RESTful APIs and 802.1Qcc

Rodney Cummings
National Instruments

Agenda

- What is a RESTful API?
- How can RESTful apply to 802.1Qcc?
- What is CoAP?

Premise of this Presentation

- Configuration of TSN requires protocol(s)
- Past TSN work has focused on invention
  - Create a new protocol from scratch
  - Publish in 802.1 or elsewhere, but that's just the 1st step
  - Momentum requires open source, tools, OS integration, etc
    - Challenging to find companies who are willing to invest
    - MRP and IS-IS struggle to this day
- What if we could find an existing protocol?
  - millions of active users… huge software development community… plenty of open source… ships with every operating system…
  - Maybe The Internet can help

What is HTTP?

- Foundational protocol of The Internet (web)
- Request/response (client/server) using TCP
  - HTTPS: HTTP Secure using TLS (SSL)
- Very simple, with two fundamental concepts
  - Resource: Identified with URI
  - Methods: Request is stateless
    - GET: get resource's data
    - POST: create a new subordinate of resource (e.g. Twitter post)
    - PUT: replace the resource's data (create if doesn't exist)
    - DELETE: delete resource

What is REST? What is RESTful API?

- REST: Architectural style for designing an API
  - Stateless: Each method executes on its own (like a browser)
  - Client/server, layered: No reliance on intermediaries
  - Most of this style is built into HTTP itself
- RESTful API: Use HTTP for an API (instead of website)
  - Other protocols are possible (e.g. CoAP), but HTTP assumed
  - Data (media type) is typically JSON
  - No formal standard, but development tool support is **huge**
    - E.g. PRMD takes a JSON schema and creates a full API user manual

Benefits of RESTful API (1 of 3)

- Creating a standard is easy
  - Many design guides and tools available
  - Step 1: Create JSON schema
    - Such as from YANG
  - Step 2: Specify rules for URI and HTTP
    - ~20 to 40 pages
    - Mostly copy & paste from other APIs
  - That's it… done!
  - Most RESTful APIs document on the web
    - E.g. GitHub, Twitter, Stripe, Facebook, …

Benefits of RESTful API (2 of 3)

- Creating a client is easy
  - Get started using simple command line (cURL)
  - Built into most programming languages
    - E.g. Stripe documentation has examples for Go (OMG!)
- Creating a server is easy
  - If your product runs a web server, you are > 90% done
    - Many industrial devices already run web servers
  - Most software teams are already familiar with HTTP tools

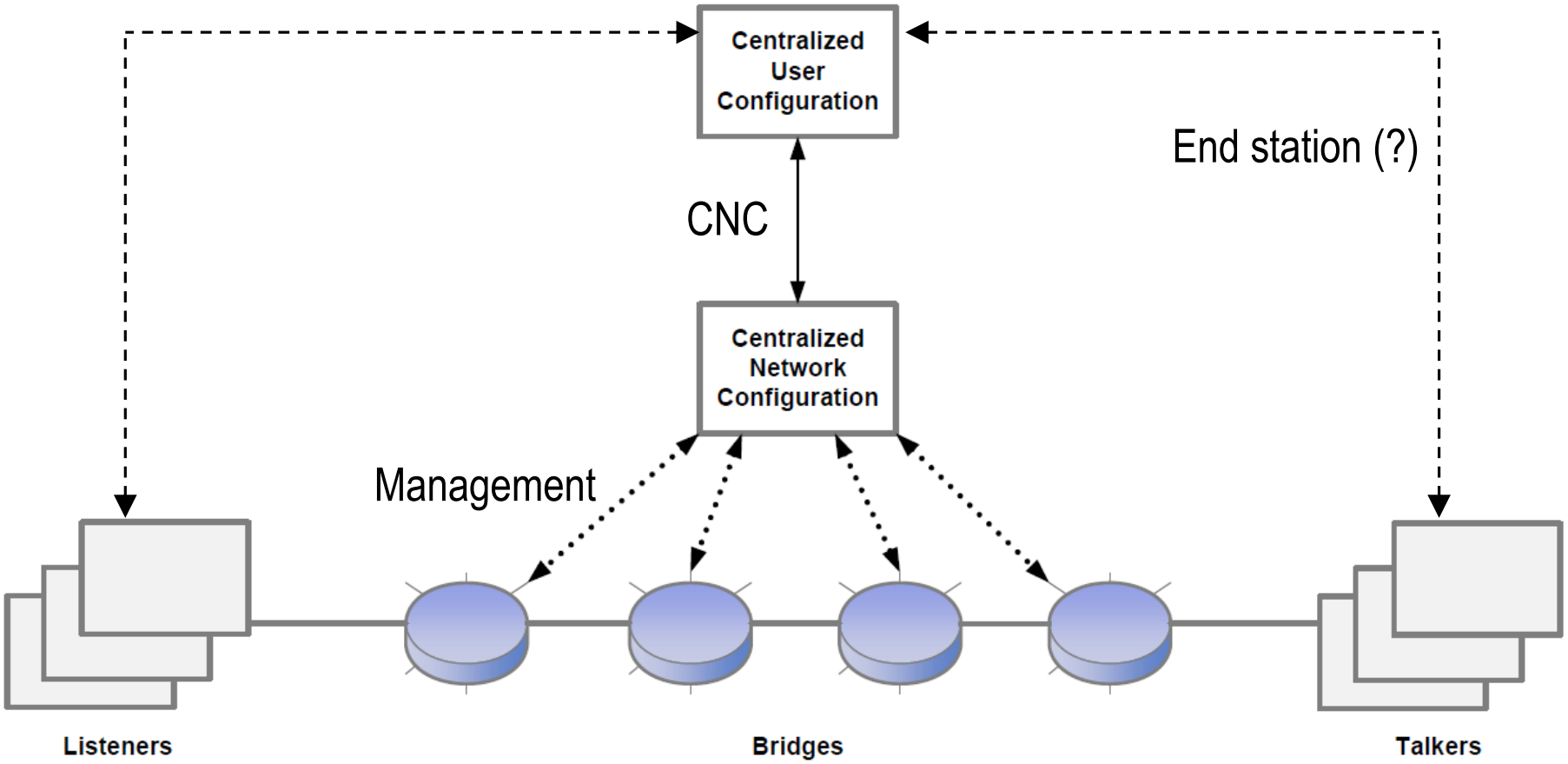
Benefits of RESTful API (3 of 3)

- Longevity
  - HTTP is not going away anytime soon
- Security
  - Based on TLS (HTTPS), and kept up to date
- Transport
  - TCP provides reliable delivery of large data
    - Scalable
  - Bridges and routers just forward to destination
- Server implicitly supports multiple simultaneous clients

How can RESTful apply to 802.1Qcc?

Qcc Fully Centralized

- Use as frame of reference



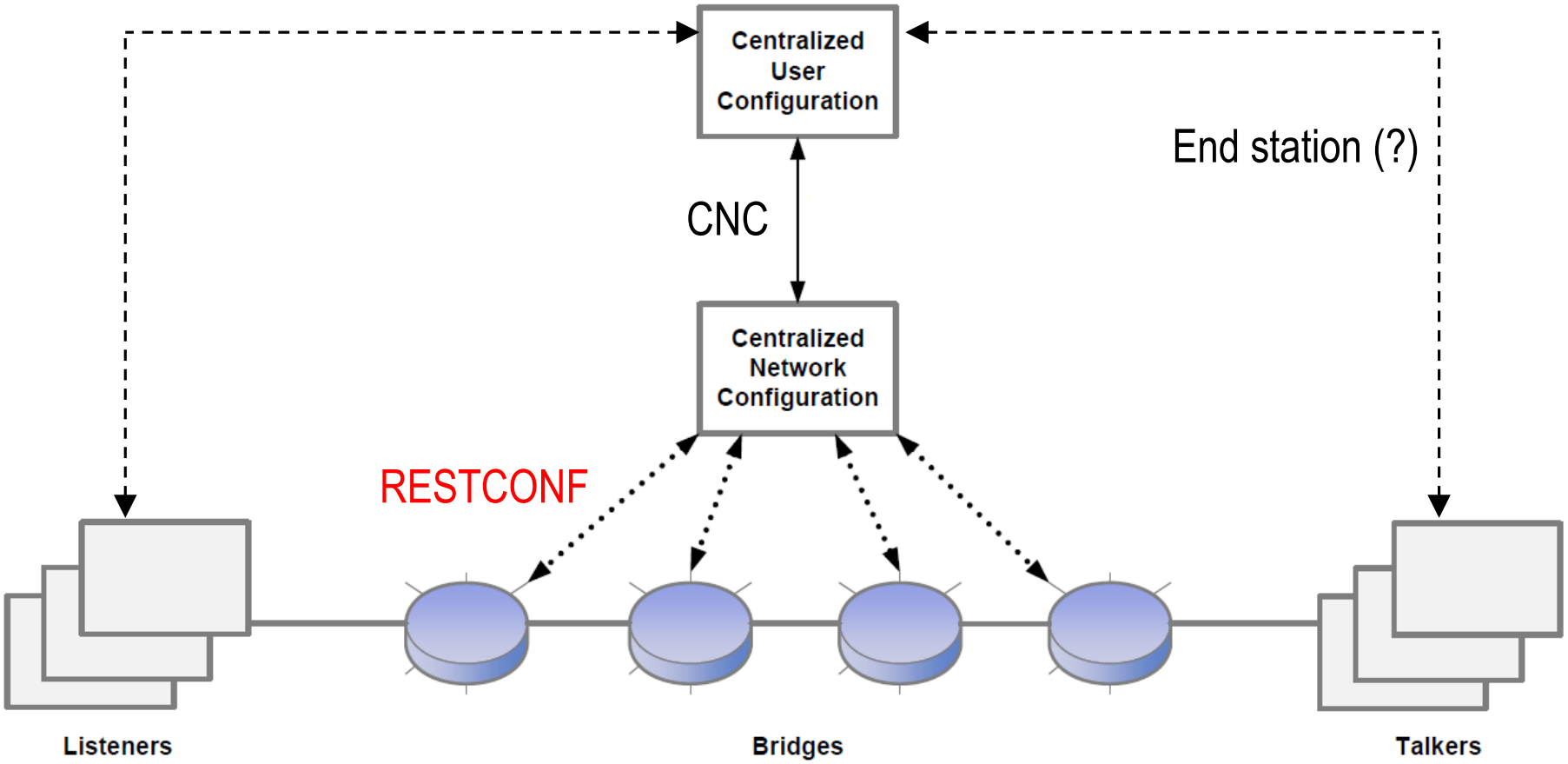
RESTCONF for Management

- RESTCONF is a RESTful API for managing YANG data
  - Client is CNC (aka NMS)
  - Server is a bridge or router (network infrastructure)
- HTTPS GET/PUT used to read/write managed objects
- Server supports JSON (typical), XML, or both

- Draft in IETF NETCONF working group
  - WG state = Submitted to IESG for Publication (as RFC)
- YANG modules in work for 802.1Q and 1588
  - 802.1AS YANG can be done as augment of 1588 YANG

RESTCONF for Management

- (red shows usage of RESTful APIs)

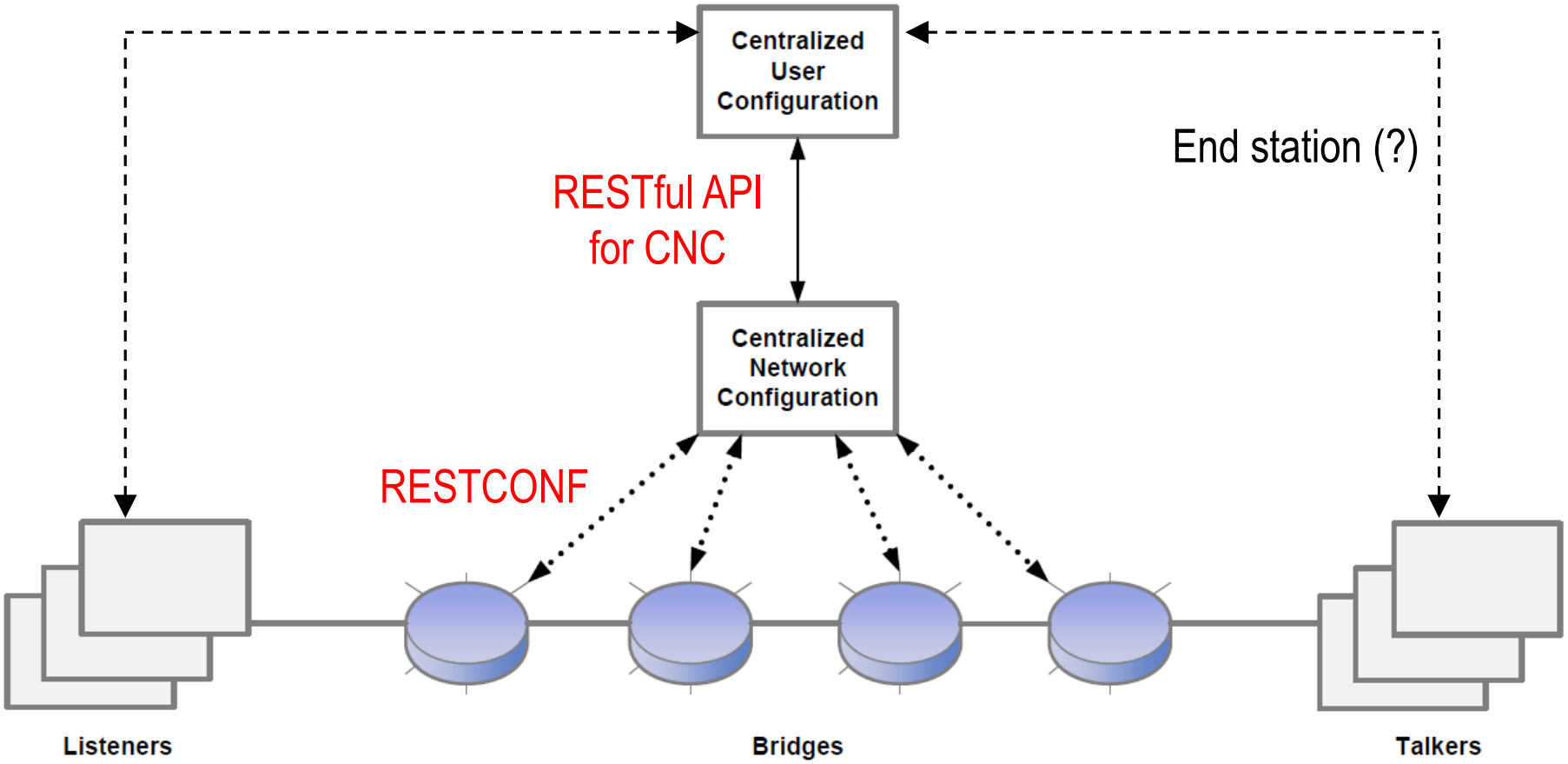


RESTful API for CNC

- Remember those steps for creating a standard?
  - Step 1: Create JSON schema
    - 1-1 translation from YANG specified in Qcc UNI (99.2)
  - Step 2: Specify rules for URI and HTTP
    - E.g. Syntax for Talker, Listener, and StreamStatus in URI
  - That's it… done!
    - Scalable, so 1000's of streams
- API can build on top of Qcc's data model as needed
  - E.g. Time-sync UNI proposal
- CNC supports multiple clients (CUCs)

RESTful API for CUC ↔ CNC

- (red shows usage of RESTful APIs)



What is the CUC?

- In most time-sensitive applications a **human** uses a software entity (tool) to:
  - Discover: End stations w/ resources & capabilities
  - Design: What goes where in the distributed application
  - Program: Write/debug software components for application
  - Connect flows: Input to output, Code to I/O, code to code, …
  - Control: Start/stop state machines
- Plug&play (i.e. no human) is a 100% software problem
- What do these tasks have in common?
  - Largely unrelated to the network
    - Requirements for code and I/O are more complex than network

CUC is the Application (User) Tool

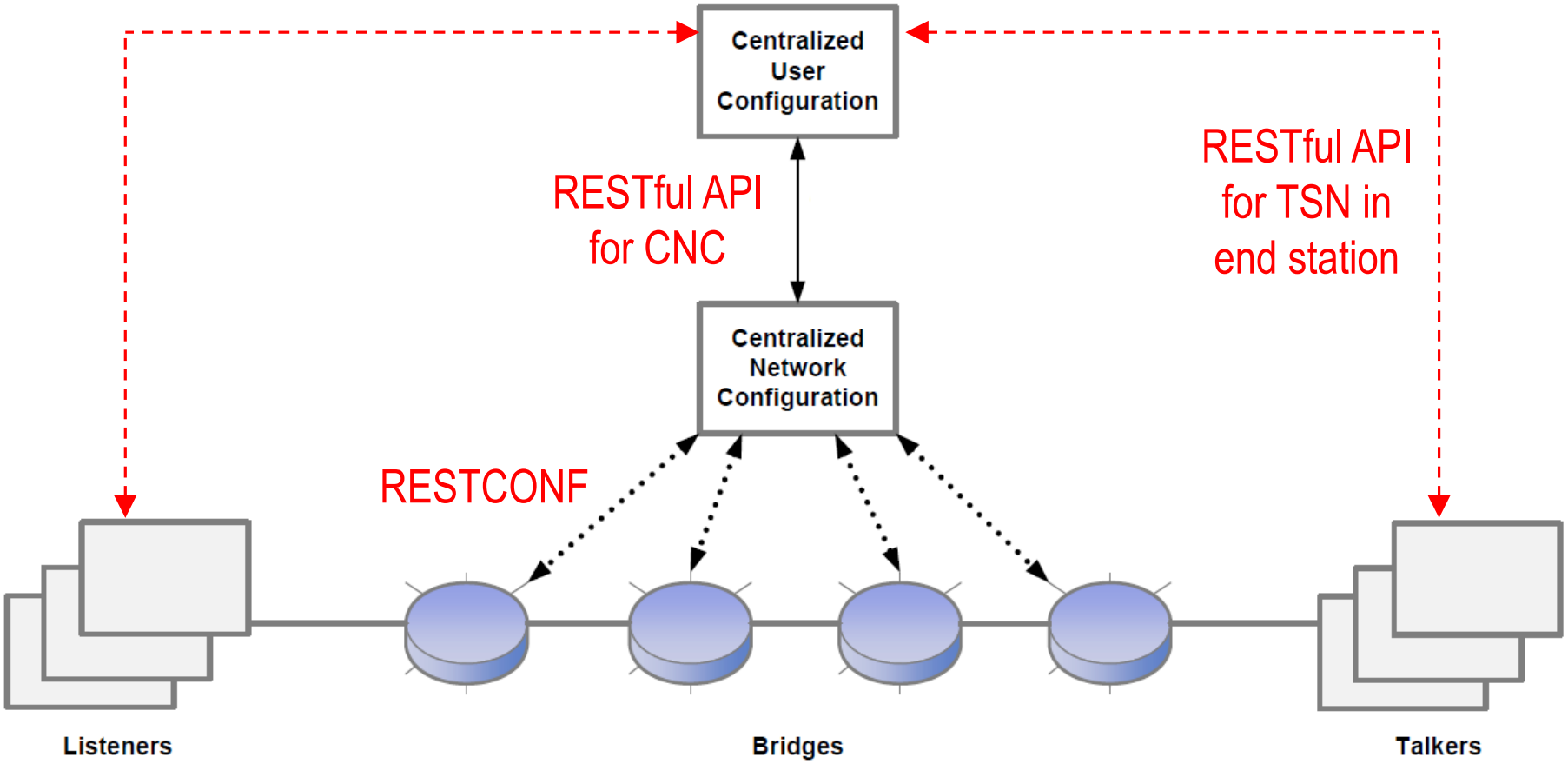
- If/when flows occur over network, TSN is relevant
- From CNC's perspective, CUC makes most decisions
  - CUC knows talker/listeners, MaxLatency requirements, etc
  - Talker/listener end stations can be 'dumb'
- Today's CUCs have their own protocols to end stations
  - Qcc is generally not relevant to those protocols
    - Exception: TrafficSpec, InterfaceCapabilities/Configuration
- Two approaches to integrating TSN into CUC protocol
  1. Create CUC protocol v2 to intimately integrate TSN
  2. Leave CUC protocol as-is; Configure TSN separately

RESTful API for TSN in End Station

- RESTful is great for option 2: Configure TSN separately
  - RESTful API is not a new CUC protocol
  - Opposite goal: Add TSN with no change to CUC protocol
- Client is the CUC
- Server is talker/listener end station
- Goal is to setup TSN for streams that CUC is connecting
  - MaxLatency and other TSN requirements decided by CUC
    - Typically driven by physical input to output time

RESTful APIs as Complete TSN Solution

- (red shows usage of RESTful APIs)



What is CoAP?

CoAP

- "I have a constrained product that cannot run HTTPS. What do I do?"
  - Where "constrained" means small CPU / memory / power
- IETF CoRE working group: Constrained RESTful
  - CoAP (RFC 7252): binary HTTP equivalent
  - CBOR (RFC 7049): compact binary JSON equivalent
    - Including YANG mapping (draft)
  - CoMI (draft): compact RESTCONF equivalent
- Used today for low power wireless (e.g. 6TiSCH)
  - Mapping for IPv6 UDP DTLS; Open source available
- Clear option for TSN

Thank you