

# Live Streaming End-to-End with Packet Recovery and libRIST Development Roadmap

Streaming live using RIST On Demand to thousands,  
how you can have your cake and eat it too.

---

**Sergio Ammirata, Ph.D.**

# What We'll Discuss


## libRIST Development "Roadmap"

- Clarifying misconceptions
- Broadening "Reach"
- Performance Enhancements and Bug Fixes
- Boast: We Think We've Achieved the Original "Interoperability" Goal!

## Live Streaming End-to-End with Packet Recovery

- What We Mean by *End-to-End*
- For *Really* Large Audiences
- Highly Secure, Authorization, Easy Management, Custom Client
- Packet Recovery *with* Low Latency



The image features a dark blue background with abstract, wavy, light blue patterns that resemble sound waves or data lines. A solid blue vertical bar is positioned on the right side. The text is centered and reads: 

**libRIST Development Roadmap**  
**2020-2023**  
**Goals for 2024**

# libRIST Development "RIST Protocol Specs"

## RIST Milestones



RIST Activity  
Group formed  
by Video  
Services  
Forum  
April 2017

VSF TR-06-1  
RIST Simple  
Profile  
published  
October 2018

Advanced  
Profile  
Development  
Completed  
September  
2021

Successful  
multi-vendor  
interop  
demonstration  
September  
2018

VSF TR-06-2  
RIST Main  
Profile  
published  
March 2020



# libRIST Development “RIST Protocol Specs”

- Simple Profile:
  - TR-06-1: First Release (Sep 2018)
  - TR-06-1:2020 RTT Echo message (Jun 2020)
- Main Profile
  - TR-06-2: First Release (Mar 2020)
  - TR-06-2:2021 PSK Security Fix update (Apr 2021)
  - TR-06-2:2022 EtherType and EAP-SHA256-SRP6a (Aug 2022)
  - TR-06-2:2023 Errata to 2022 version (Aug 2023)
- Advanced Profile
  - TR-06-3: First Release (Oct 2021)
  - TR-06-3:2022 EAP-SHA256-SRP6a (Sep 2022)

# libRIST Development “RIST Protocol Specs”

The TR-06-4 series of recommendations define ancillary features for the RIST protocol that are applicable to multiple profiles.

- TR-06-4 Part1: Source Adaptation (Nov 2022)
- TR-06-4 Part2: RIST over Wireguard VPN (Jan 2023)
- TR-06-4 Part3: Relay (Jun 2023)
- TR-06-4 Part4: Decoder Synchronization (Jan 2024)
- TR-06-4 Part5: Multicast Discovery (Oct 2023)

Other recommendations are still in the works ...

# libRIST Development “Roadmap 2020-2023”

We assume, for this talk, you are familiar with RIST

- A new protocol for transmission of IP data across lossy networks using UDP/RTP with NACK-based retransmissions.

## Clarifying Misconceptions

- It is NOT limited to MPEGTS video streams, advanced profile includes support for any payload with clearly identifiable payload types, even raw binary payloads.
- It is NOT only for delayed streams that can afford high latency. It includes support for real time unprotected data channels and very low latency recovery channels.
- It is NOT for transmissions in only one direction. It allows bi-directional communication with and without packet recovery.

# libRIST Development “Roadmap 2020-2023”

## RIST Specification Improvements That Extend Reach

- Highly Secure Authorization for Mass Audience Live Streaming
- New “use scenarios” such as a one-way satellite-friendly “intrusive” protocol

## Distribution

- Liberal licensing for linkage in your own projects, BSD2
- Compile options in popular encoder, decoder platforms

Timely performance enhancements and bug fixes





# Roadmap 2020-2023: RIST Specification

## EAP SRP 6a Authentication

- Introduced 2022
- Free and more secure than most commercial DRMs, supports large audiences
- We'll Focus on this in part 2 of this presentation

## One-Way Network/Satellite

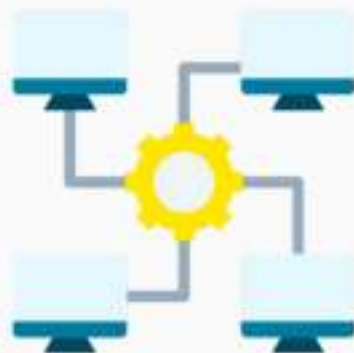
- “Intrusive” method adds error correction for one-way networking and/or highly asymmetrical networks



# Roadmap 2020-2023: Distribution

## Distribution

- Now available in OpenBSD and Debian among other distros and also integrated as compile options for ffmpeg, vlc, OBS and many others.
- Liberal BSD2 licensing provisions encourage incorporation into other software and hardware lines, such as my own “day job”



# Roadmap 2020-2023: Performance Enhancements

- Automatic configuration and adjustment to network conditions
- Logging and metrics for improved control/fine tuning
- Latest minor version was released this past Halloween



# Roadmap: Goals for 2024

- Add support for DTLS encryption and authentication
- Add support for the new Advanced Profile Specification
- Backport support into VLC 3.0 (patches in review)



# libRist “Roadmap:” the Original “Interoperability” Goal

Bearing in mind that this was the *original* RIST goal, we think that we’ve now got enough vendors, plus with the libRIST FOSS implementation, we believe that interoperability among packet recovery applications is now a reality.

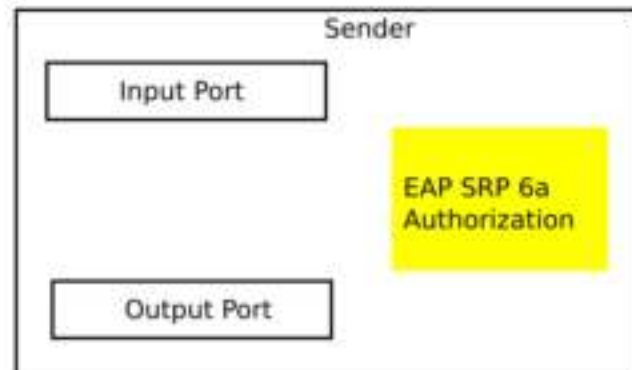
Additionally, with the liberal licensing of libRIST, there’s no reason that any foss project should think they can’t easily and quickly link in a library to support RIST flavor packet recovery into any encoder/decoder branded product.



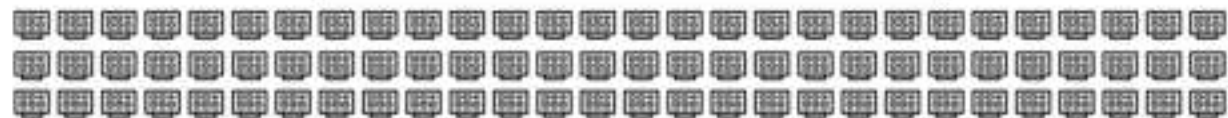


# Live Streaming End-to-End with Packet Recovery

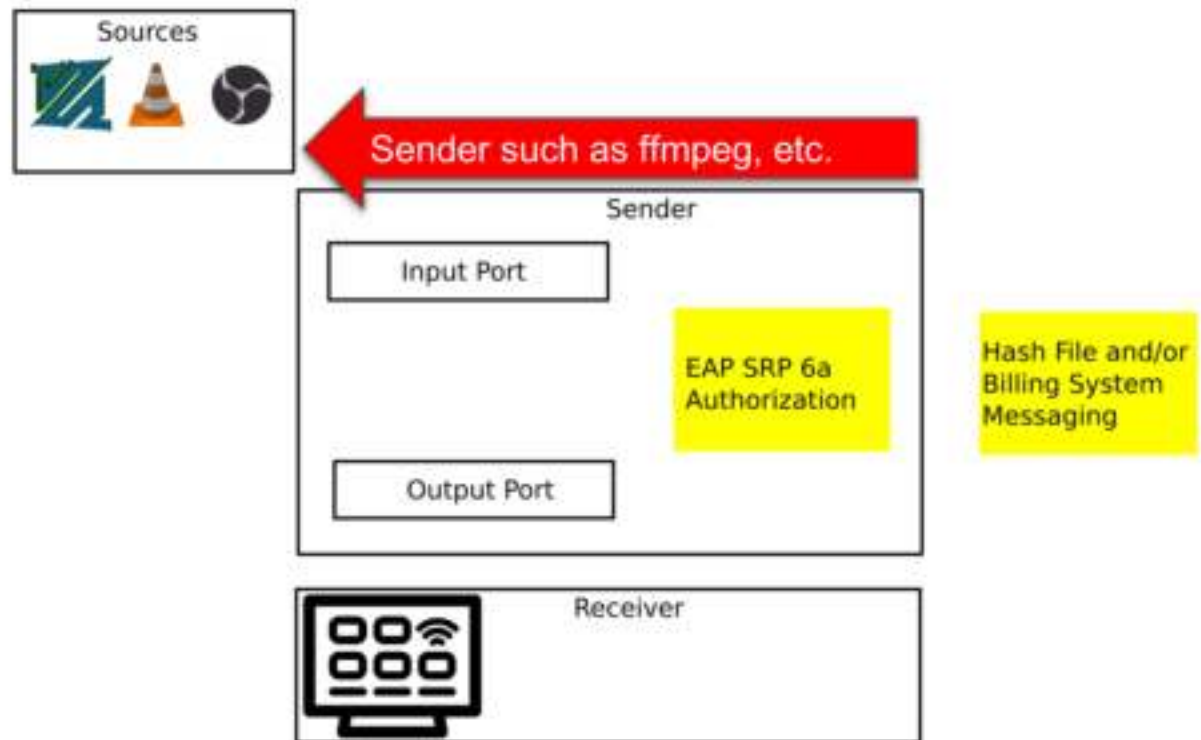
# Live Streaming: Architectural Overview



Hash File and/or  
Billing System  
Messaging

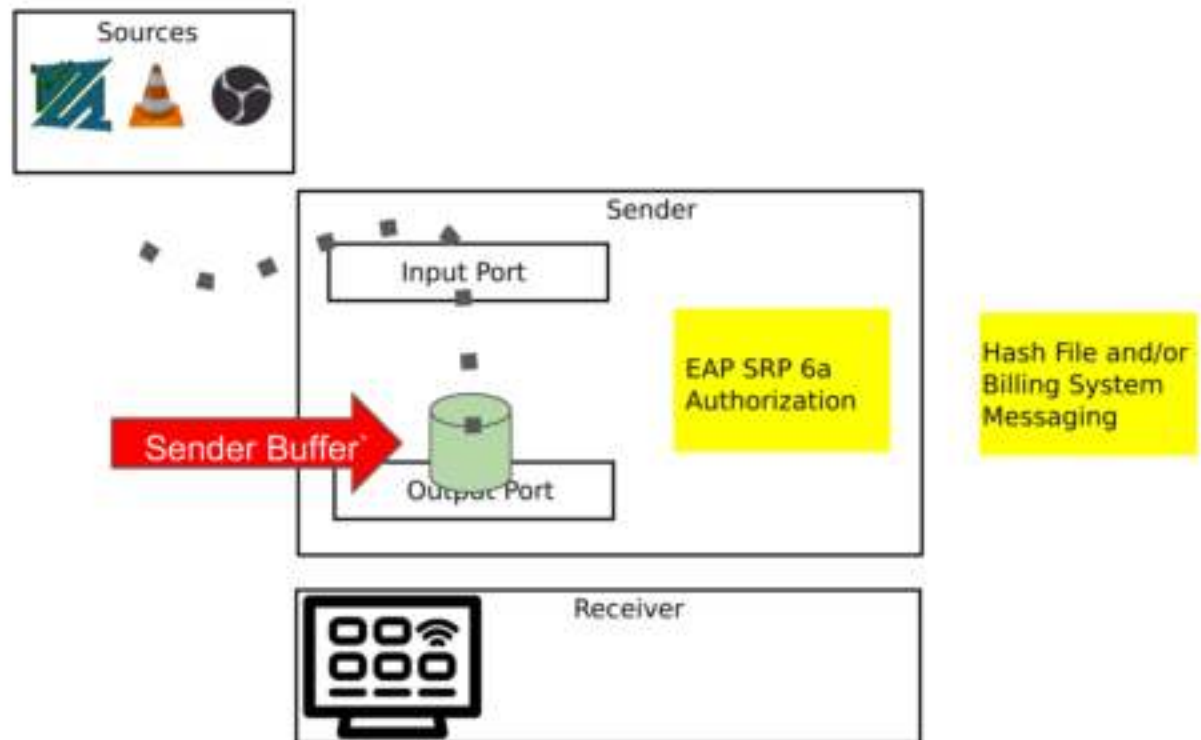


# Live Streaming: Architectural Overview

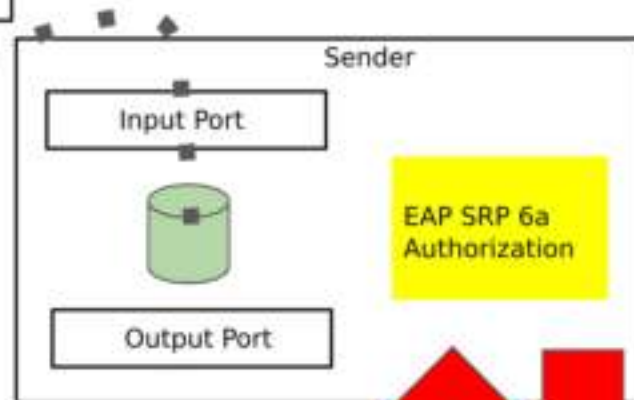




# Live Streaming: Architectural Overview



# Live Streaming: Architectural Overview

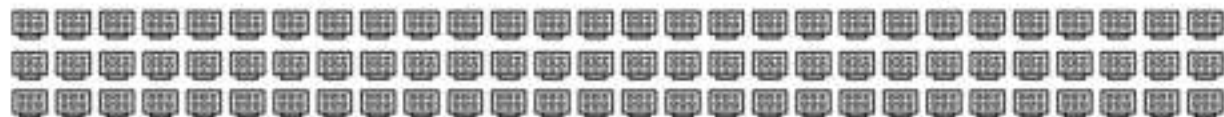


EAP SRP 6a  
Authorization

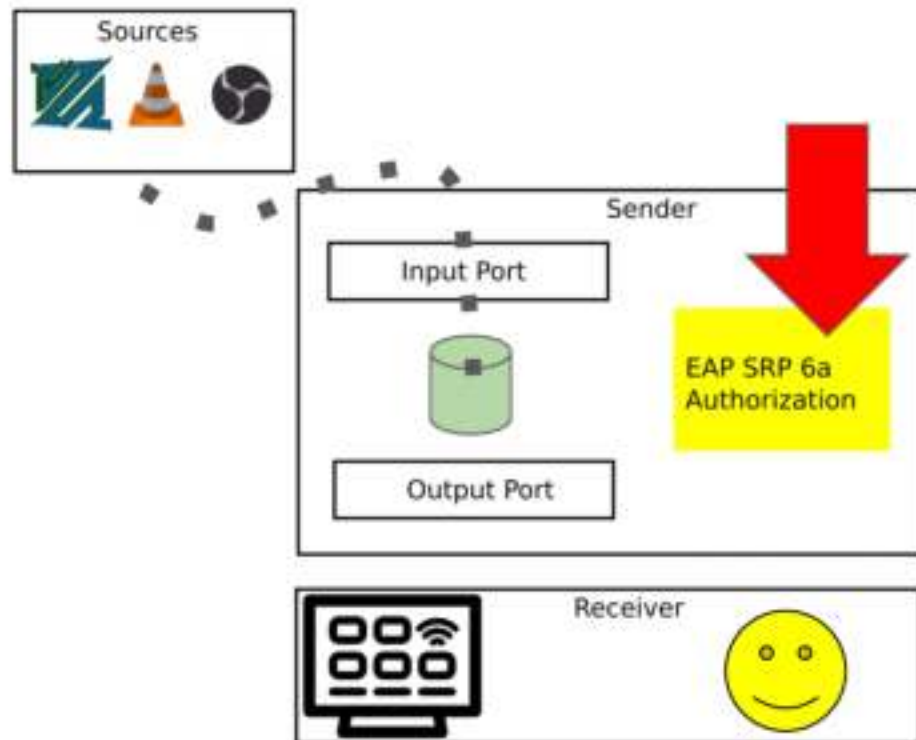
Hash File and/or  
Billing System  
Messaging



Handshake: Exchange of messages containing a username and passphrase hash.



# Live Streaming: Architectural Overview

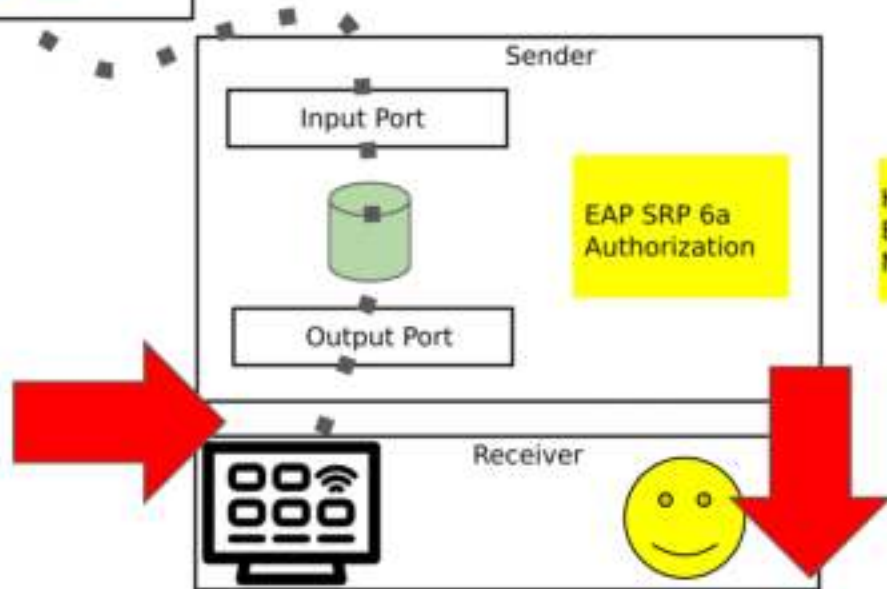


If username found, the next messaging provides the passphrase hash. This is checked against the passphrase store.

Hash File and/or  
Billing System  
Messaging



# Live Streaming: Architectural Overview



We can now switch to PSK, and send a key for decrypting the stream... and then start sending the stream itself.



# Live Streaming: Latency

- With planning, a 300ms glass-to-glass latency can be achieved
- (Obviously this assumes a good network quality and proximity)
- We already achieved this anywhere within the U.S.
- When expanding the audience to borderline-quality or inter-continental audiences, of course latency increases... but it's still superior to HLS/DASH.



# Live Streaming: Cloud Deployment

We have a *rist2rist* utility

- It acts as a relay. It doesn't decrypt or encrypt.
- It can live in a central cloud or in multiple CDN-type locations.
- It features the listener/auth, so can act as a gatekeeper.
- Adds no appreciable latency since it doesn't decrypt the stream.
- Thus, you can put your source and first sender on your end, multiple *rist2rist*'s in the cloud, and create a virtual CDN



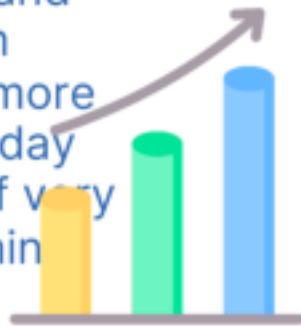
# Live Streaming: Scalability

## “Quality”

- When using h264/av1 compression, the “sweet spot” seems to be between 3 to 5Mbps for 720p or 1080p streams. This traverses most corporate VPNs and can be handled by any hardware device.

## “Quantity”

- Each rist2rist can handle up to 100 simultaneous connections and still deliver bullet proof packet loss protection. The system can scale by starting more processes. Our largest installation has more than 1500 simultaneous viewers monitoring/producing day in, day out, spread across two continents, viewing multiple streams of very high or ultra high quality. Performance, up time, ease of switching program-to-program etc. is rock solid.



# ristsrppasswd

```
x
$ ristsrppasswd --help
Usage: ristsrppasswd [username] [password]
$ ristsrppasswd username@somewhere.com MyPassphrase > /tmp/passtore
$ ristsrppasswd othername@somewhere.com OtherPassphrase >> /tmp/passtore
$ cat /tmp/passtore
username@somewhere.com: BK/egnzAkUmH5cQxpfxiNopYIu8IcJ5qmaZMsW7Rw2qYEPgzAgshhp2d4
VwfwTceEVZstrkCEgzpUTgH6wbb/IRjD1k7f1YftrTUPf1QKjttUSD6npMadCVkVmQT+3WpMVyU9Et0E
V9f2pUbW7KhXFludynruTZI5eIi6CG+rUAdkuzvzUG0tTr/UQadF1ehW7o+ElekLuNB9kGN9iCK/HHyo
zccC8p/+LyiLqdJl3nvgGn8CNNQvW7u5LAYtBE0BxNR0d8S+bbjsPn2kS6Jb75nJARDKUo//RuikWoyt
EA7jNqBew8qX0Pof9NJa08TFZh4ShAienGX808MHcTztw:EFJn6Wzs3cx0L9Pl02mRkkcsyR6nWyxsCE
vhmn8zPh8:3
othername@somewhere.com: N5XCjnVHaeVFigG0k47fnrneMjnl71PsRIu79LP9It98ShPtHEqd0CS3
Qkqx0rXSs14DUNLsnvCQgsWEdKKh+qbGcoDuWup1+N08GyWs53wjNIFLSTwXy2jUdoMko0SfvaqaWHxr
o1wBYrDNQ3pUJgTgldjhupXIMEc01GTjJy8T+ArTAMmrWg3AT86Ap8DBqqlduzYbdywygnuV9zB+xEYZ
yDA6kGkDg4RILFcUxHFJCNIT/37/NpbZ+WvPN8eb9k0hImVX8Qn5L85056pYMC4RrbzD7IjjIbFG3xn8
b/eUJwhGDuGm8dy0d+Vg5ffylJK87y9G2xJMmjLxFc09Pg:aob0wo0mfW0oB9/VEt/j7hjHAzq8PPSmK
5cT9/B/R8k:3
$
```



# Command Line: Sender

```
x
$ristsender --inputurl udp://127.0.0.1:8192 --encryption-type 256 --secret pre-s
hared-passphrase --profile 1 --srpfile /tmp/passtore --outputurl rist://@127.0.
0.1:10001
1706467862.727016|0.0|[INFO] Starting ristsender version: v0.2.7-27-g234c2e2 lib
RIST library: v0.2.7-27-g234c2e2 API version: 4.2.0
1706467862.727136|0.0|[INFO] Assigning stream-id 0 to this input
1706467862.727178|0.0|[INFO] Starting in Main Profile Mode
1706467862.727194|0.0|[INFO] RIST Sender Library v0.2.7-27-g234c2e2
1706467862.727212|0.0|[INFO] Link configured with maxrate=100000 bufmin=1000 buf
max=1000 reorder=25 rtmin=50 rtmax=500 congestion_control=1 min_retries=6 max_
retries=20
1706467862.727218|0.139954411921424|[INFO] Using 256 bits secret key
1706467862.727228|0.139954411921424|[INFO] URL parsed successfully: Host 127.0.0
.1, Port 10001
1706467862.727287|0.139954411921424|[INFO] Starting in URL listening mode (socket# 4)
1706467862.727298|0.139954411921424|[INFO] Configured the starting socket receive buffer size to 419430 Bytes.
1706467862.727305|0.139954411921424|[INFO] Configured the starting socket send buffer size to 419430 Bytes.
1706467862.727314|0.139954411921424|[INFO] Peer cname is XEON@127.0.0.1:10001
1706467862.727320|0.139954411921424|[INFO] Setting max nacks per cycle to 88
1706467862.727323|0.139954411921424|[INFO] Setting buffer size to 2000ms (Max buffer size + 2 * Max RTT)
```

# Receiver Side

- You probably want the client viewer to present a dialog box for user name and passphrase \*need to convince VLC to allow this for RIST input modules
- You probably want to handle the “secret” parameter behind the scenes
- With that, all that’s needed is to mirror the parameters set up on the RIST sender



# Live Streaming: Summary

## EAP SRP 6a Authentication

- Secure handshake, hashed password store similar to Apache, AES encryption once authenticated, key rotation, easy back-end communications with accounting or similar store for authorizations
- With the Sender in "Listen" mode enables many, many receivers
- Ability to incorporate a degree of forward error correction in addition to backwards error correction for very, very large audiences as per RIST spec
- Multicast addressing as per RIST spec

## Security of Stream

- AES encryption combined with highly secure authorization coordinated with existing billing systems provides as much as, if not more security of DRM over https transport – over much faster udp transport. And adding an option for a minimal degree of forward error correction could enable incredibly huge audiences. Note also that DRM via HLS/DASH over http/tcp can be encapsulated, if necessary, over VPN like RIST connections, though it won't enjoy as much of a speed boost

# Conclusion

## libRIST Development “Roadmap” Solid

- Improvements in “Reach” and Features
- Boast Stands: We Think We’ve Achieved the Original “Interoperability” Goal!

## Live Streaming End-to-End w/ Packet Recovery

- High Quality Source Tools Compatible with libRIST
- Supports very Large Audiences
- Security for the IP/Content





# Thank You

Live Streaming End-to-End with Packet Recovery and  
libRIST Development Roadmap

**By Sergio Ammirata, Ph.D.**