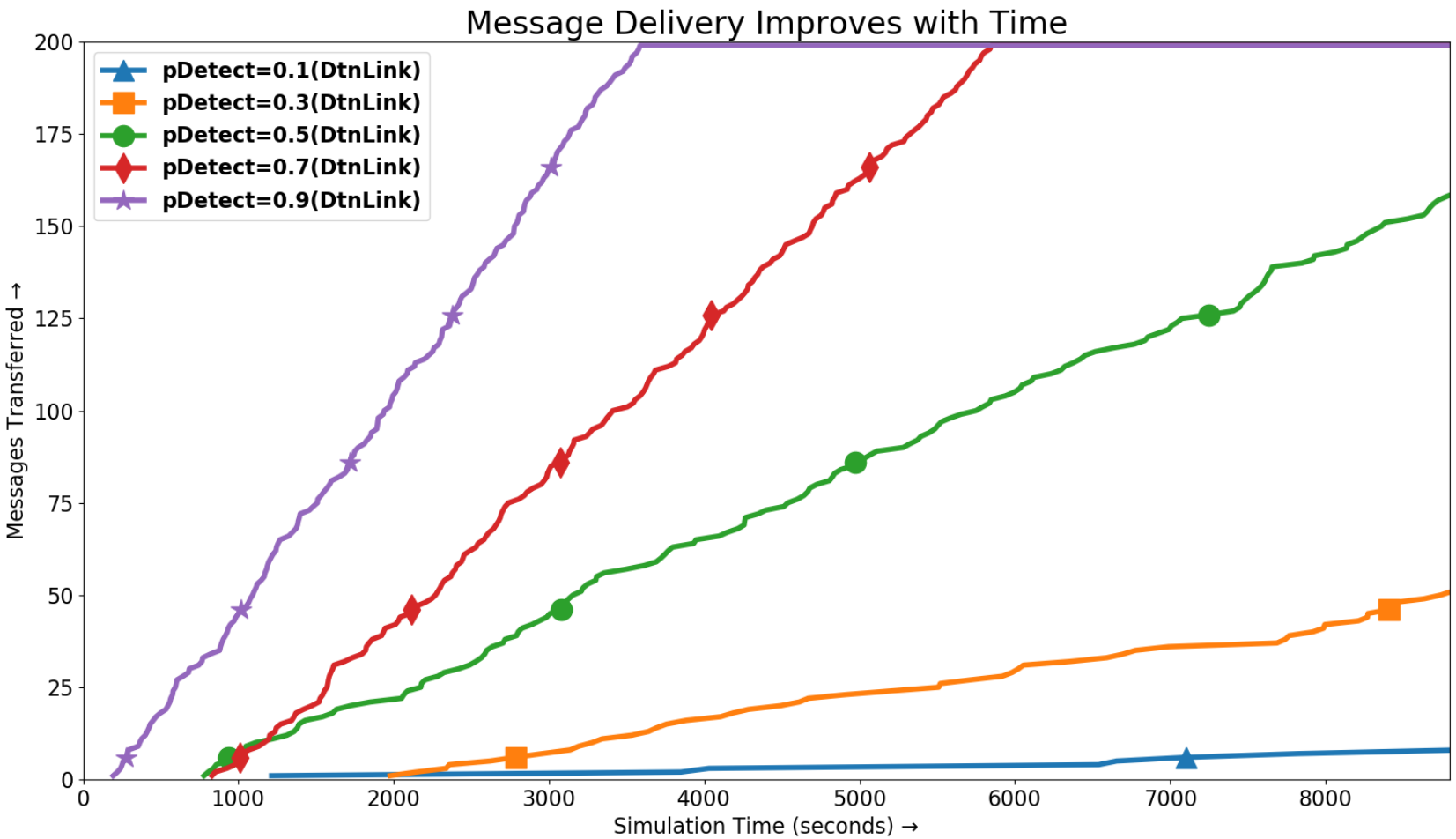
Multihop simulation



DtnLink versus ReliableLink



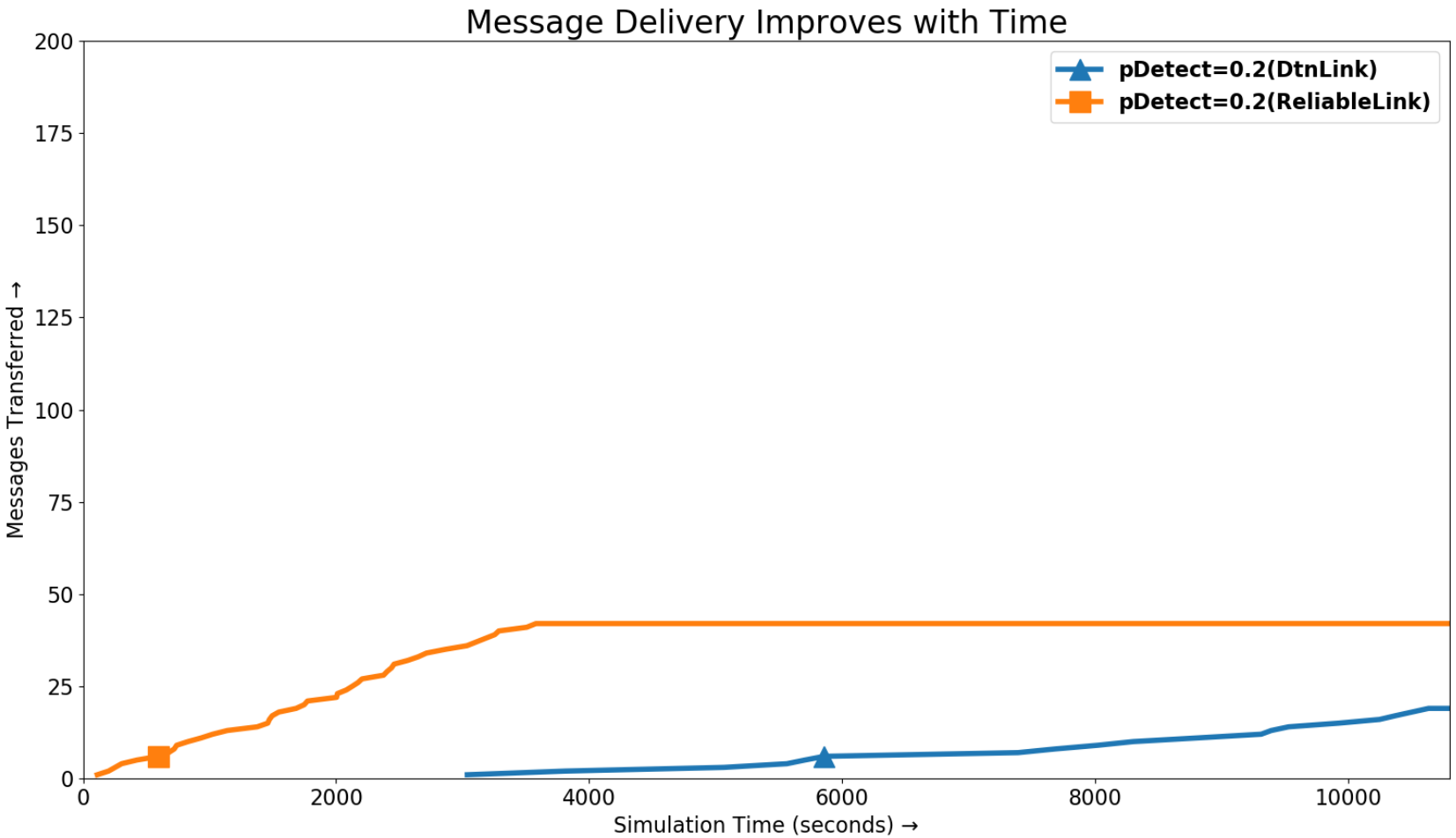
Message delivery keeps improving with time!



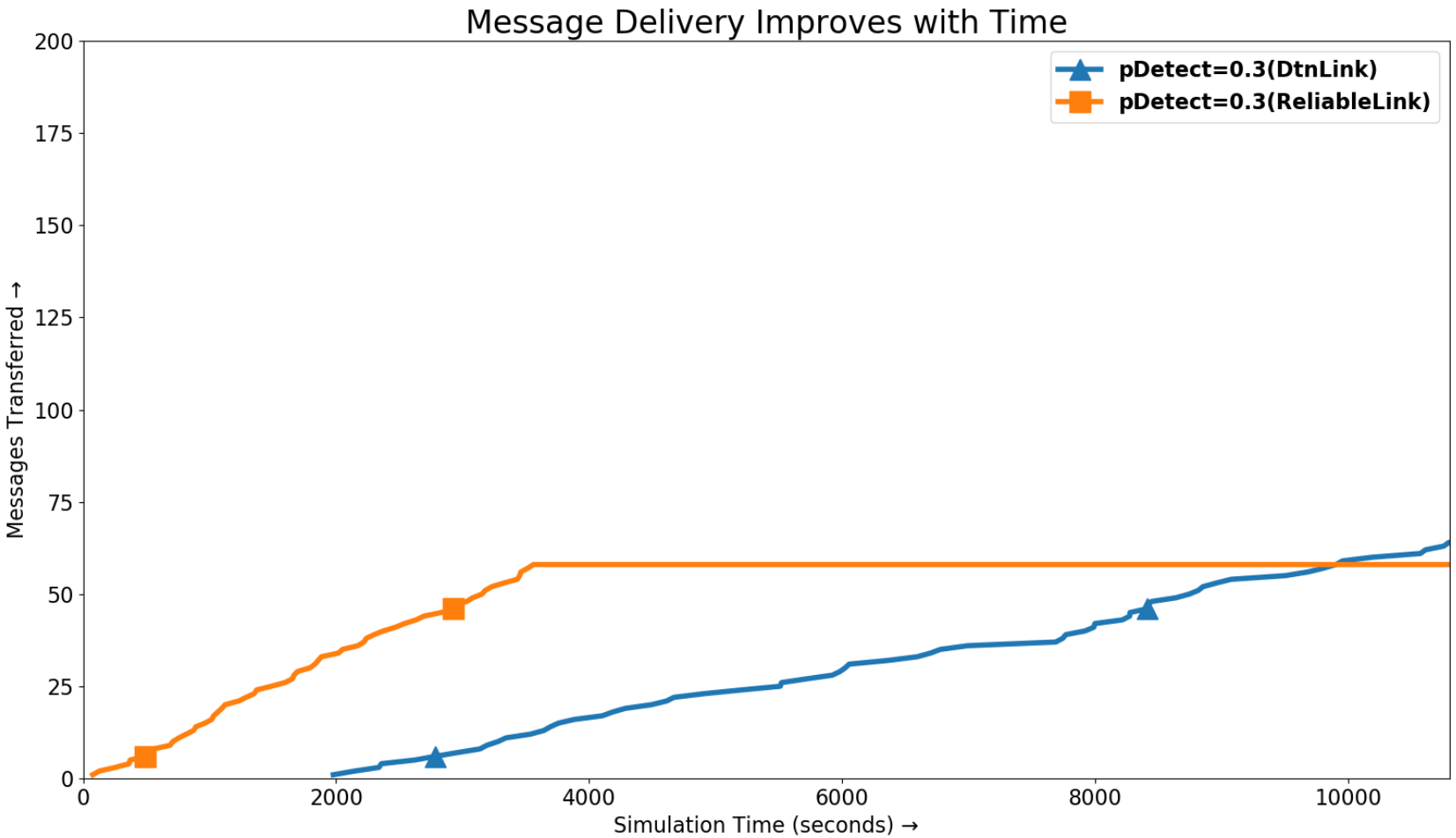
However...

`DtnLink` is not a panacea!

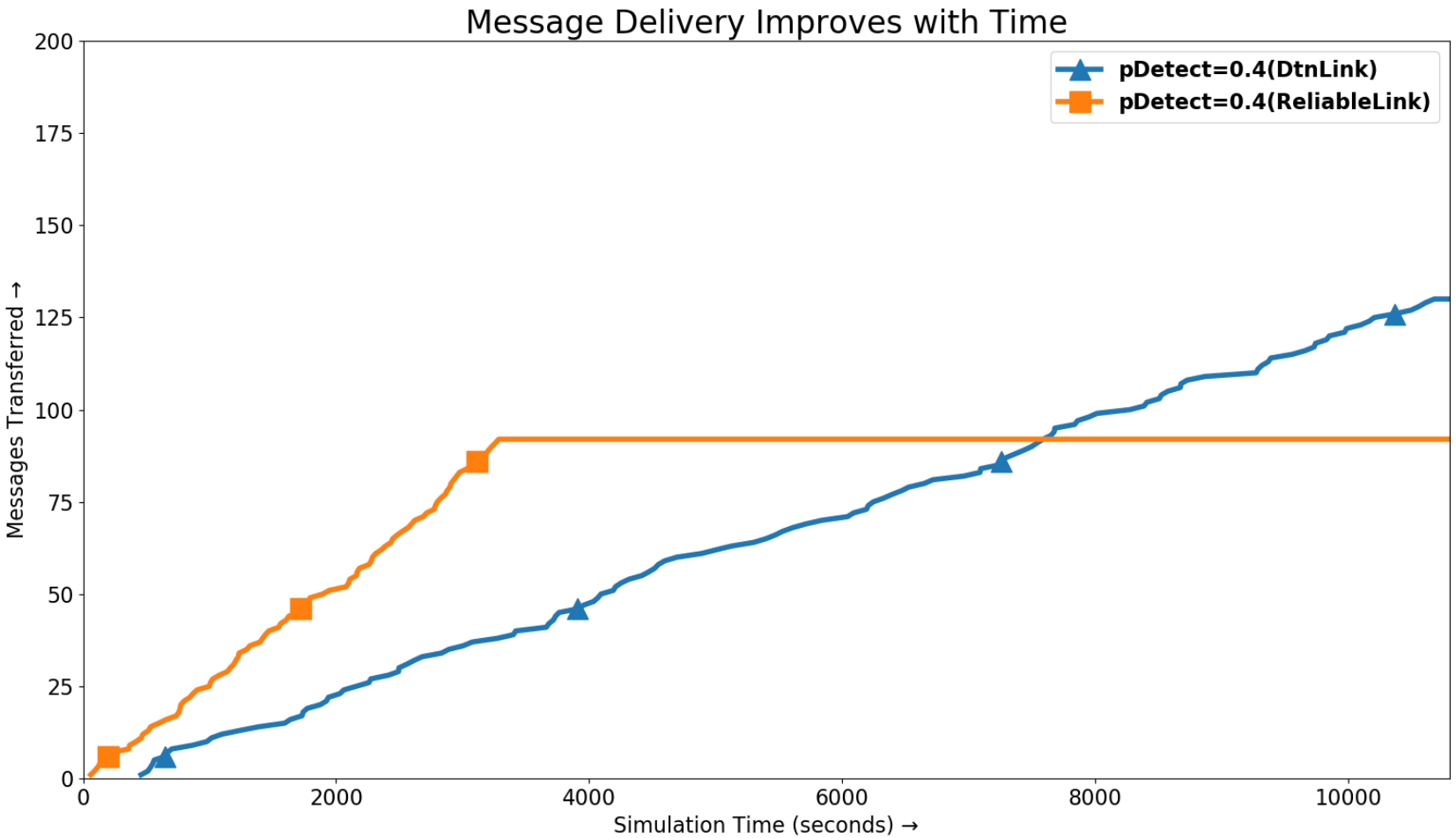
ReliableLink versus DtnLink, varying pDetection



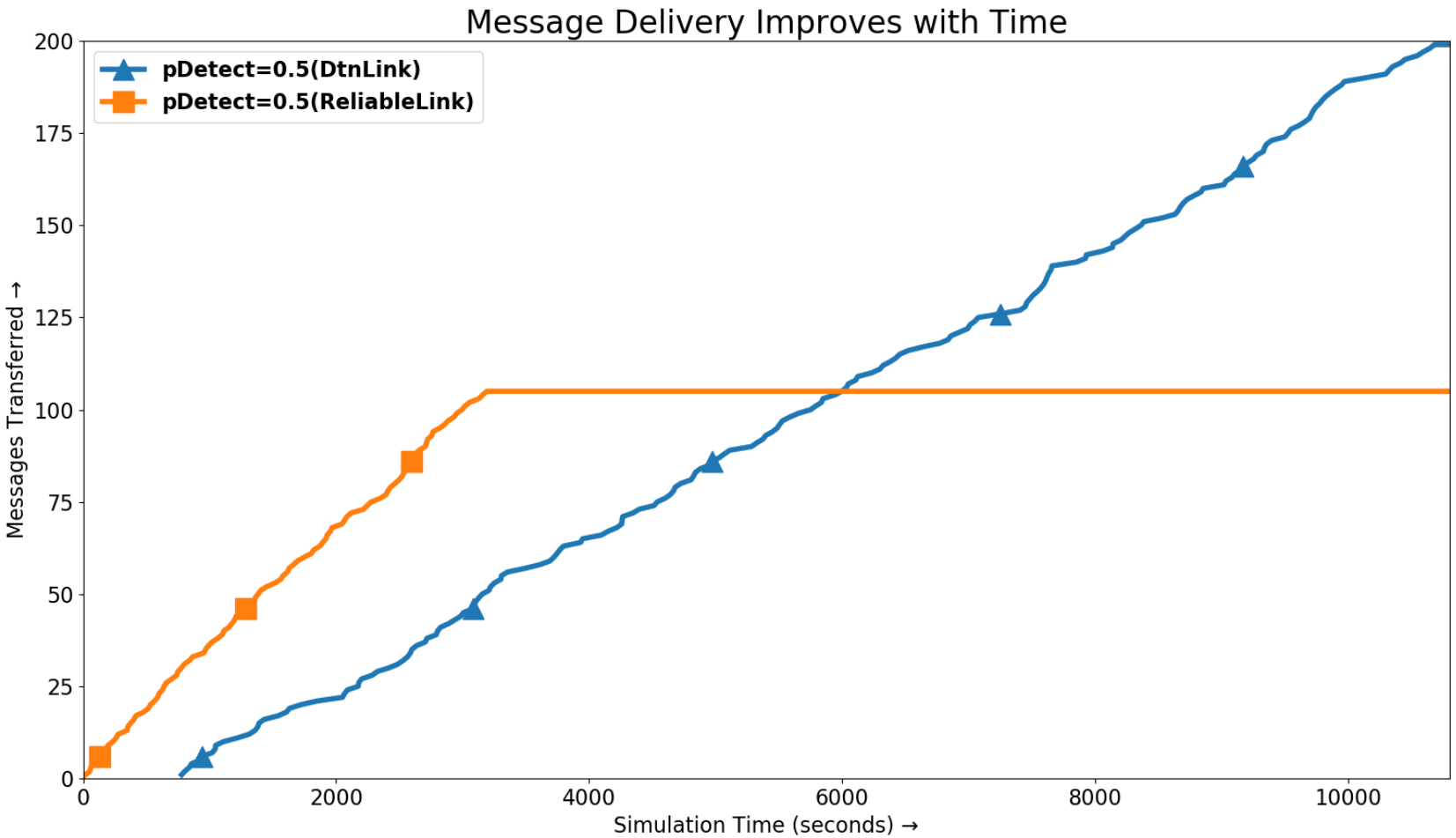
ReliableLink versus DtnLink, varying pDetection



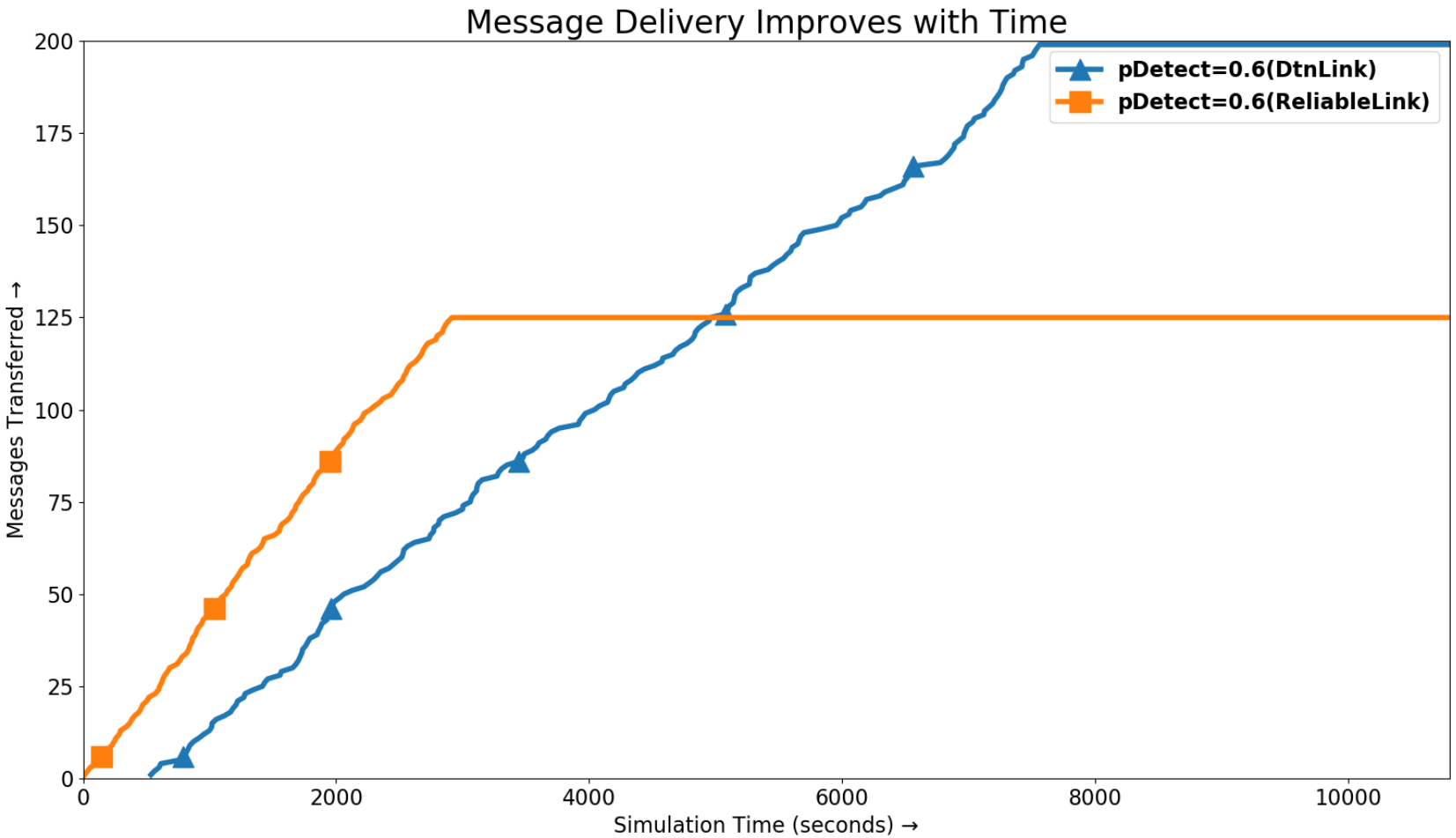
ReliableLink versus DtnLink, varying pDetection



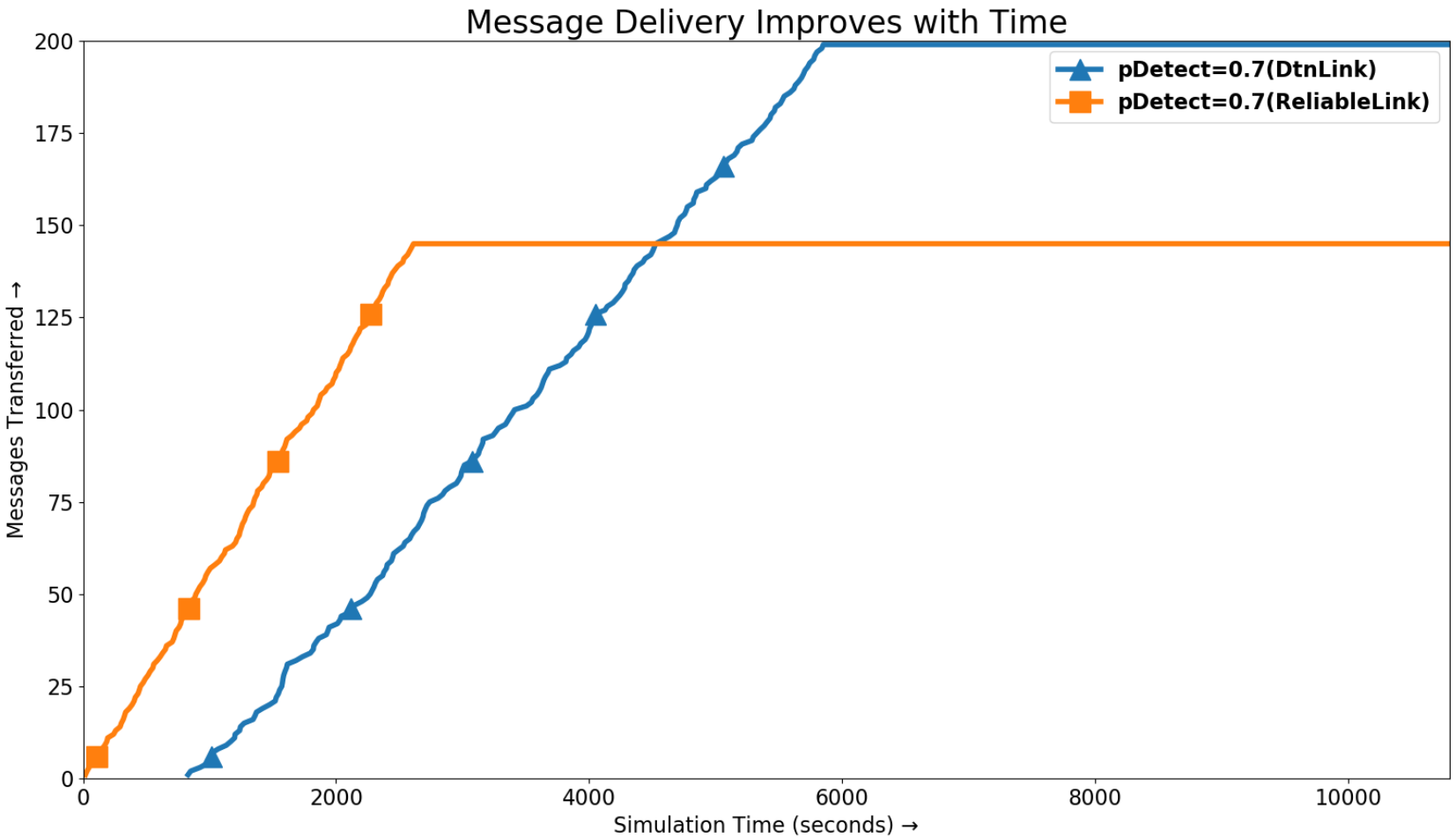
ReliableLink versus DtnLink, varying pDetection



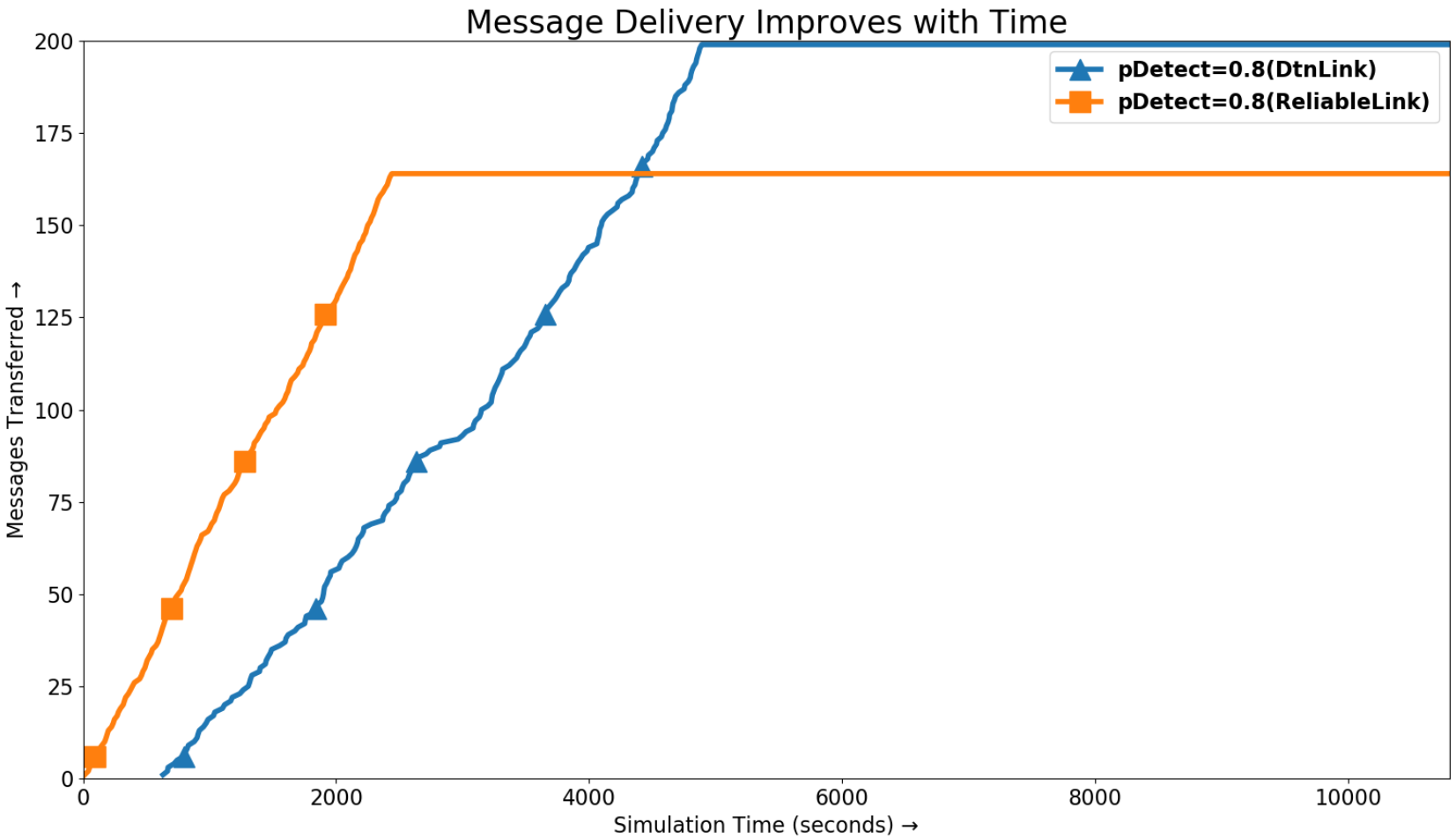
ReliableLink versus DtnLink, varying pDetection



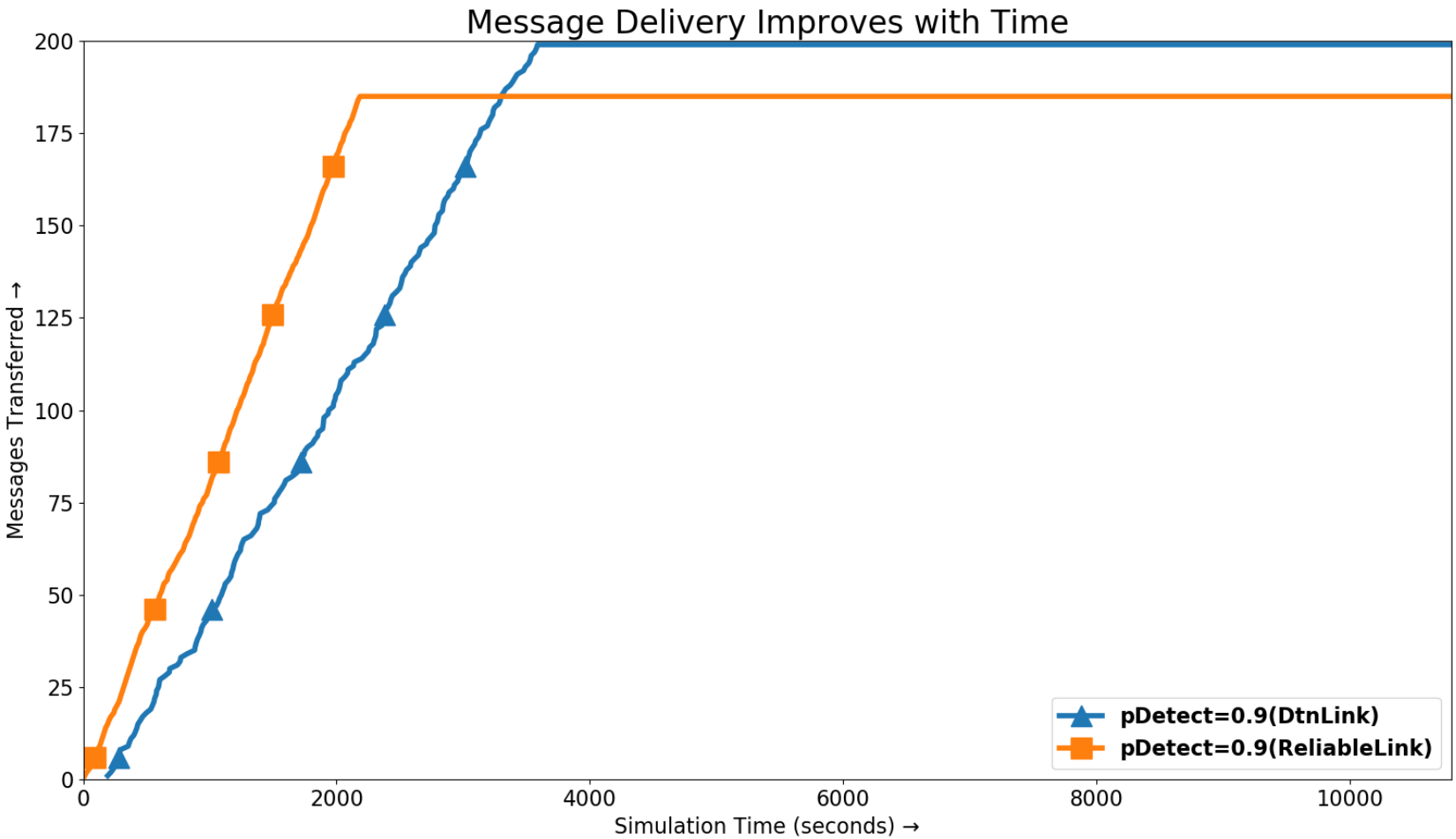
ReliableLink versus DtnLink, varying pDetection



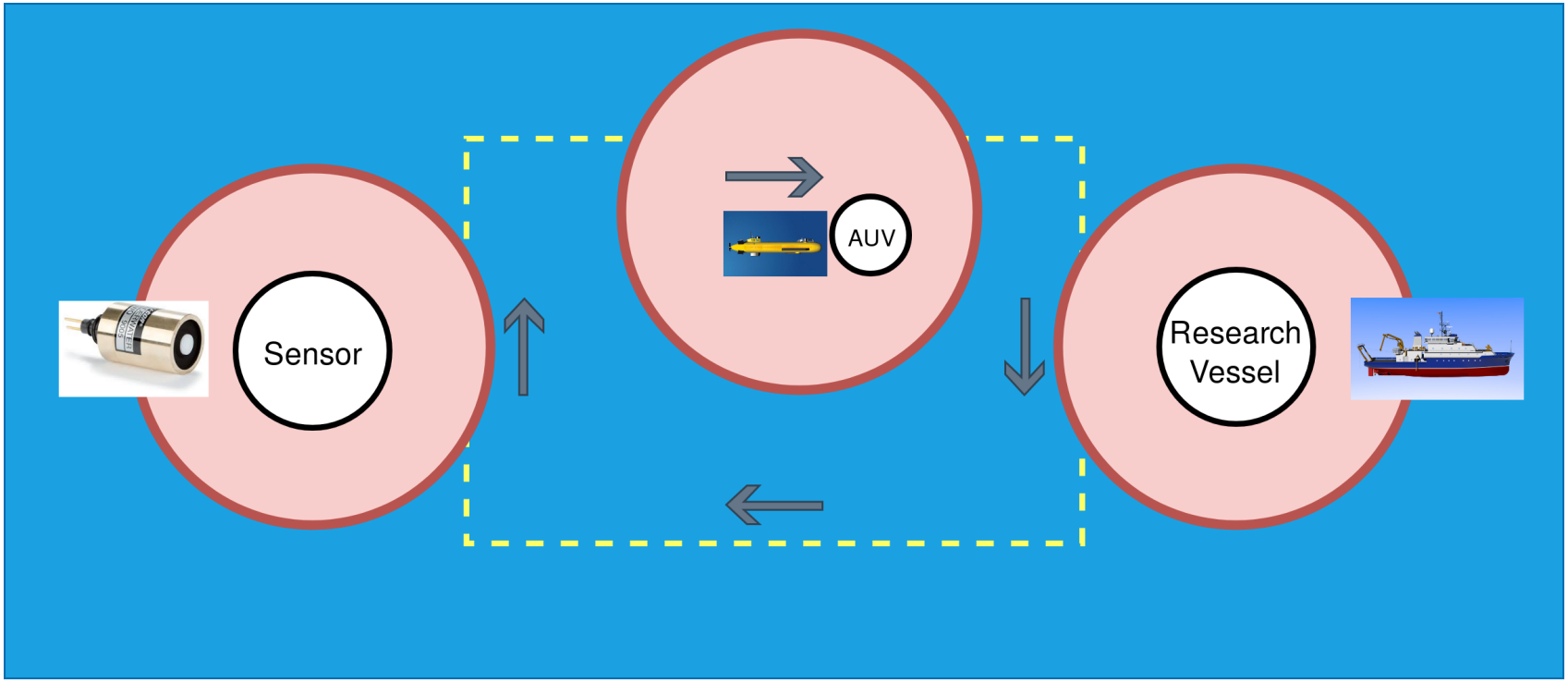
ReliableLink versus DtnLink, varying pDetection



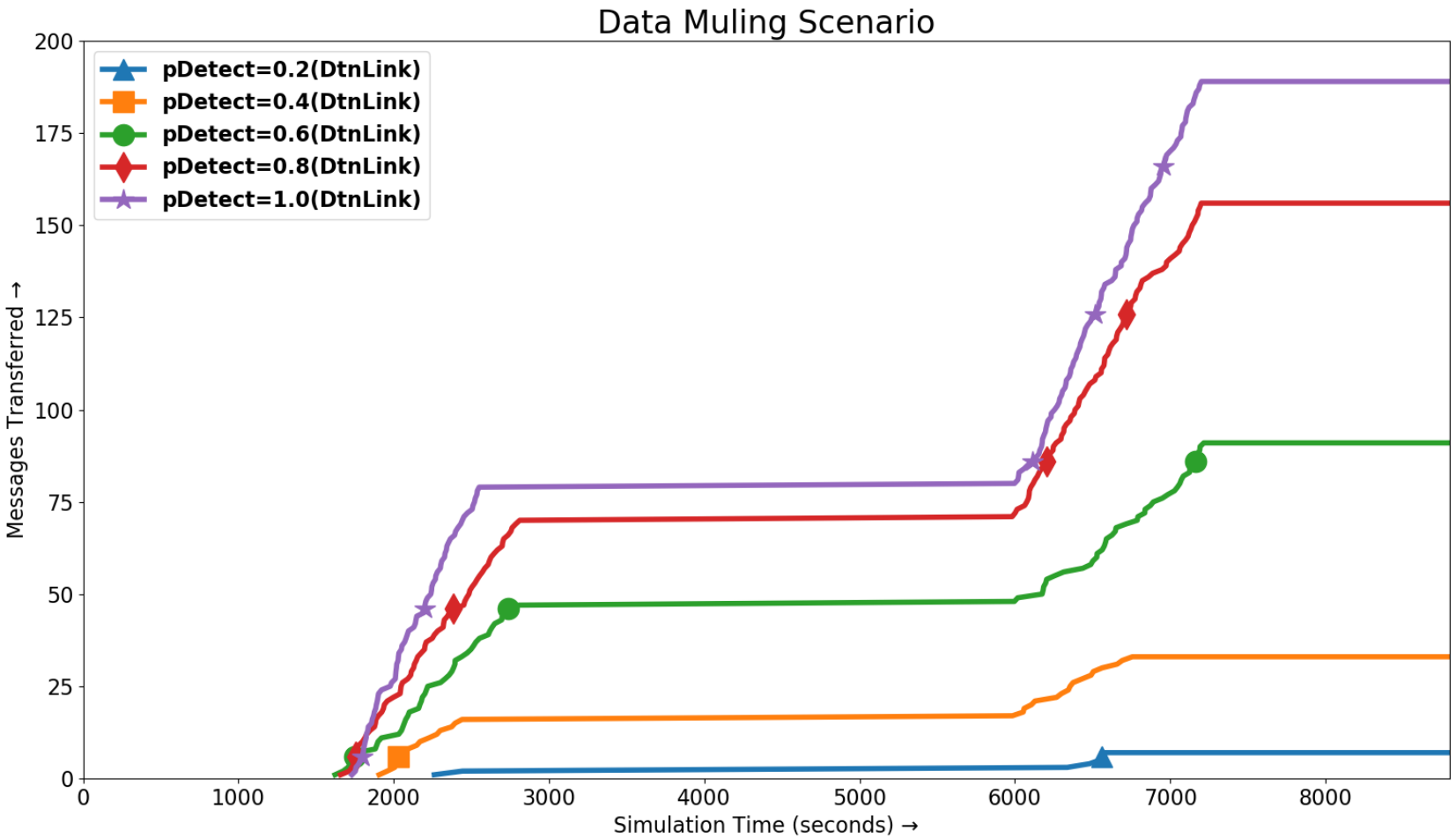
ReliableLink versus DtnLink, varying pDetection



Data Muling Simulation



Data Muling Results



Agenda

Why Unit Testing?

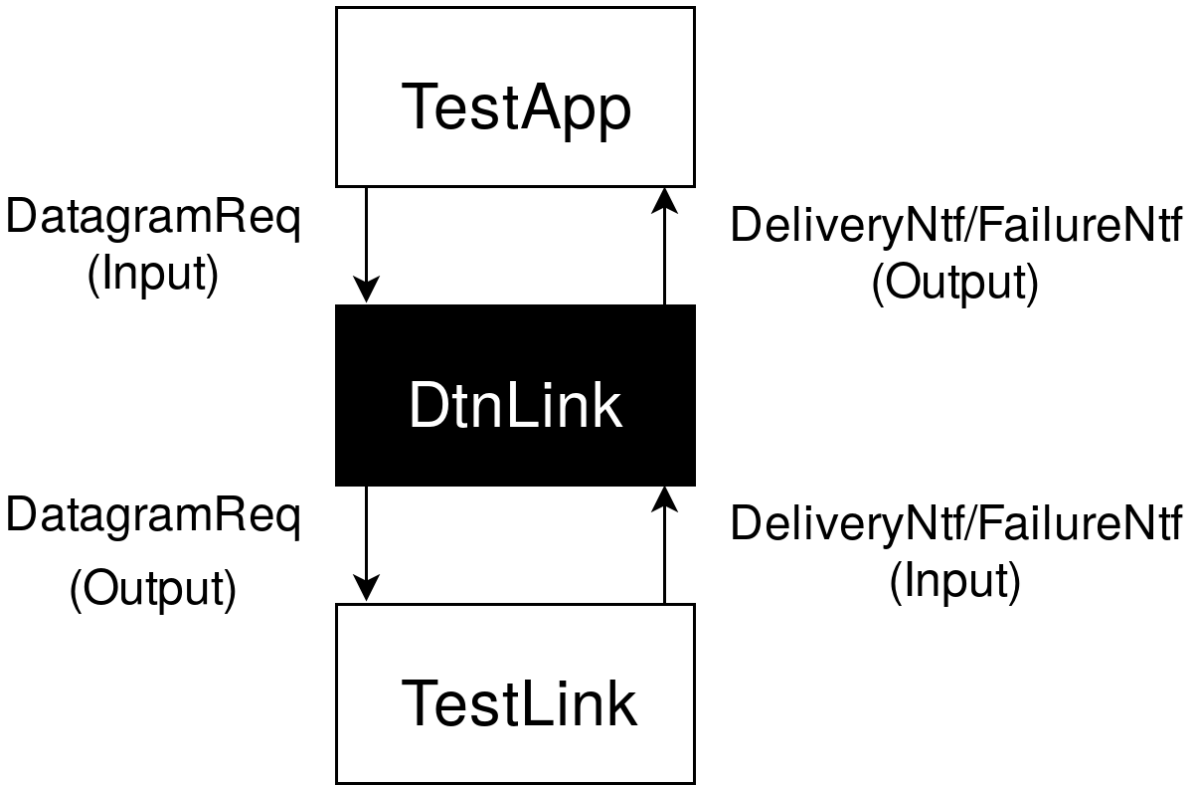
- Simulations are great for testing but...
  - they take a long time to run
  - make it hard to catch bugs
  - only test specific scenarios

Why Unit Testing?

- Simulations are great for testing but...
  - they take a long time to run
  - make it hard to catch bugs
  - only test specific scenarios
- Unit testing can help us by *automatically* testing crucial functionality of `DtnLink`
- *Regression testing* is checking for if anything breaks between changes

Black Box Testing

DtnLink Test Harness



Tests Conducted

- TRIVIAL_MESSAGE
- SUCCESSFUL_DELIVERY
- ROUTER_MESSAGE
- BAD_MESSAGE
- EXPIRY_PRIORITY
- ARRIVAL_PRIORITY
- RANDOM_PRIORITY
- LINK_TIMEOUT
- MULTI_LINK
- PAYLOAD_MESSAGE
- REBOOT

Agenda

Future Work

- `DtnLink` has some limitations which can be addressed in the future:
  - Stop-And-Wait is slow
  - TTL of messages doesn't include propagation delay
  - Multi-copy routing
  - End-to-end acknowledgements for multihop will need transport level control - `DtnTransport`

Agenda

Conclusions

- Underwater networks are more disrupted, need different protocols

Conclusions

- Underwater networks are more disrupted, need different protocols
- `DtnLink` can be useful when the channel medium is lossy and successful delivery is prioritised
    - Useful in data muling and switching between different links

Conclusions

- Underwater networks are more disrupted, need different protocols
- `DtnLink` can be useful when the channel medium is lossy and successful delivery is prioritised
  - Useful in data muling and switching between different links
- Stop-And-Wait sending reduces collisions but this can make it slower than `ReliableLink` in some situations

Conclusions

- Underwater networks are more disrupted, need different protocols
- `DtnLink` can be useful when the channel medium is lossy and successful delivery is prioritised
  - Useful in data muling and switching between different links
- Stop-And-Wait sending reduces collisions but this can make it slower than `ReliableLink` in some situations
- Simulations help us understand the use cases better

Thank you!