# Optimizing NAT and Firewall Keepalives using PCP

draft-reddy-pcp-optimize-keepalives-00

T.Reddy, M.Isomaki, D.Wing, P.Patil

# Purpose

- Many applications need to keep their NAT and FW mappings alive to stay reachable
  - NAT/FW mapping timers are short/unknown – resulting in high frequency of keep-alives
  - High keep-alive frequency leads to battery consumption
- PCP-base Section 10.3 explains how to use PCP for "Reducing NAT or Firewall keep-alive Messages"
  - But, some details are missing on when and how it can be used
  - **More guidance to app developers needed**

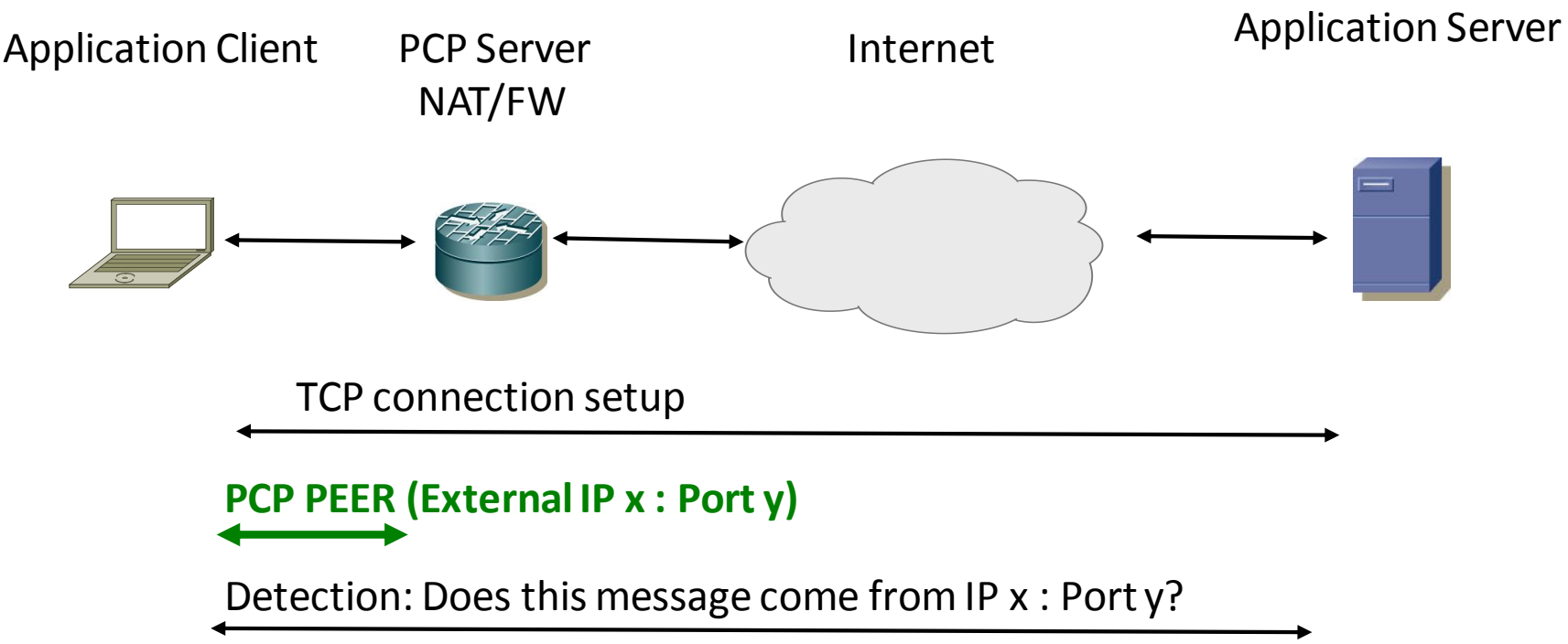
# Assumptions

- Applications often want to do ALSO end-to-end liveliness checks, but less frequently than typical NAT keep-alives
  - Example: 30 min. vs. 3 min. interval
- Applications need to detect when they can rely on PCP for reducing its keep-alive rate
  - No unexpected extra NATs of firewalls on the path
- Good old heuristics are still recommended even with PCP
  - Gradually reduce rate and be prepared to fall back to higher rate if it does not work…
  - Cache detection results…

# In the Draft

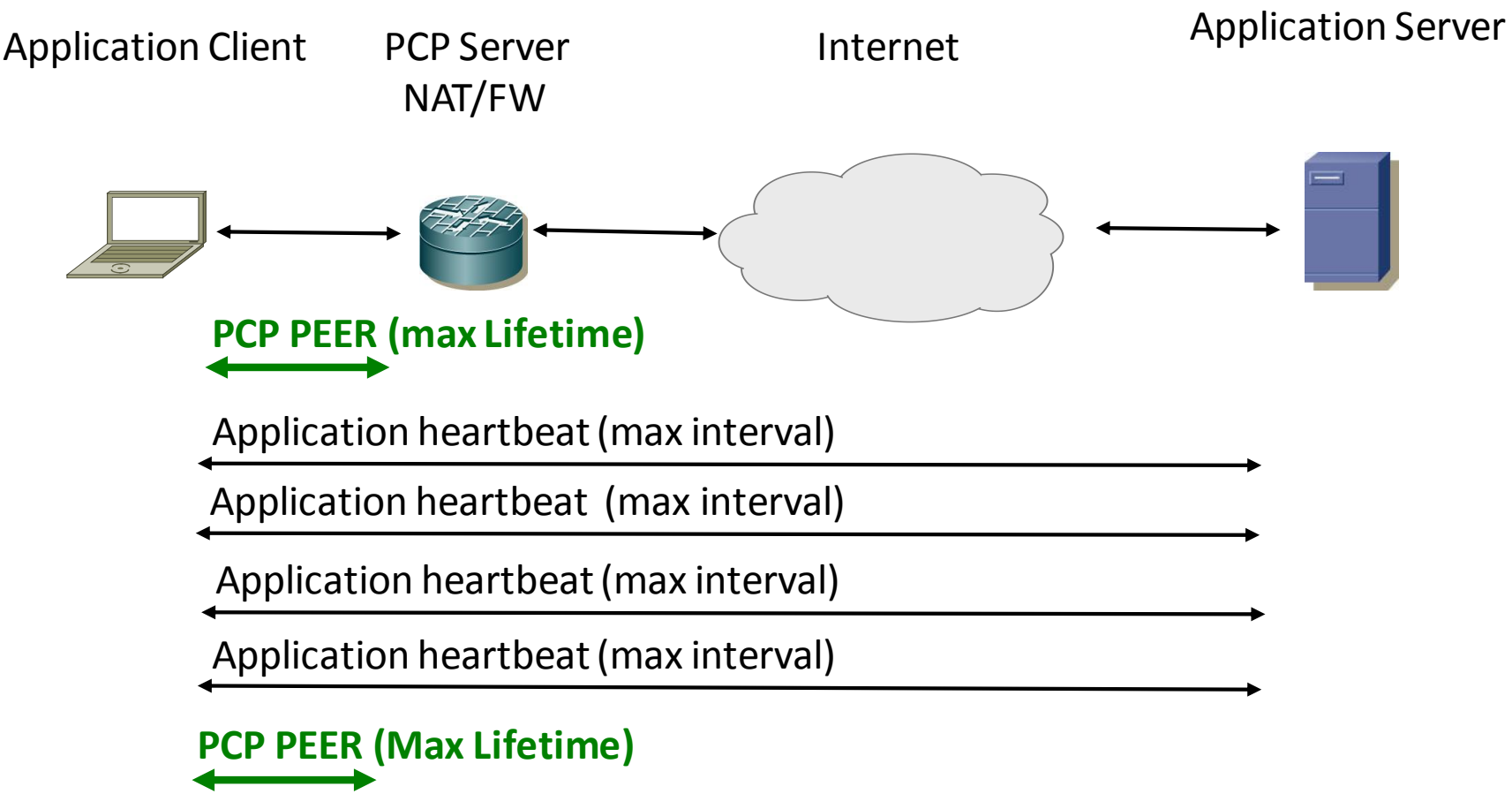
- Scenarios
  - Client-Server applications
  - Peer-to-Peer applications
- Detection
  - Unexpected NATs before or after PCP server
  - (Unexpected firewalls)
- (Keep-alive optimization)
- Operation with App protocols
  - SIP, HTTP, RTP, RTCWeb Data Channel
  - (XMPP, WebSocket, …)

# Detecting Unexpected NATs




- PCP itself can detect unexpected NATs between client and PCP server
- Application can detect unexpected NATs behind PCP server

# Keep-alive Optimization




- Synchronize PCP and application messages to save power

# To Do & Open Issues

- Better explanation of the actual optimization and when it applies
- Best detection strategies
- Firewall detection
- Working with HTTP proxies
- Querying or setting the “no traffic” mapping timeout with PCP?