# Performant TCP for Low-Power Wireless Networks

**Sam Kumar**, Michael P Andersen, Hyung-Sin Kim, David E. Culler

*University of California, Berkeley*

IRTF Applied Networking Research Prize

Berkeley UNIVERSITY OF CALIFORNIA

riselab UC Berkeley

# Low-Power Wireless Personal Area Networks (LoWPANs)

~1999: LoWPAN research begins, eschewing the Internet architecture

S-MAC X-MAC Trickle

B-MAC WiseMAC

TinyOS

Contiki
The Open Source OS for the Internet of Things

~2008: IP introduced in LoWPANs

6LoWPAN
IPv6-based Low-power Wireless Personal Area Networks

~2012: IP becomes standard in LoWPANs

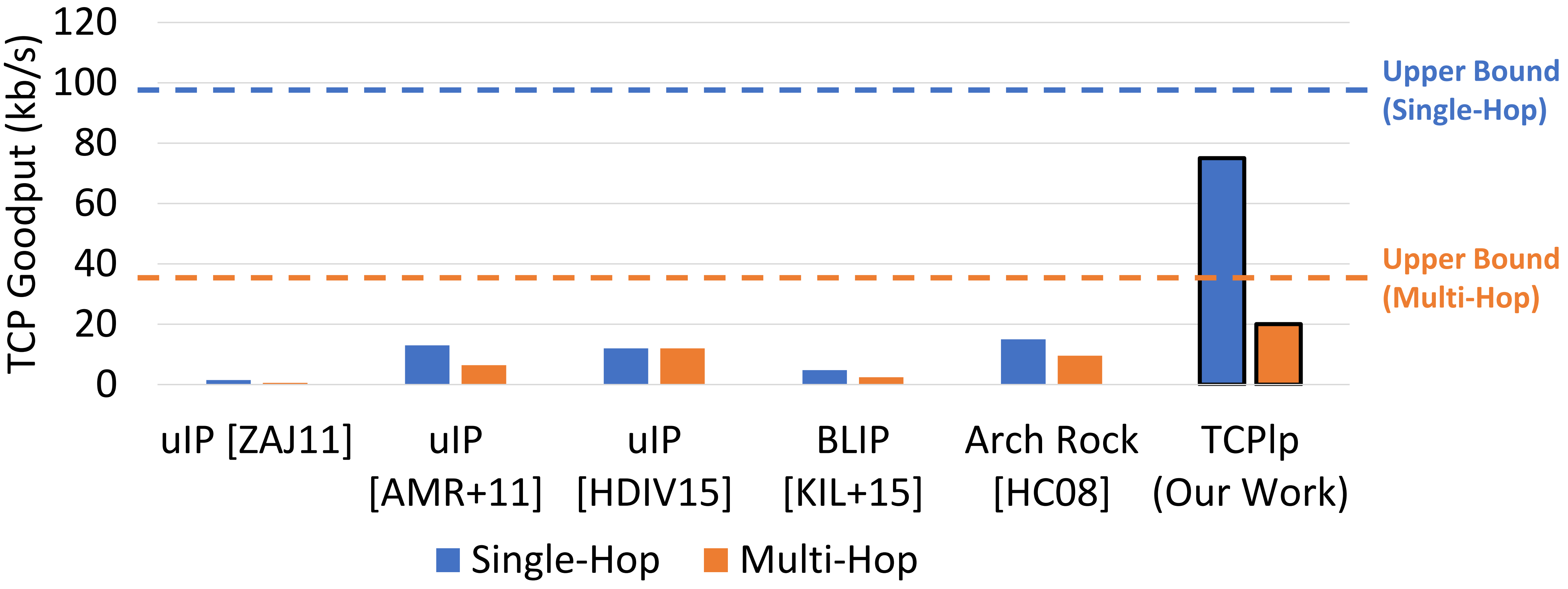
CoAP
RFC 7252

OPENTHREAD
released by Nest

**2020: Our Research**
We show how to make **TCP** work well in LoWPANs

**TCPlp**

# Making TCP work well in LoWPANs



# As of 2022, OpenThread Supports TCPlp!

OpenThread > Reference

## TCP

SEND FEEDBACK

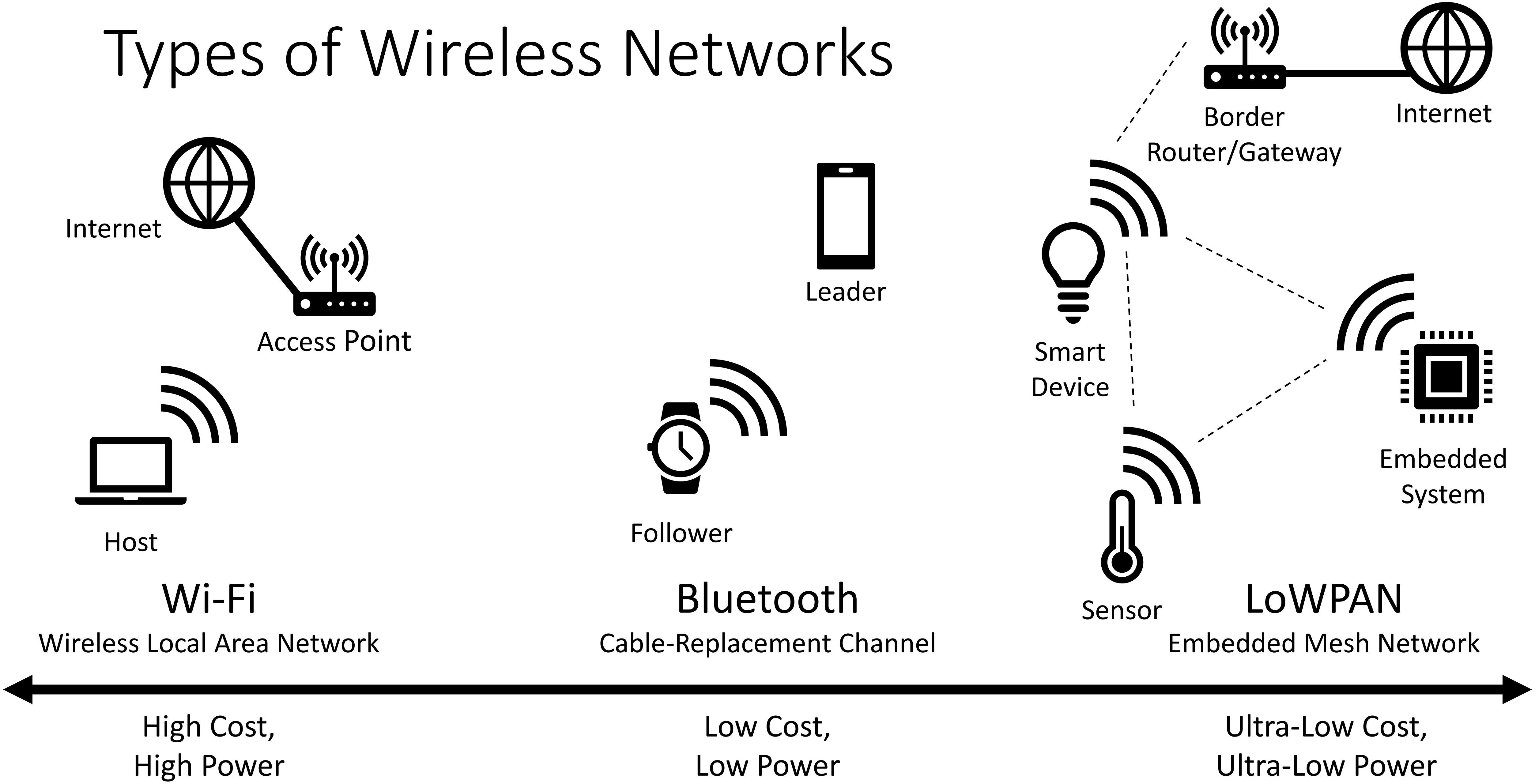
This module includes functions that control TCP communication.

### Summary

| Enumerations | |
|---|---|
| `anonymous enum` | enum<br>This enumeration defines flags passed to otTcpConnect(). |
| `anonymous enum` | enum<br>This enumeration defines flags passed to `otTcpSendByReference`. |

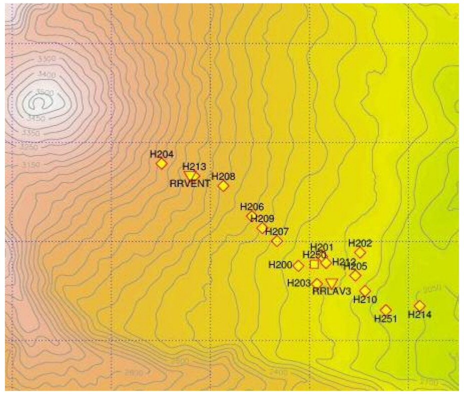
# What is a LoWPAN?

LoWPAN = Low-Power Wireless Personal Area Network

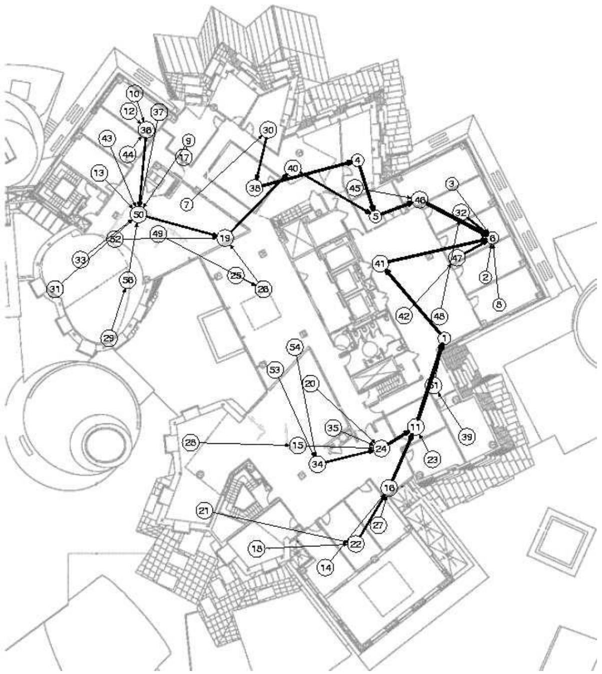

Types of Wireless Networks
Internet
Access Point
Host
Wi-Fi
Wireless Local Area Network
Leader
Follower
Bluetooth
Cable-Replacement Channel
Border
Router/Gateway
Internet
Smart
Device
Embedded
System
Sensor
LoWPAN
Embedded Mesh Network
High Cost,
High Power
Low Cost,
Low Power
Ultra-Low Cost,
Ultra-Low Power

# What are LoWPANs used for?

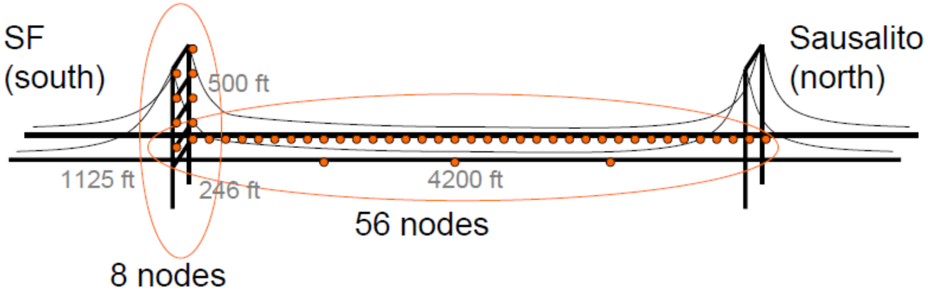
Smart grid [4]



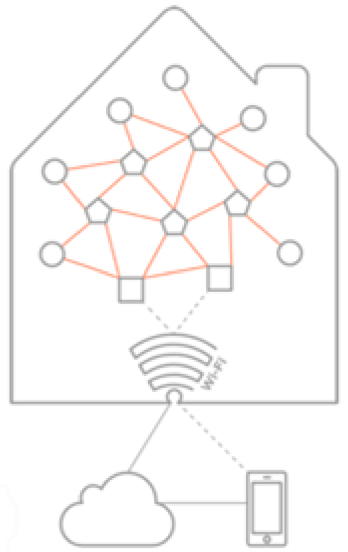
Volcano monitoring [1]



Indoor environment [3]




Structural monitoring [2]



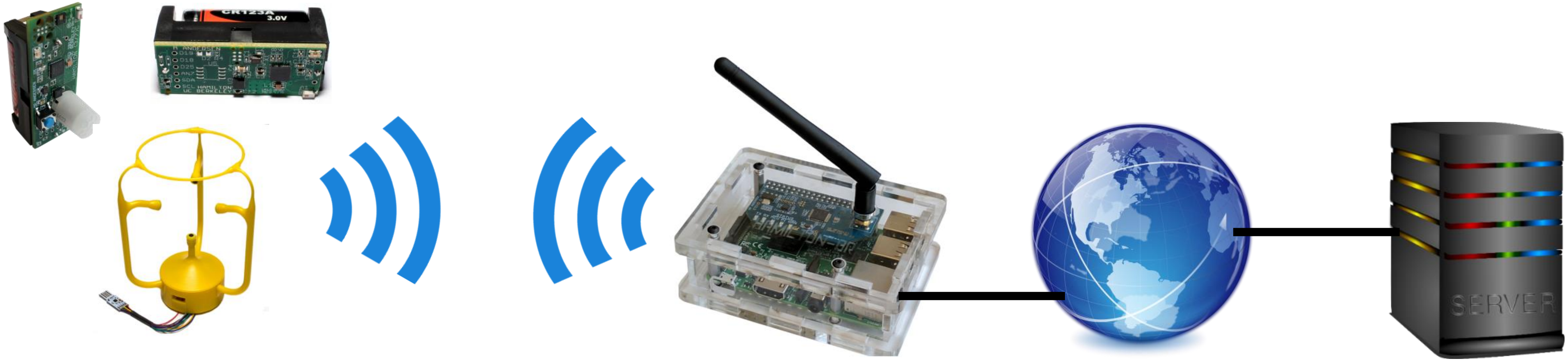
Smart home and IoT [5]

[1] Werner-Allen, G., Lorincz, K., Johnson, J., Lees, J., & Welsh, M. Fidelity and yield in a volcano monitoring sensor network. In OSDI 2006.
[2] Kim, S., Pakzad, S., Culler, D., Demmel, J., Fenves, G., Glaser, S., & Turon, M. Health monitoring of civil infrastructures using wireless sensor networks. In IPSN 2007.
[3] Hull, B., Jamieson, K., & Balakrishnan, H. Mitigating congestion in wireless sensor networks. In SenSys 2004.
[4] https://www.cisco.com/c/en/us/products/collateral/routers/1000-series-connected-grid-routers/datasheet-c78-741312.html
[5] https://www.automatedhome.co.uk/new-products/thread-a-new-wireless-networking-protocol-for-the-home.html

# Why use TCP in a LoWPAN?

LoWPAN = Low-Power Wireless Personal Area Network

# LoWPANs use *Gateway-Based Architectures*




LoWPAN
Embedded Devices

Application-Layer
Gateway

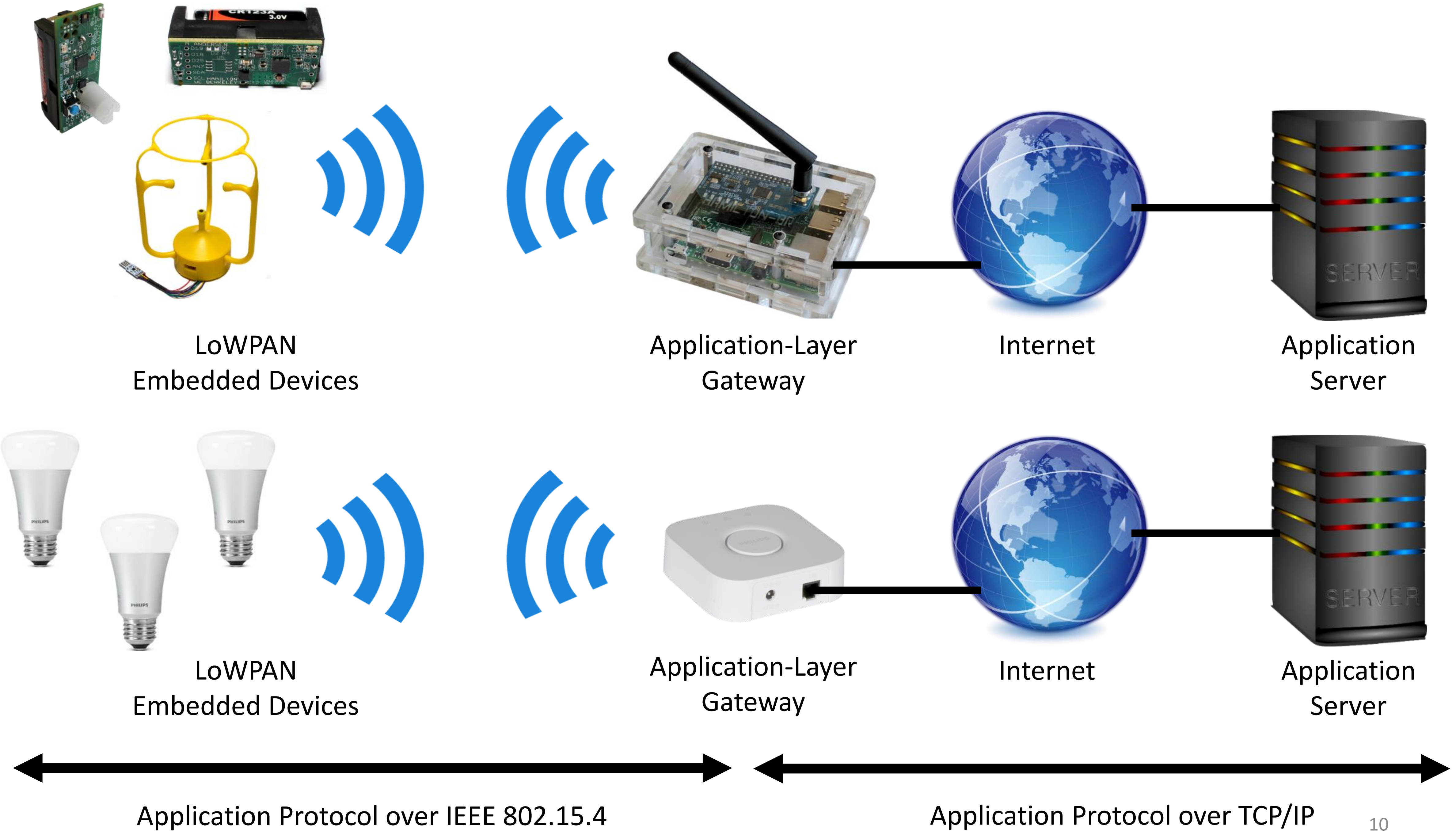
Internet

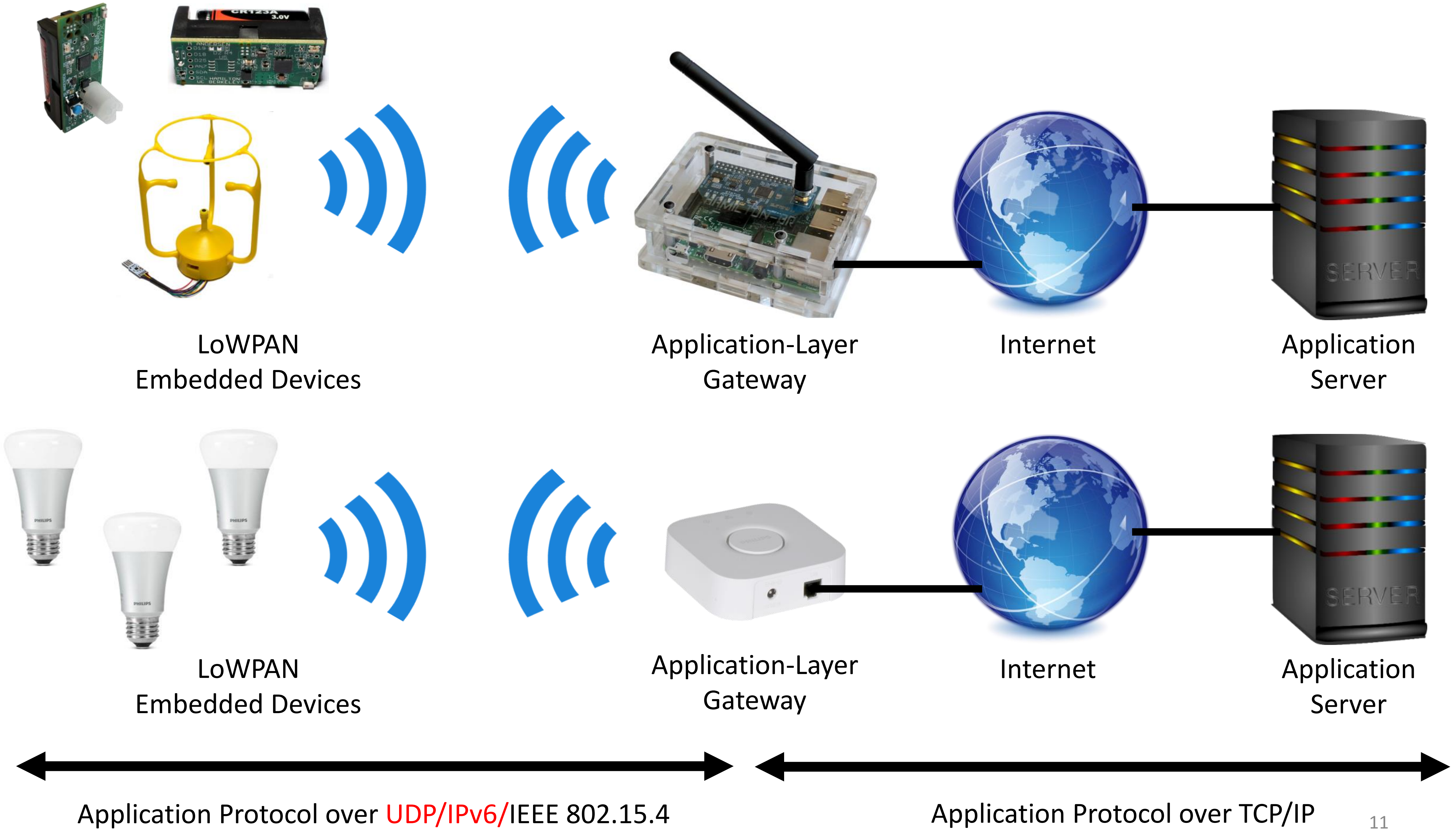
Application
Server

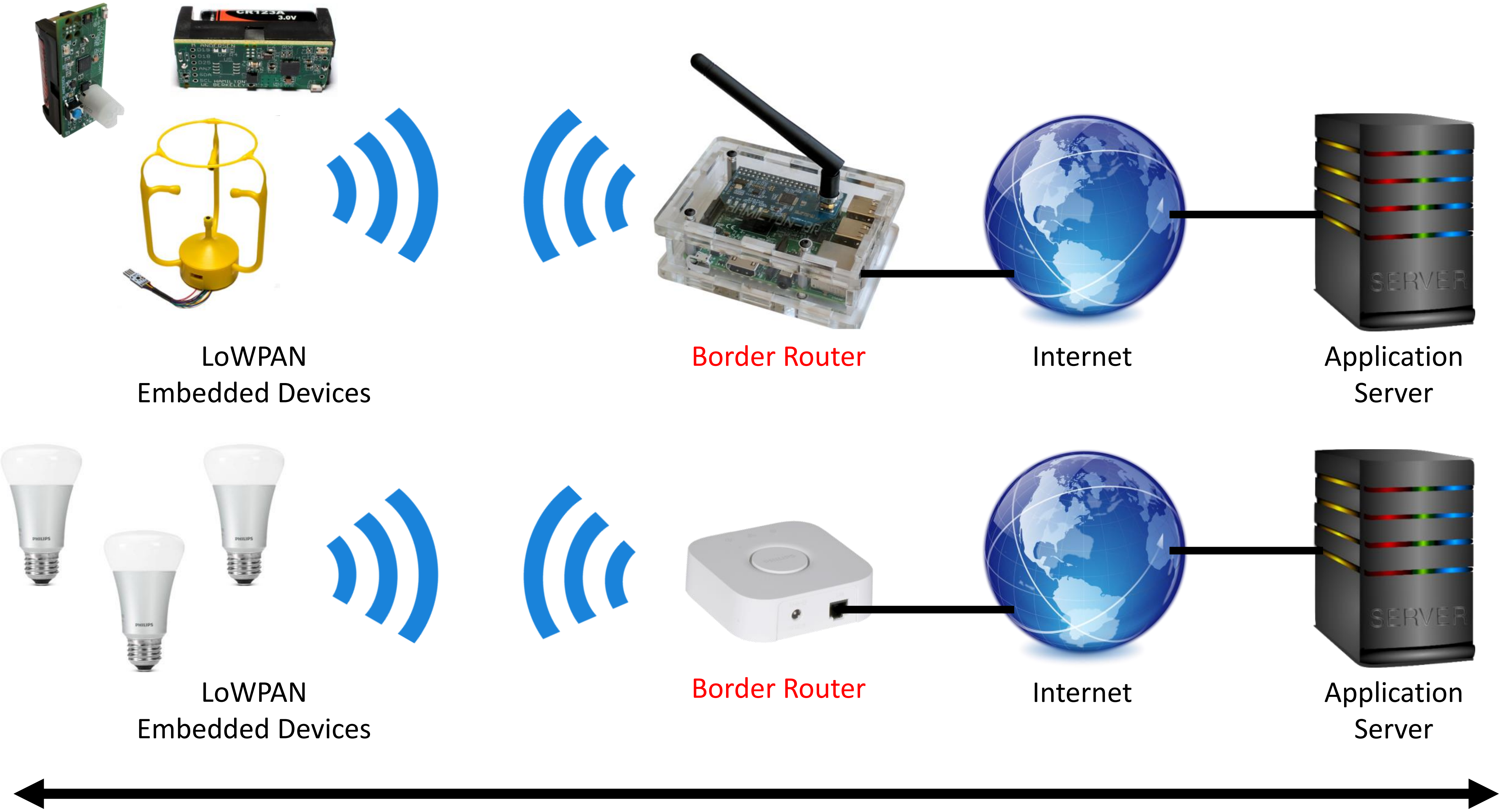
Application Protocol over IEEE 802.15.4

Application Protocol over TCP/IP

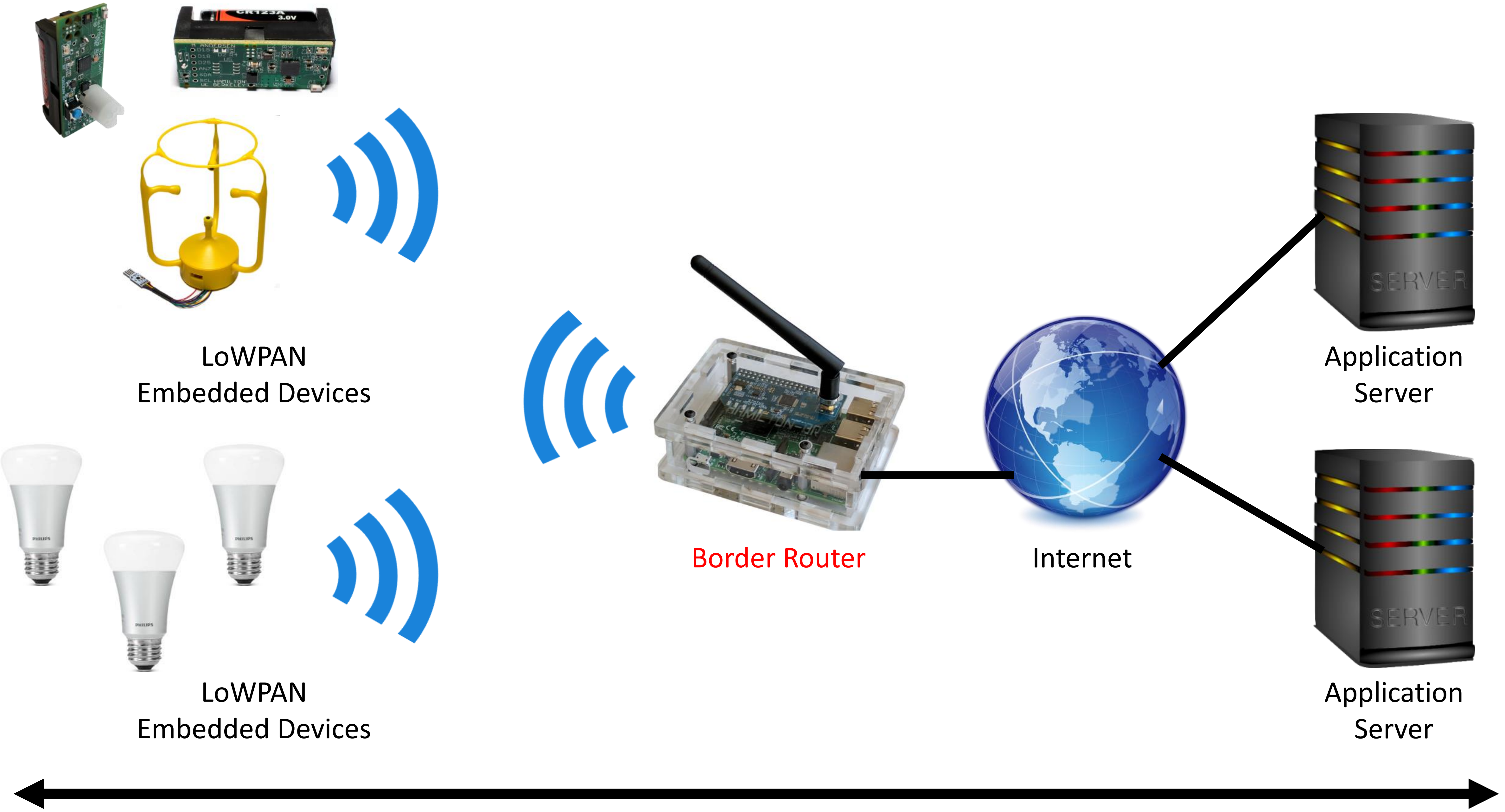
**Gateway-based architecture limits interoperability**


LoWPAN
Embedded Devices
Application-Layer
Gateway
Internet
Application
Server
SERVER
LoWPAN
Embedded Devices
Application-Layer
Gateway
Internet
Application
Server
SERVER
Application Protocol over IEEE 802.15.4
Application Protocol over TCP/IP


LoWPAN
Embedded Devices
Application-Layer
Gateway
Internet
Application
Server
LoWPAN
Embedded Devices
Application-Layer
Gateway
Internet
Application
Server
Application Protocol over UDP/IPv6/IEEE 802.15.4
Application Protocol over TCP/IP




**Want End-to-End TCP/IP**




**Want End-to-End TCP/IP**

# Why are LoWPANs Challenging for TCP?

LoWPAN = Low-Power Wireless Personal Area Network

# Challenges of Low-Power Networks

## Resource Constraints

- Limited CPU/RAM

## Link-Layer Constraints

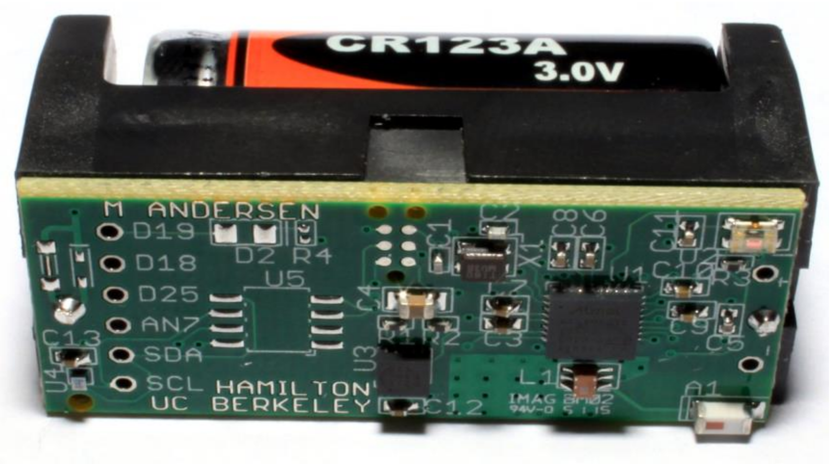
- Small MTU
- Low wireless range
  - *Multi-hop* wireless

## Energy Constraints

- *Duty-cycled* radio

# Low-Power Embedded Devices

- **32 KiB** Data Memory (RAM)
- **250 kb/s IEEE 802.15.4** radio
- 32-bit ARM Cortex M0+ @ 48 MHz
- 256 KiB Code Memory (ROM)




← ≈ 5 centimeters →

Hamilton Sensor Platform [KACKZMC18]

**Q: How should devices like these connect to the Internet?**

**We show TCP/IP works well**

# LoWPAN Research has Steered Clear of TCP

- “TCP is **not light weight** … and may not be suitable for implementation in low-cost sensor nodes with limited processing, memory, and energy resources.”
- That “TCP is a **connection-oriented** protocol” is a poor match for WSNs, “where actual data might be only in the order of a few bytes.”
- “TCP uses a single packet drop to infer that the network is **congested**.” This “can result in extremely poor transport performance because wireless links tend to exhibit **relatively high packet loss rates**.”

# LoWPAN Research has Steered Clear of TCP

**Expected Reasons for Poor Performance:**

- TCP is too heavy
- TCP’s features aren’t necessary and bring additional overhead
- TCP performs poorly in the presence of wireless loss

# Finding: TCP Can Perform Well in LoWPANs

**We show why these don't actually apply**

**Expected Reasons for Poor Performance:**

- TCP is too heavy
- TCP's features aren't necessary and bring additional overhead
- TCP performs poorly in the presence of wireless loss
- **These would be *fundamental***

**We show how to address these issues**

**Actual Reasons for Poor Performance:**

- LoWPANs have a small L2 frame size → high header overhead
- Hidden terminals
- Link-layer scheduling not designed with TCP in mind
- **These problems are *fixable within the paradigm of TCP!***

# Roadmap

1. **Overview**

2. Why the expected reasons for poor TCP performance don’t apply

3. Addressing the actual reasons for poor performance

4. Evaluation and conclusions

# Roadmap

1. Overview
2. **Why the expected reasons for poor TCP performance don't apply**
3. Addressing the actual reasons for poor performance
4. Evaluation and conclusions

# Overview of Techniques

| | |
|---|---|
| Resource Constraints | • Zero-Copy Send Buffer<br>• In-Place Reassembly Queue |
| Link-Layer Constraints | • Atypical Maximum Segment Size<br>• Link Retry Delay |
| Energy Constraints | • Adaptive Duty Cycle<br>• Link-Layer Queue Management |

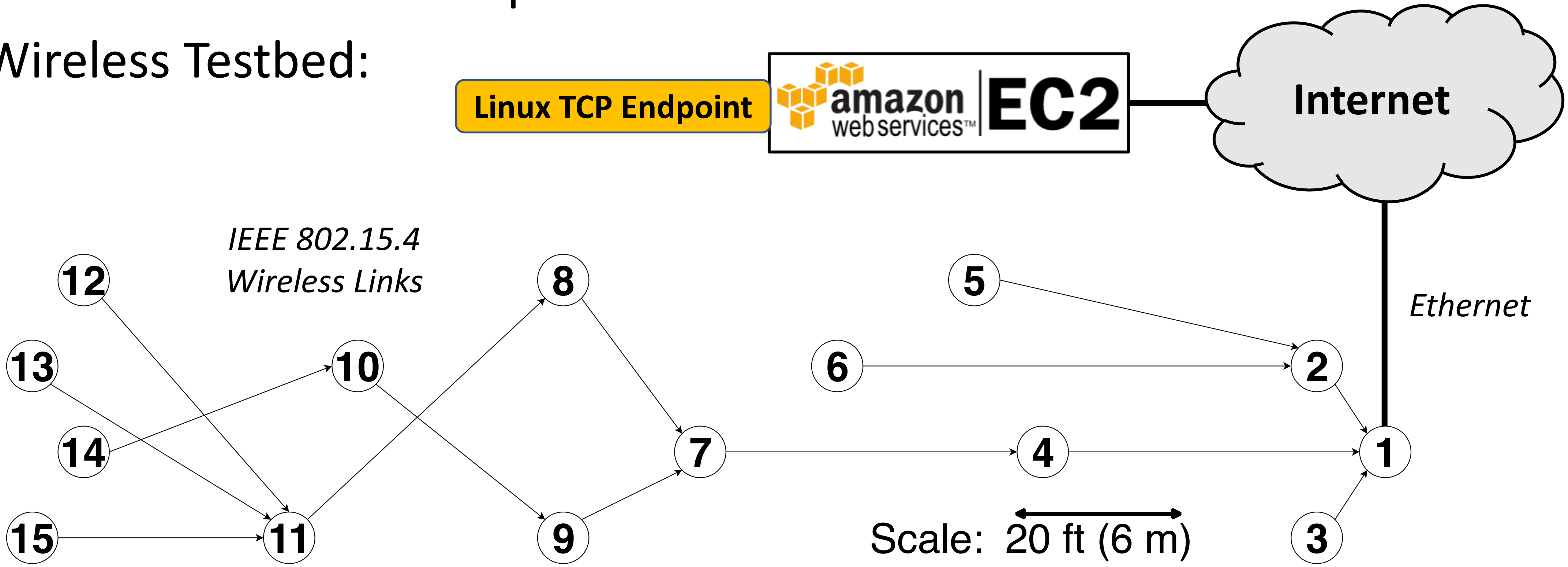
# Focus of this Section of the Talk

| | |
|---|---|
| Resource Constraints | • Zero-Copy Send Buffer<br>• **In-Place Reassembly Queue** |
| Link-Layer Constraints | • **Atypical Maximum Segment Size**<br>• Link Retry Delay |
| Energy Constraints | • Adaptive Duty Cycle<br>• Link-Layer Queue Management |

# Experimental Methodology

- ARM Cortex-M0+
- 32 KiB RAM

- Nodes based on Hamilton Platform (SAMR21)
- Use RIOT-OS with the OpenThread network stack
- Wireless Testbed:



# Implementation of TCP

- Start with the mature, full-scale TCP implementation in FreeBSD
- Re-engineer key parts for the embedded platform
- Resulting implementation: *TCPlp*

**Known TCP Implementation Problems**

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice



Table of Contents



RFC 2525: Known TCP Implementation Problems

# Resource Consumption of TCPlp

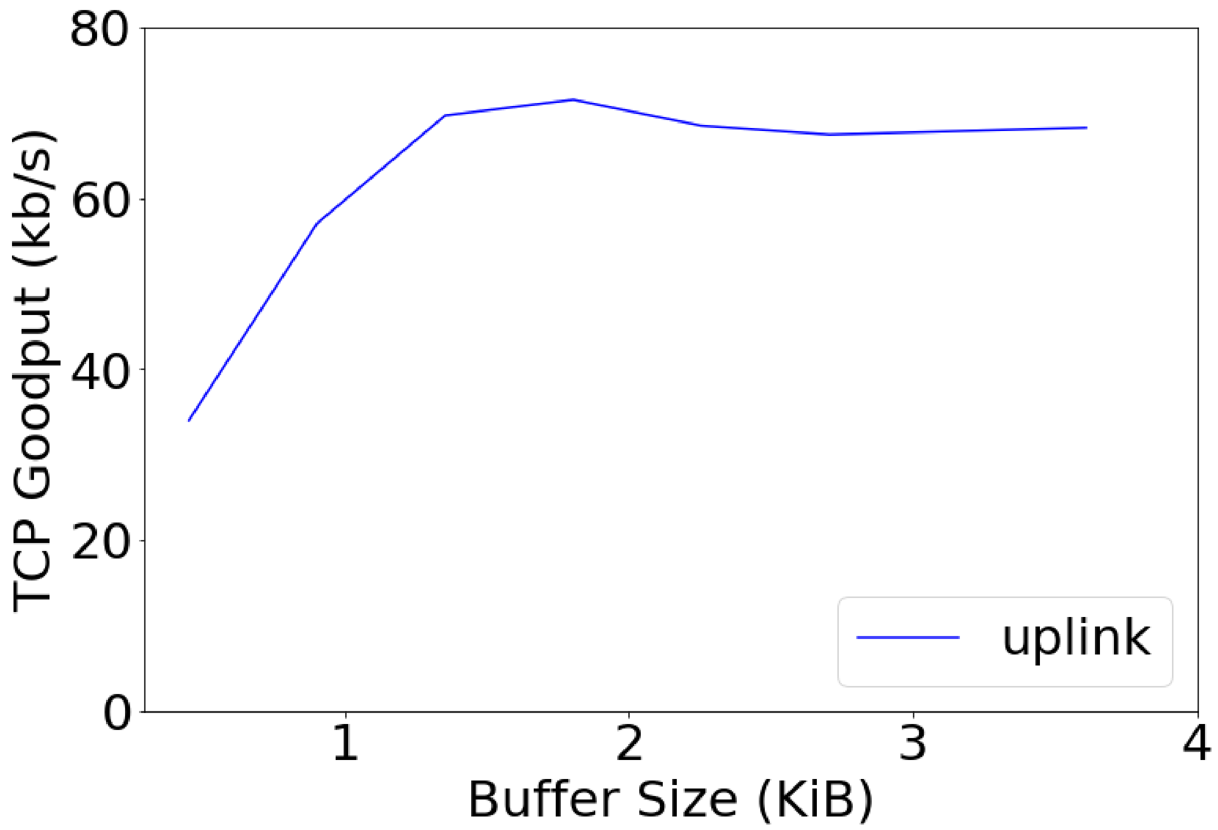
- TCPlp requires:
  - ≈ 32 KiB of code memory (ROM)
  - ≈ 0.5 KiB of data memory (RAM) per connection

- Hamilton platform has:
  - 256 KiB of code memory (ROM)
  - 32 KiB of data memory (RAM)

- Optimization in TCPlp: use separate structures for *active sockets* and *passive sockets*

# How Large do TCP Buffers Need to Be?

- Bandwidth-Delay Product (BDP)
- Empirical BDP: ≈ 2-3 KiB

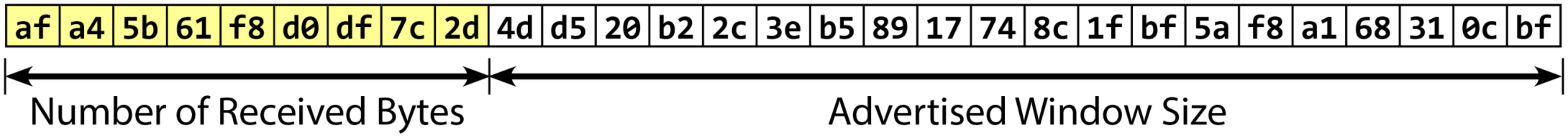



**TCP, including buffers, can fit comfortably in memory**

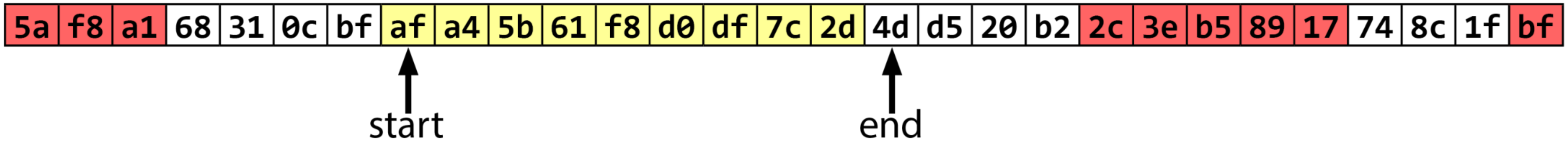
# TCPlp’s Receive and Reassembly Buffers

- Naïve strategy: separate buffers for receive and reassembly queues

- Observation: advertised window size decreases with size of buffered data

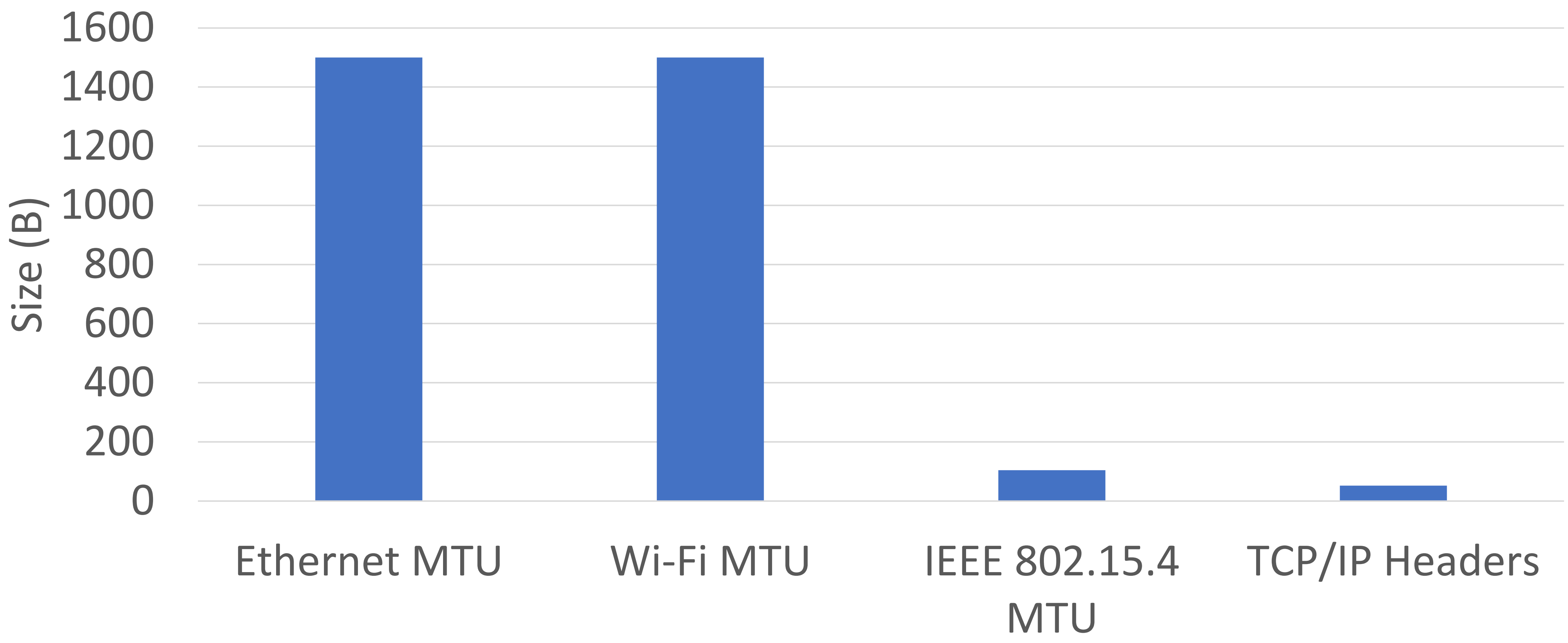


# TCPlp's In-Place Reassembly Queue




- In-sequence data is yellow
  - Use circular buffer to keep track of which bytes contain in-sequence data
- Out-of-order data is red
  - Use bitmap to keep track of which bytes contain out-of-order data

# MTU and Header Sizes



# Large Header Overhead

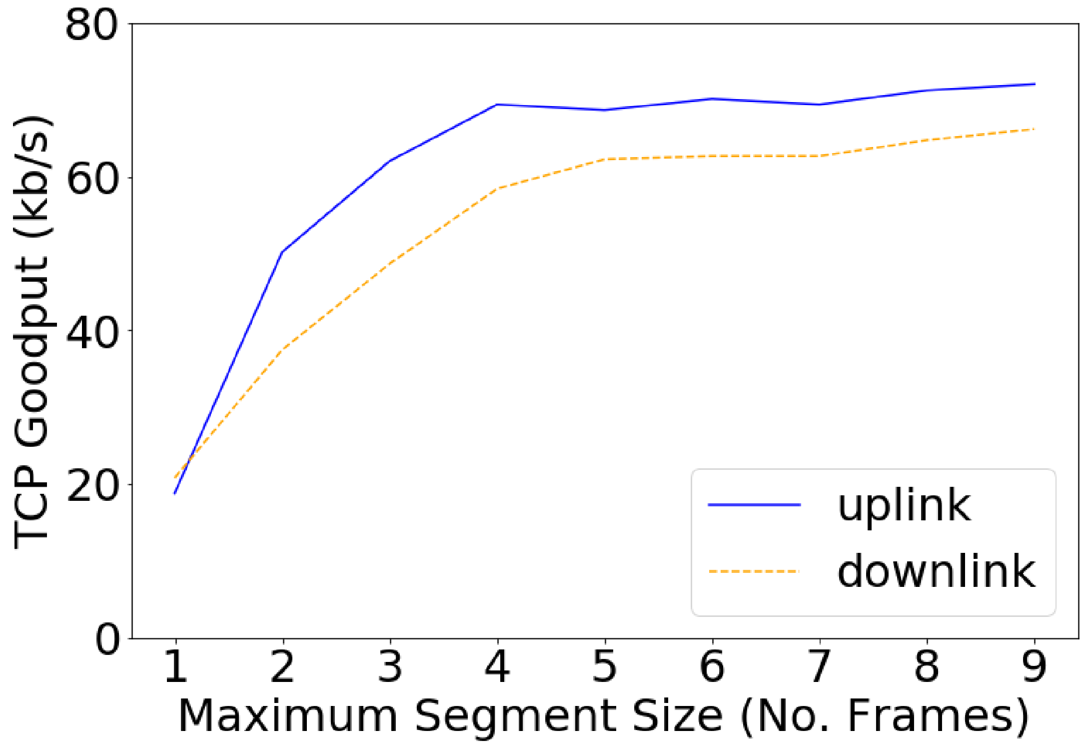
- Normally, TCP segments are chosen to be as large as the link supports, but no larger
- IEEE 802.15.4 MTU is only 104 bytes (excluding link-layer header)
- TCP/IP headers are > 52 bytes

# Managing Large Header Overhead

- ~~Normally, TCP segments are chosen to be as large as the link supports, but no larger~~
- TCPlp allows TCP segments to span multiple link-layer frames
- 6LoWPAN handles fragmentation and reassembly

# Choosing the Maximum Segment Size

- Idea: allow TCP segments to span multiple link-layer frames
- A 3-5 frame MSS substantially amortizes header overhead
- Stateful TCP header compression could potentially result in even greater gains

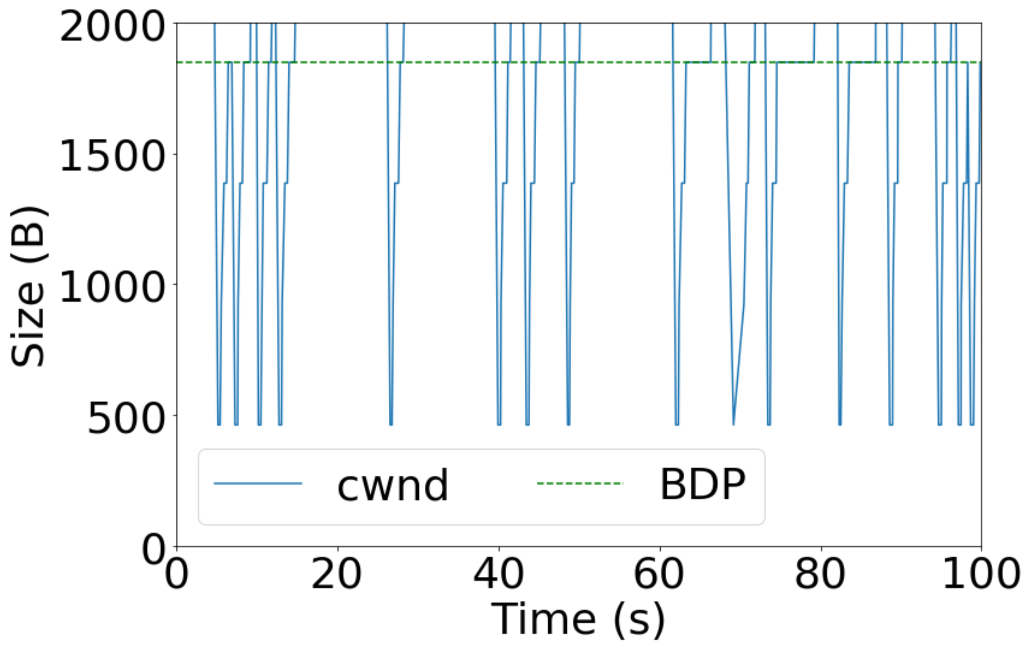


# How Many In-Flight Segments?

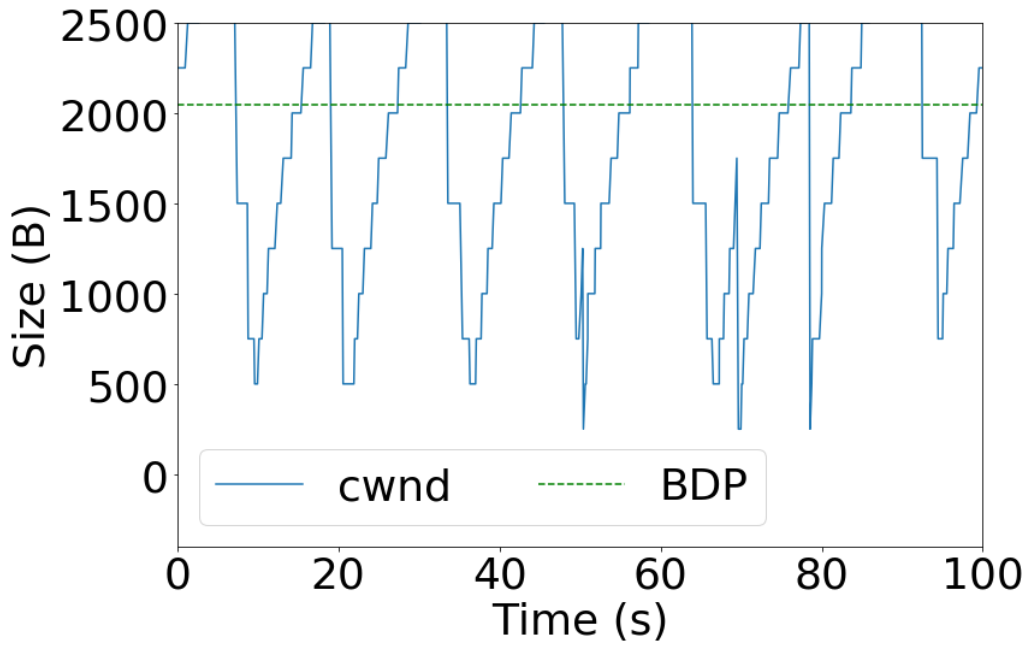
- Bandwidth-delay product is 2-3 KiB
- Each segment is ≈ 250 B to 500 B

- ≈ 4 to 12 in-flight TCP segments
- **This affects TCP's congestion control**

# TCP New Reno in a LoWPAN




**MSS = 462 B**




**MSS = 250 B, RED/ECN**

- Congestion window recovers to BDP quickly (because BDP is small)

**TCP in a LoWPAN is more resilient to wireless losses**

# Roadmap

1. Overview
2. Why the expected reasons for poor TCP performance don’t apply
3. **Addressing the actual reasons for poor performance**
4. Evaluation and conclusions

# Focus of this Section of the Talk

| | |
|---|---|
| Resource Constraints | • Zero-Copy Send Buffer<br>• In-Place Reassembly Queue |
| Link-Layer Constraints | • Atypical Maximum Segment Size<br>• **Link Retry Delay** |
| Energy Constraints | • **Adaptive Duty Cycle**<br>• Link-Layer Queue Management |

# Duty-Cycling the Radio

- The *duty cycle* is the proportion of time that the radio is listening or transmitting
- OpenThread uses a *receiver-initiated* duty cycle protocol

# Receiver-Initiated Radio Duty Cycle


**B**attery-Powered Node
Radio is Duty-Cycled
(“Sleepy End Device” in OpenThread)

**W**all-Powered Node
Radio is Always On
(“Router” in OpenThread)

- Packets can be sent to W at any time

# Receiver-Initiated Radio Duty Cycle


**B**attery-Powered Node
Radio is Duty-Cycled
("Sleepy End Device" in OpenThread)

**W**all-Powered Node
Radio is Always On
("Router" in OpenThread)

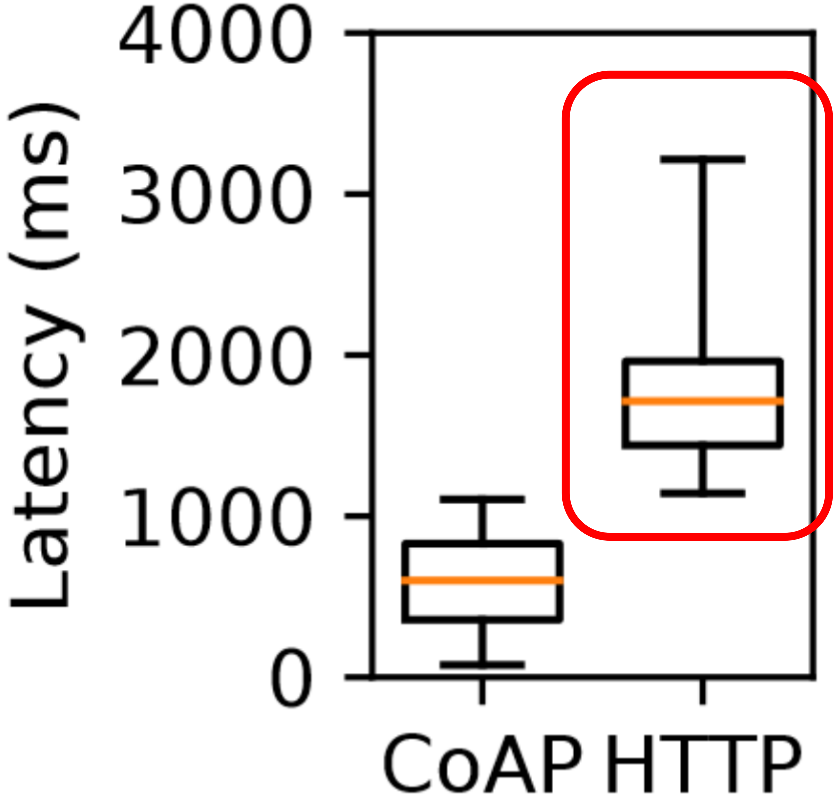
- Packets can be sent to W at any time
- Packets for B wait until B is listening

**B's idle duty cycle is determined by how frequently it sends DataReqs**

# How does Radio Duty Cycle affect TCP?

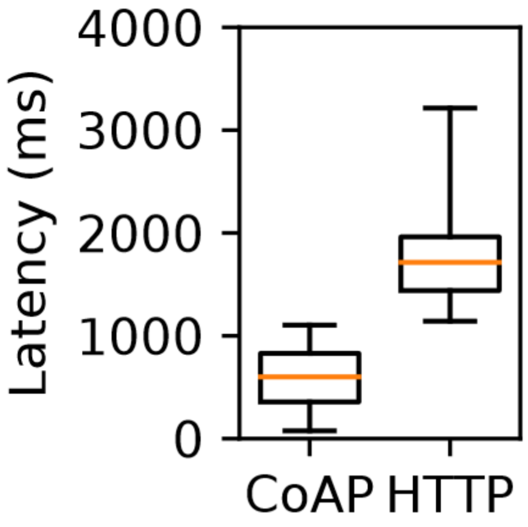
- Let's compare HTTP/TCP to CoAP
- Setup: B sends W a DataReq frame every 1000 ms

- HTTP request requires *two* round trips
- CoAP request requires *one* round trip

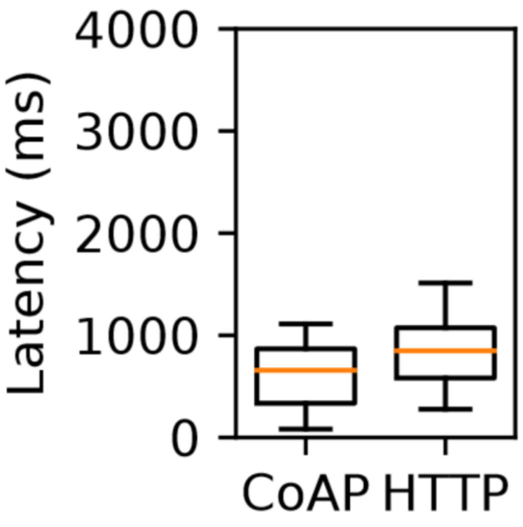


# Solution: *Adaptive* Radio Duty Cycle

- Use HTTP/TCP protocol state to adapt the duty cycle
- Send DataReqs more frequently when a packet is expected

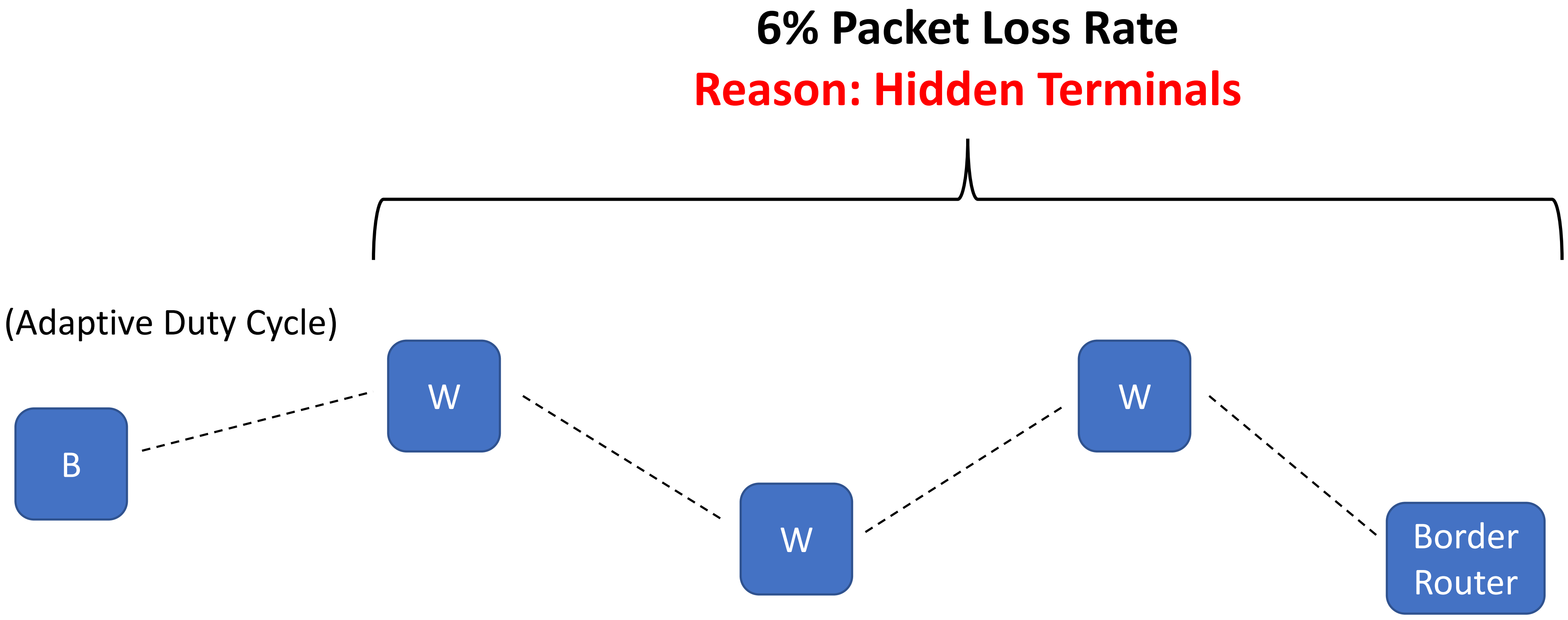



Without Adaptive Duty Cycle

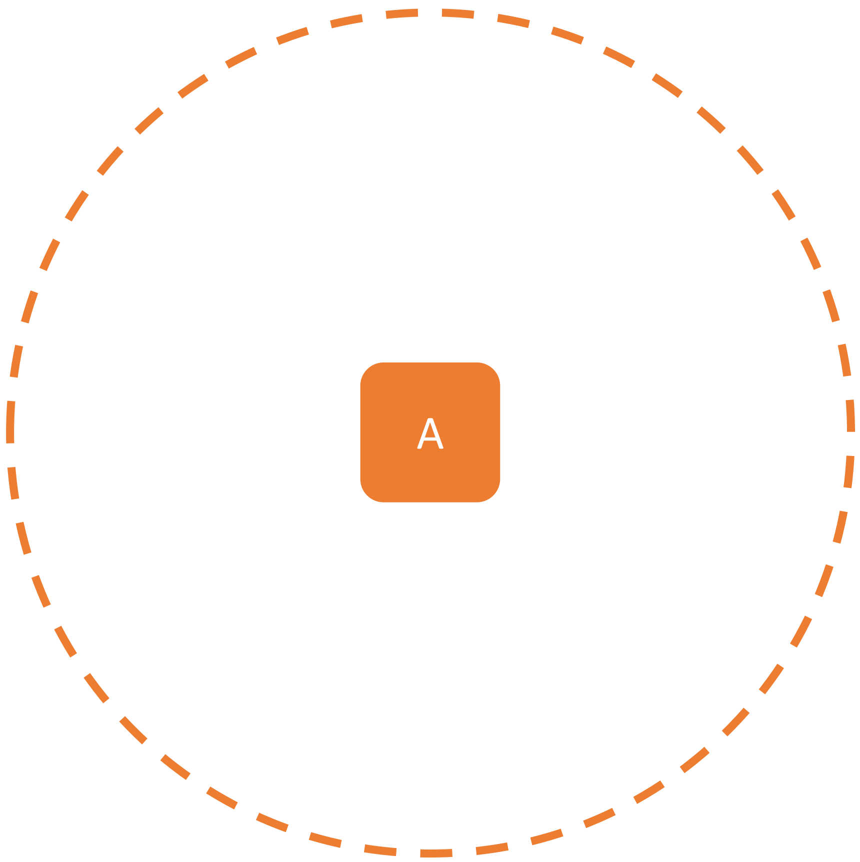



With Adaptive Duty Cycle

# Multiple Wireless Hops

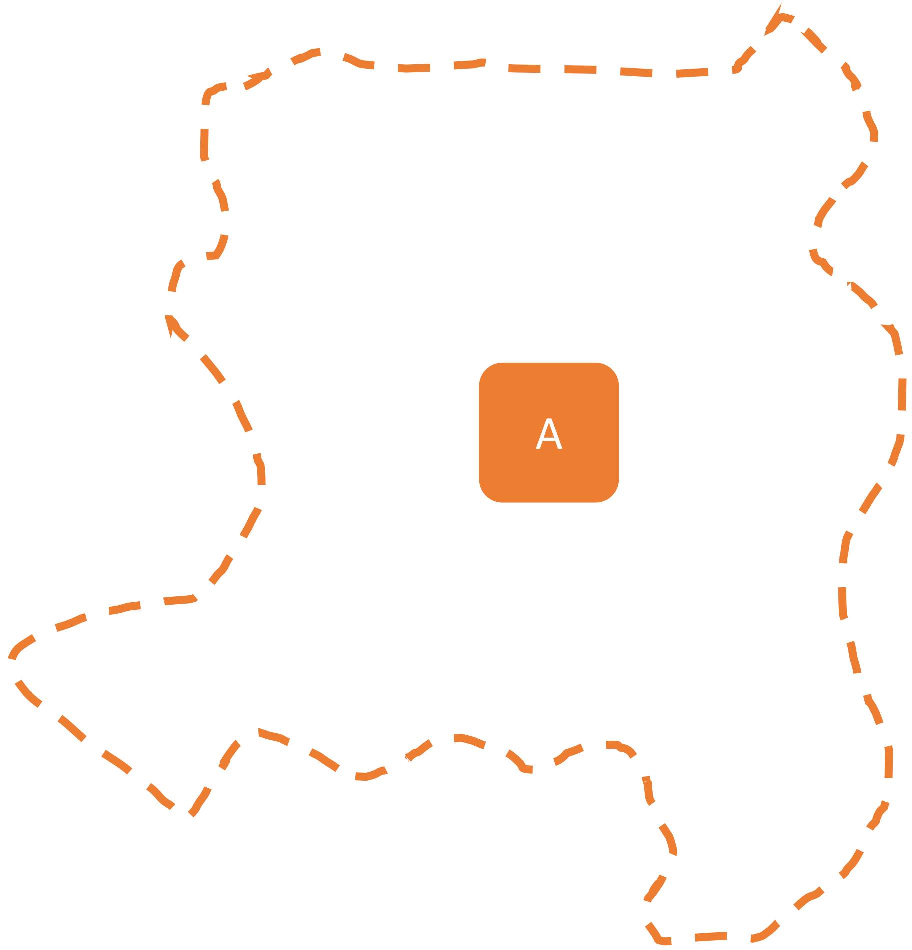


# Wireless Range of a Node



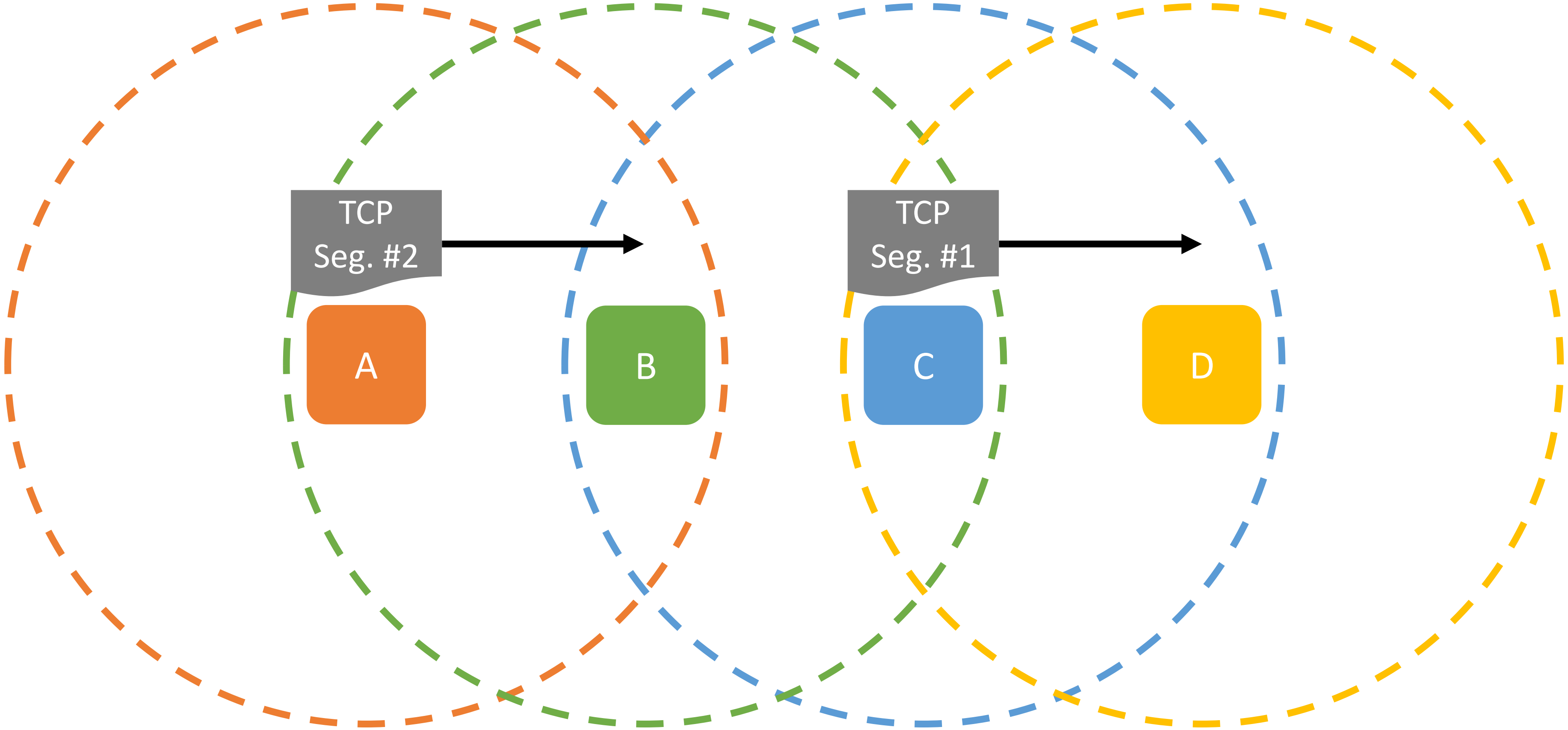

**(unit disc model)**

# Wireless Range of a Node



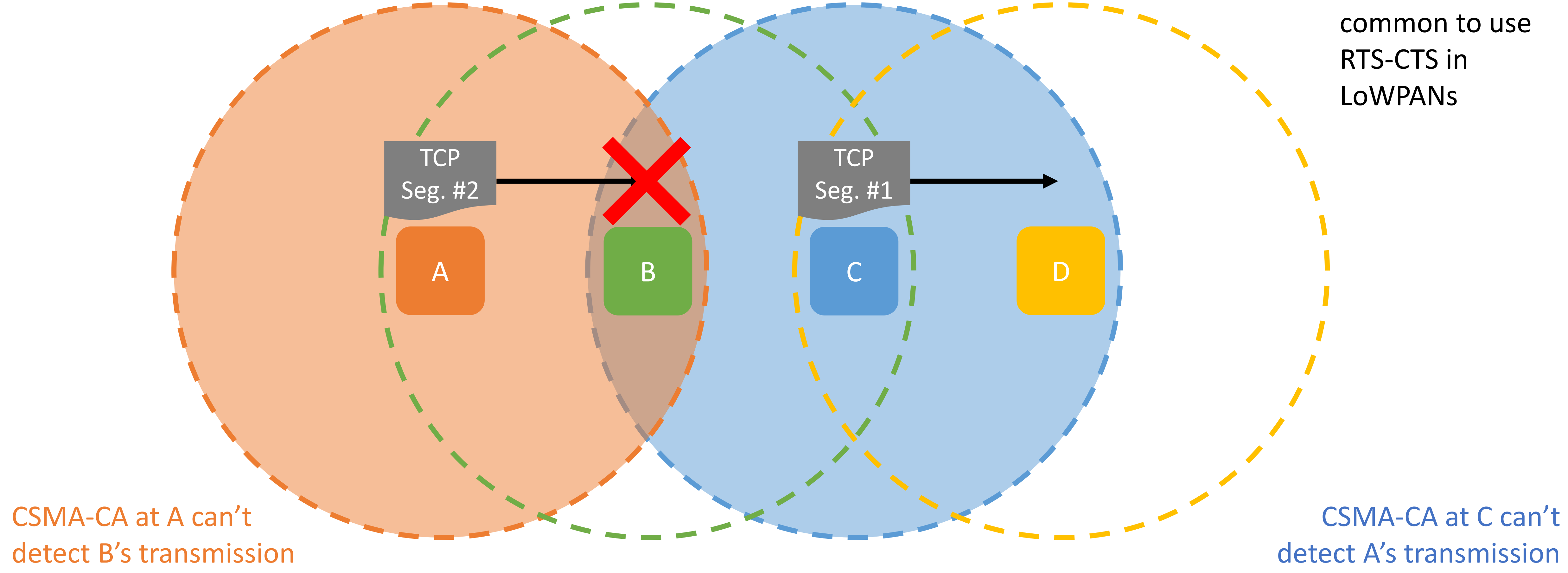

**(more realistic)**

# Hidden Terminal Problem in LoWPANs

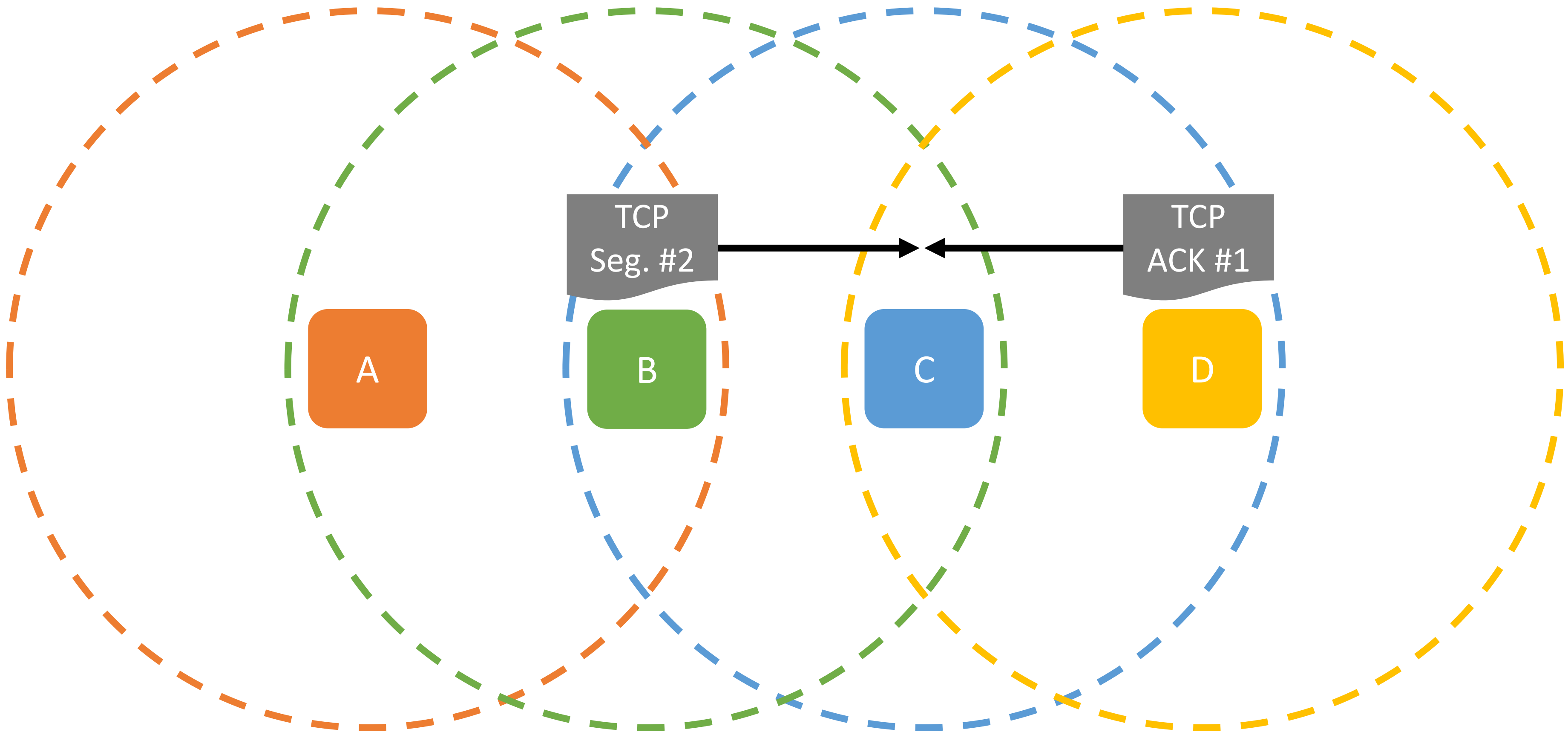


# Hidden Terminal Problem in LoWPANs

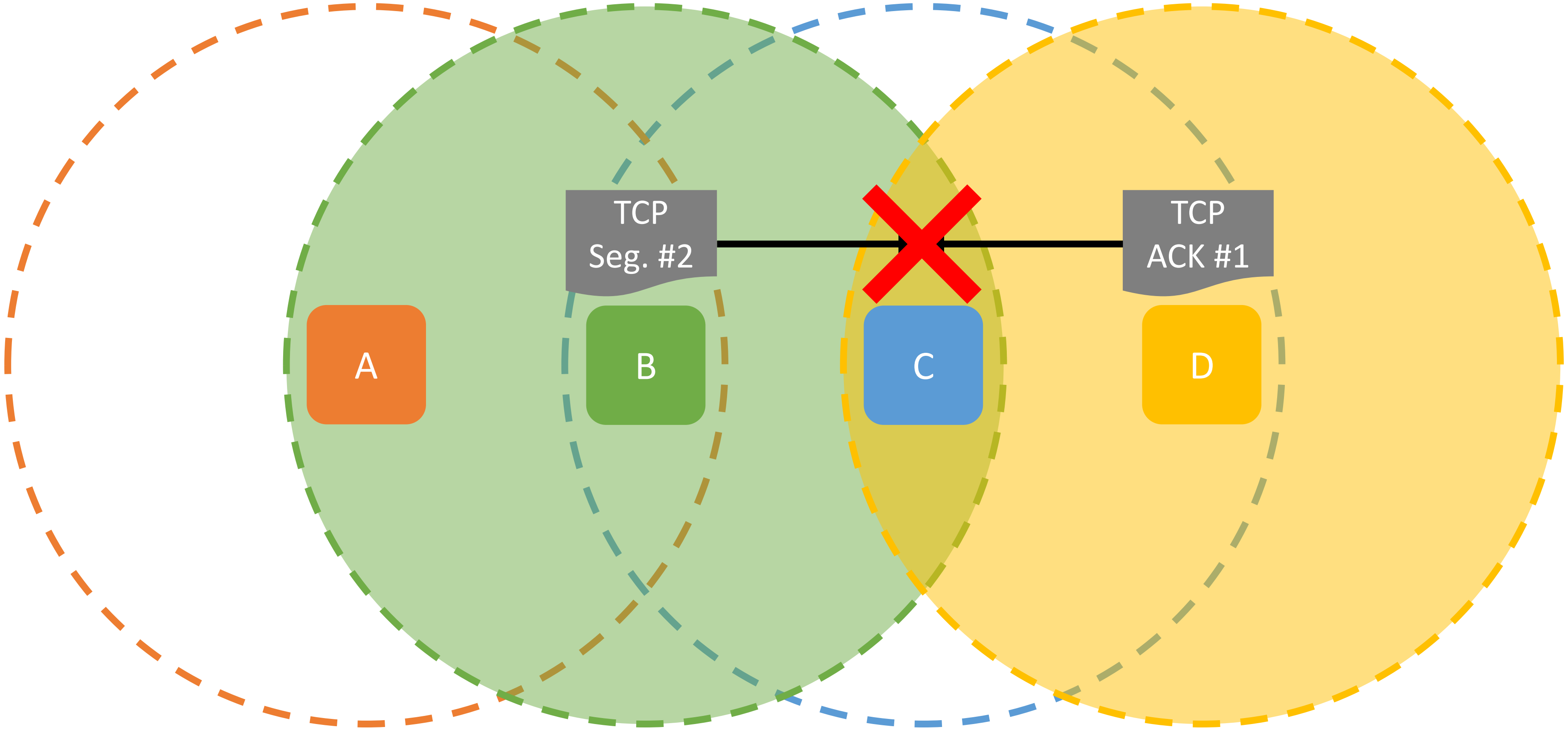
Due to the small MTU, it is not common to use RTS-CTS in LoWPANs



# Hidden Terminal Problem in LoWPANs



# Hidden Terminal Problem in LoWPANs

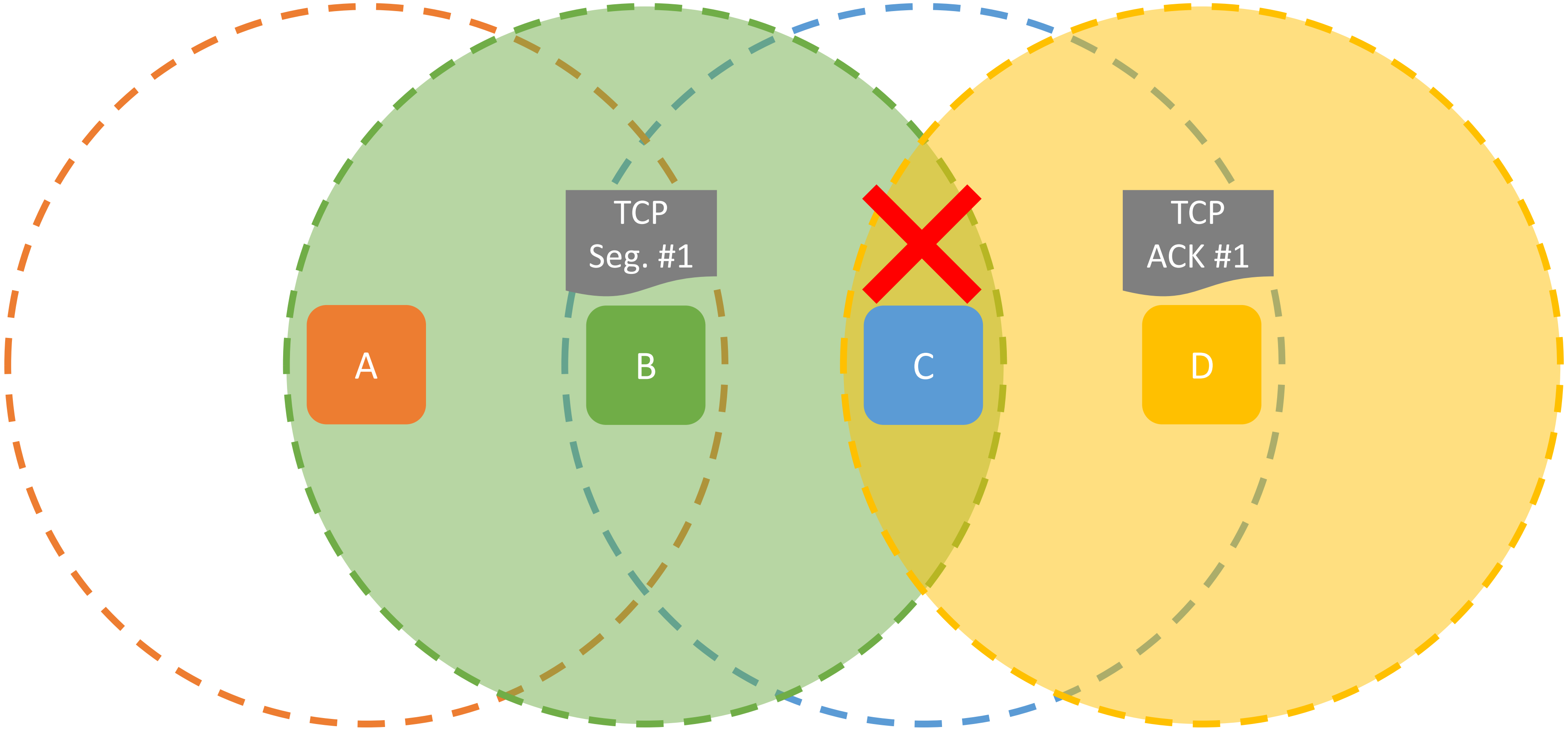


# Mitigating Hidden Terminals

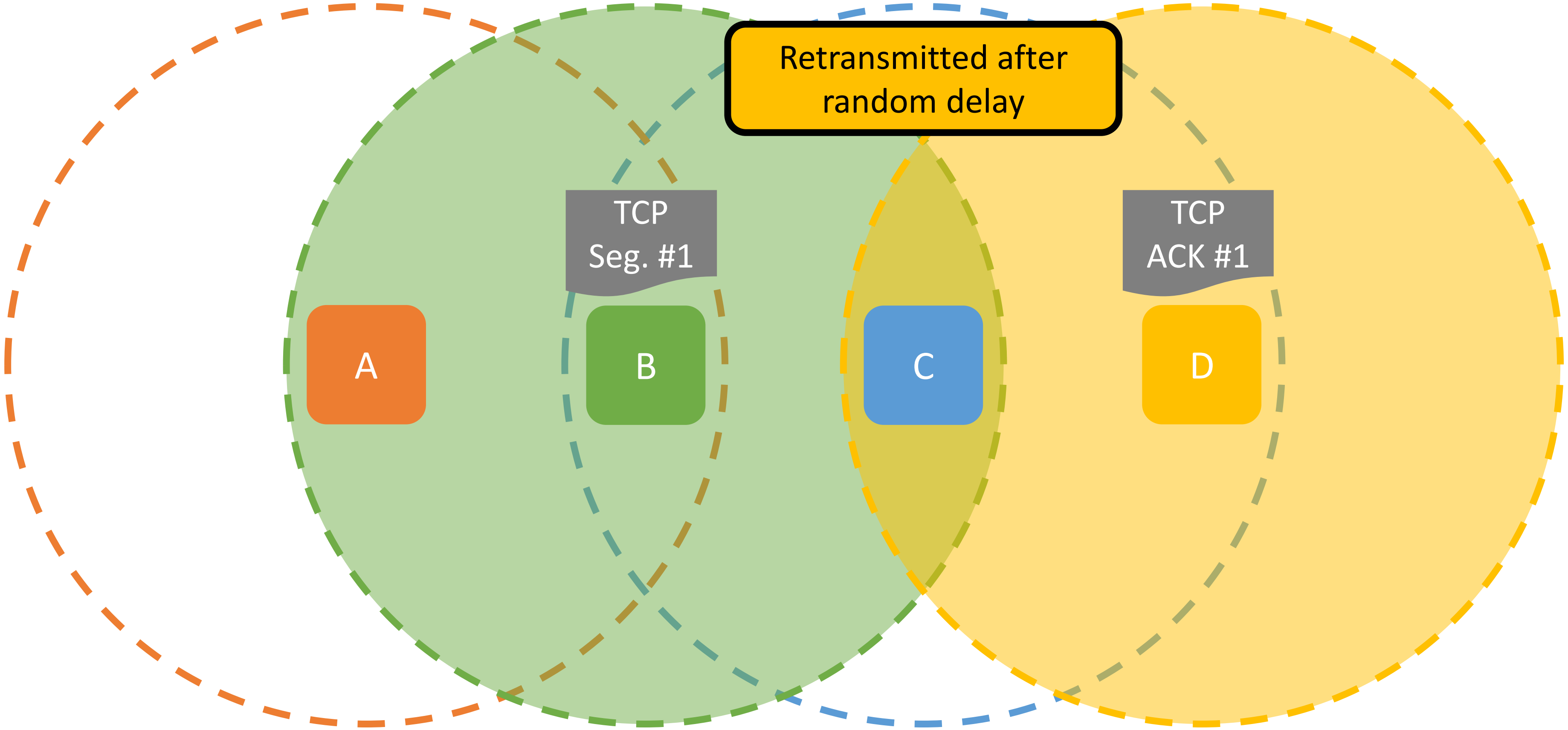
- If transmission fails (no link-layer ACK), wait a **random** amount before retrying

- This is different from CSMA
  - Longer delay (10× the time to transmit a frame)
  - Delay occurs if transmission fails, even if channel appears clear

# Mitigating Hidden Terminals

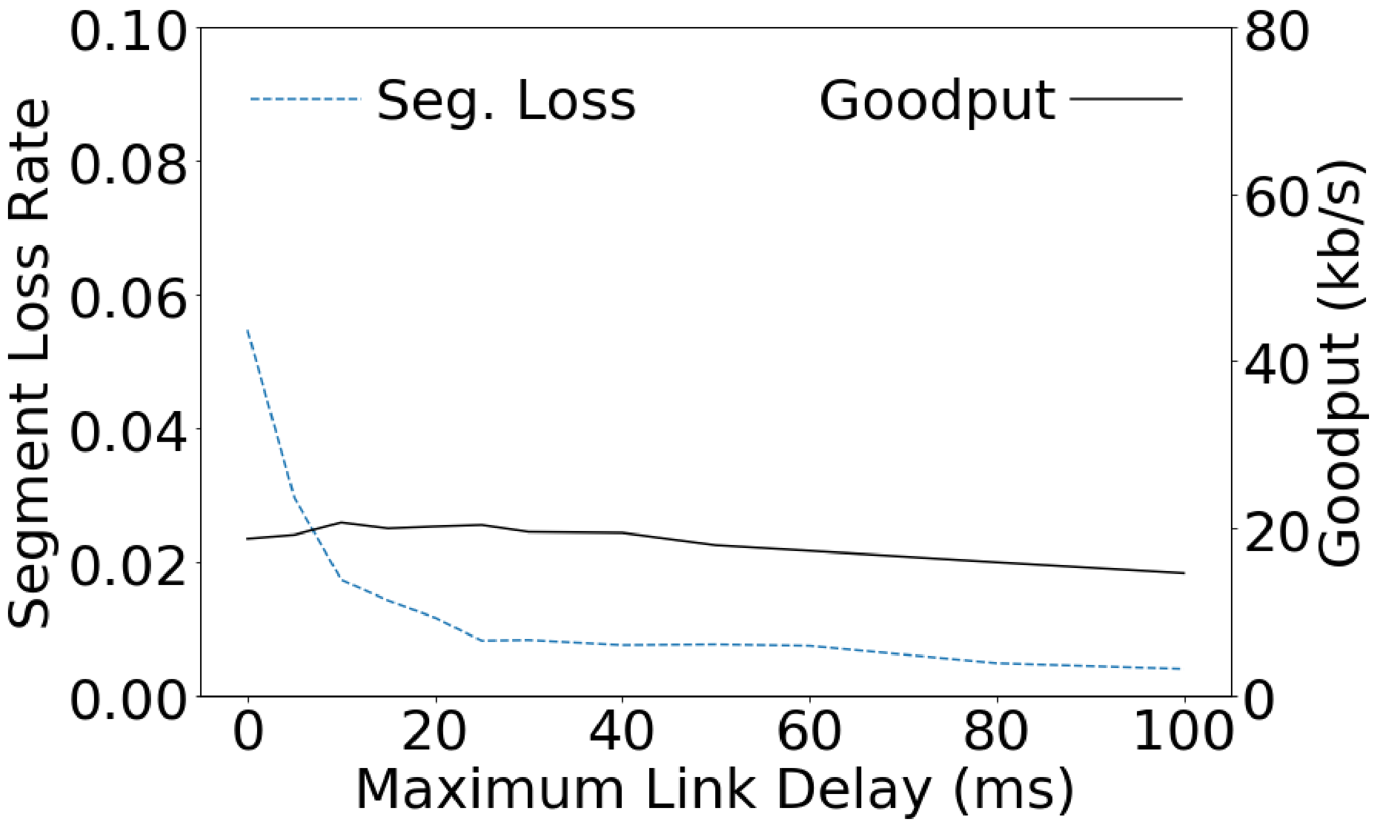


# Mitigating Hidden Terminals

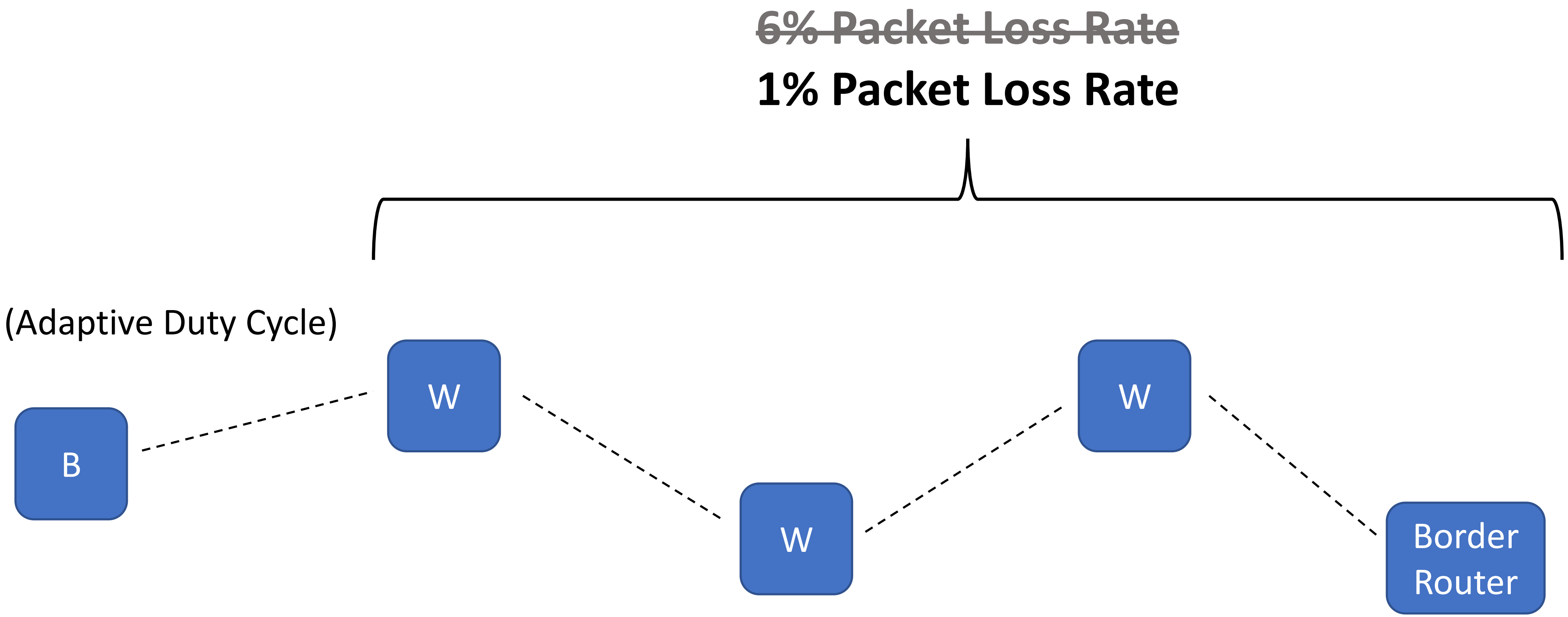


# Mitigating Hidden Terminals

- If transmission fails (no link-layer ACK), wait a **random** amount before retrying



# Multiple Wireless Hops



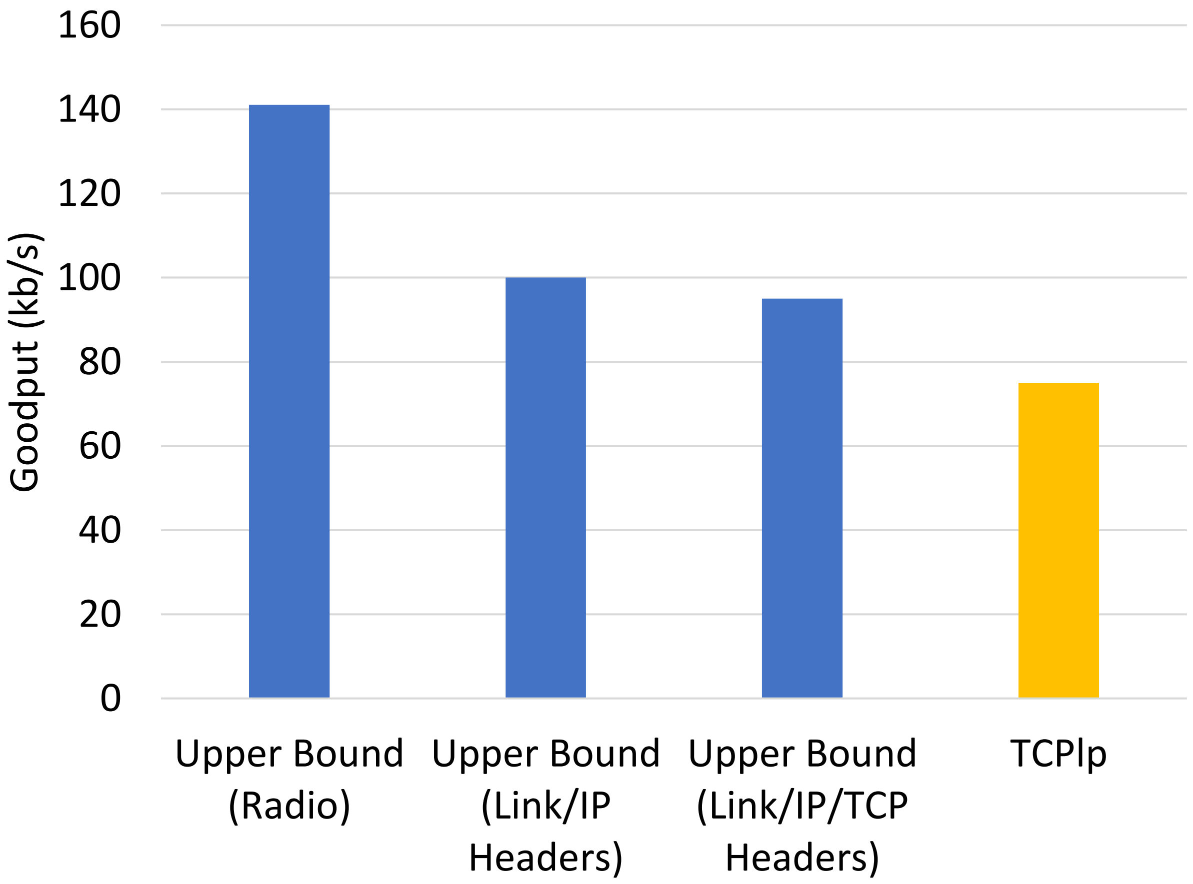
# Roadmap

1. Overview
2. Why the expected reasons for poor TCP performance don’t apply
3. Techniques to improve TCP performance in LoWPANs
4. **Evaluation and conclusions**

# TCP uses the Link Efficiently

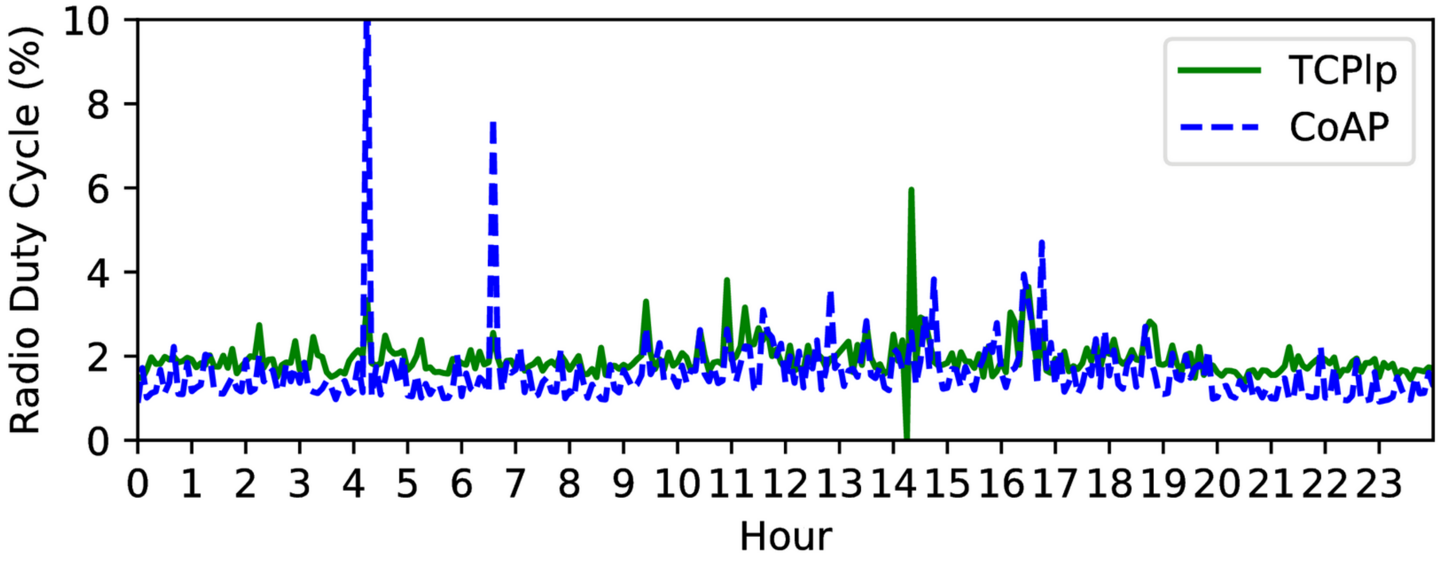
- 75 kb/s goodput over one hop
  - 5–40x more than prior studies

- Within 25% of a reasonable upper bound with headers



# TCP uses Energy Efficiently

- We used TCP and CoAP for a sense-and-send task, and measured radio duty cycle over a 24-hour period




**Both TCP and CoAP have a radio duty cycle of ≈2%**

# Now that TCP is a Viable Option…

1. We should reconsider the use of lightweight protocols that emulate part of TCP's functionality (e.g., CoAP)

2. TCP may influence the design of LoWPAN networked systems
   - Rethink gateway-based architectures
   - TCP allows for better *interoperability*

3. UDP-based protocols will still be used in LoWPANs
   - For applications where specialized protocols substantially outperform TCP

# Summary

1. We implement TCPlp, a full-scale TCP stack for LoWPAN devices
2. We explain why expected reasons for poor TCP performance don’t apply
3. We show how to address the actual reasons for poor TCP performance
4. We show that, once these issues are resolved, TCP performs comparably to LoWPAN-specialized protocols

# **Thank you!**

Code (reproducibility): https://github.com/ucbrise/tcplp
Code (deployment): https://github.com/openthread/openthread

Paper: https://arxiv.org/abs/1811.02721
(Published at Usenix NSDI 2020)

This material is based on work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1752814. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.