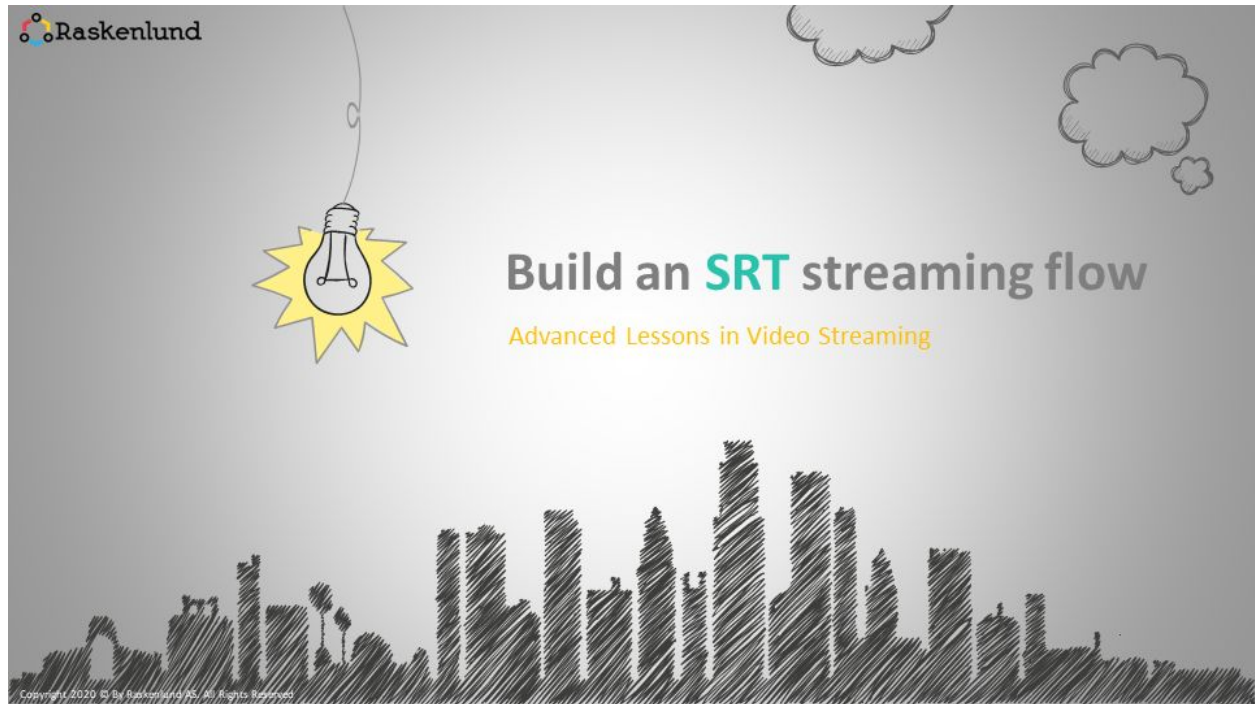


Raskenlund Webinar Transcript

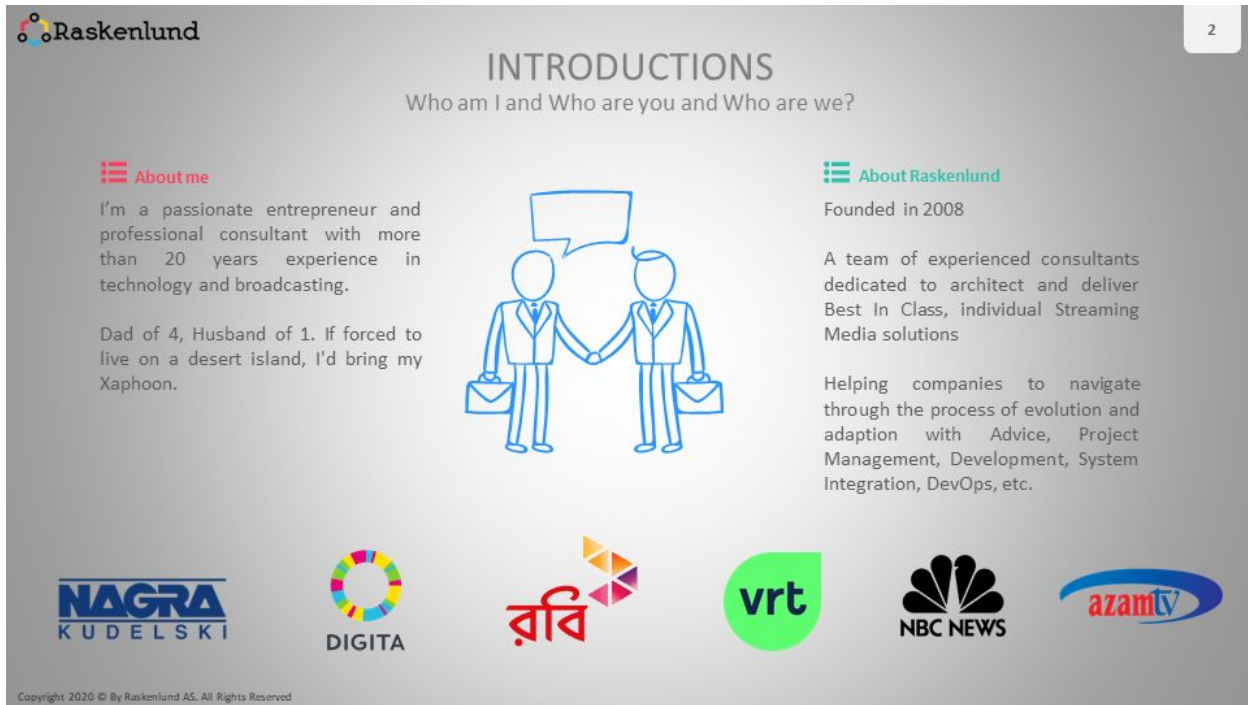
How to Build an SRT Streaming Flow from Encoder to Edge



Okay, we'll start this webinar. And I would like to welcome everyone. And thank you for making time to attend this webinar. And it's probably appropriate to say a special welcome to all the people who have tried to reach out to me in the past week as I may not have been as responsive as usual. Probably you have just tried to figure out what I've been up to. And in the next hour, I'll let you know.

The subject for today's webinar is how to build an SRT streaming flow from encoder to edge. And before we dive into the subject, I would like to give some practical information. First of all, throughout the webinar you will be able to answer a few polls. On the right side of your screen you can go to the poll section and there you will, at some point in the webinar, you will see that we have a few questions which you are free to answer and it will be very interesting and the results will be immediately shared anonymously so that you can see what your fellow participants have chosen.

Another interesting thing is that for all people who participate in this webinar, we'll offer a free consultation. So if you go to the offer section on the right, you'll be able to click a button and fill out the form and we'll contact you as soon as possible.



The slide is titled "INTRODUCTIONS" with the subtitle "Who am I and Who are you and Who are we?". It features a central illustration of two business people shaking hands. On the left, under the heading "About me", is a personal bio of the speaker. On the right, under the heading "About Raskenlund", is a company bio. At the bottom, there is a row of logos for partner companies: NAGRA KUDELSKI, DIGITA, রবি (Rabi), vrt, NBC NEWS, and azamTV. A small page number "2" is in the top right corner.

INTRODUCTIONS
Who am I and Who are you and Who are we?

About me
I'm a passionate entrepreneur and professional consultant with more than 20 years experience in technology and broadcasting.

Dad of 4, Husband of 1. If forced to live on a desert island, I'd bring my Xaphoon.

About Raskenlund
Founded in 2008

A team of experienced consultants dedicated to architect and deliver Best In Class, individual Streaming Media solutions

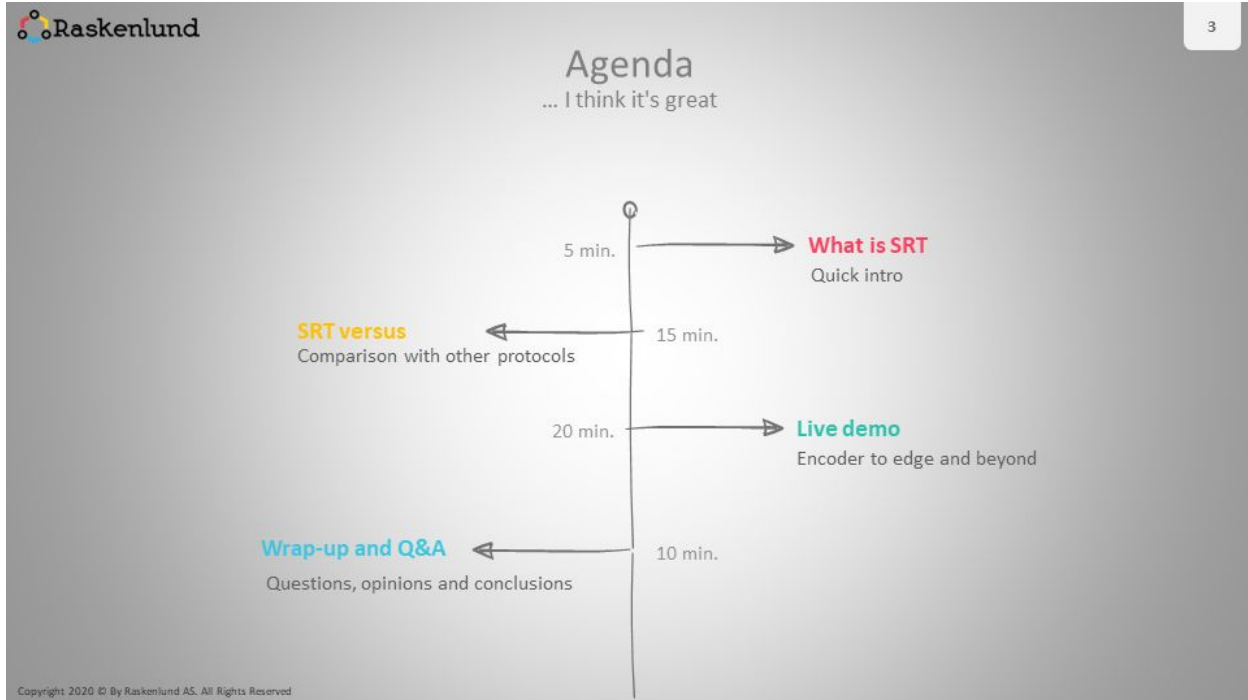
Helping companies to navigate through the process of evolution and adaption with Advice, Project Management, Development, System Integration, DevOps, etc.

Logos: NAGRA KUDELSKI, DIGITA, রবি, vrt, NBC NEWS, azamTV

Copyright 2020 © By Raskenlund AS. All Rights Reserved

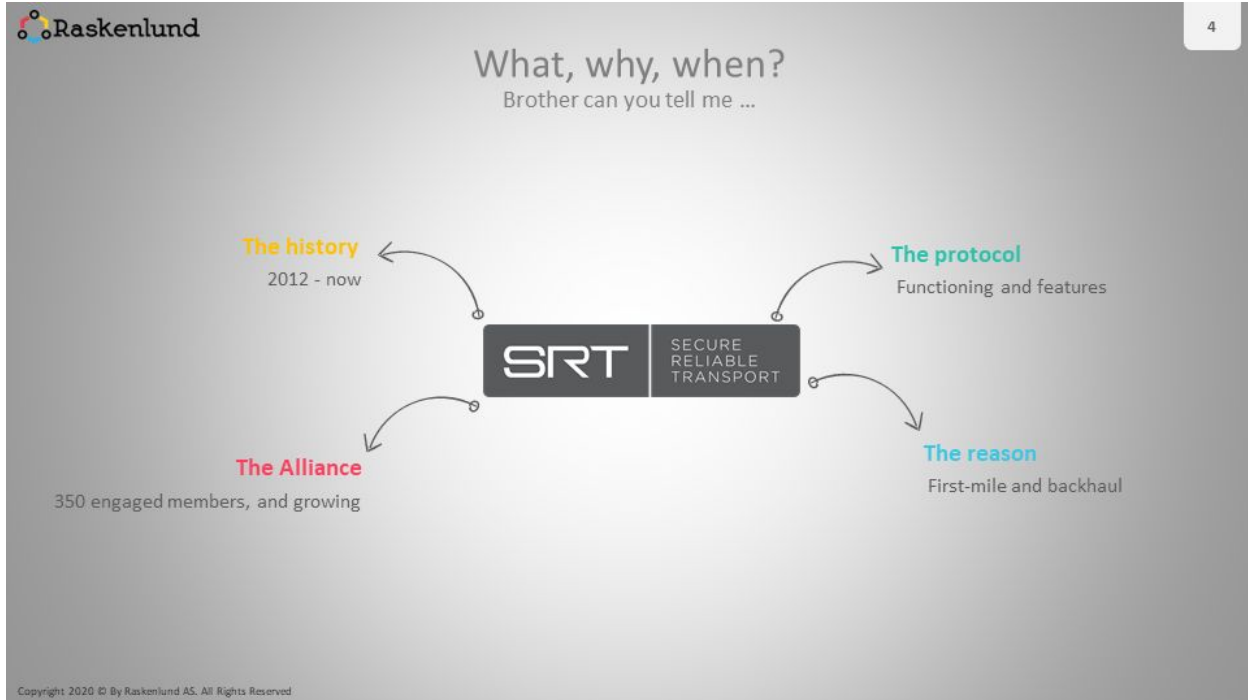
With that, let's start with a quick introduction. First a little bit about me, I started with streaming media in the late 90s. And around that time is probably Windows Media Server or Windows Media Services that was the standard. And we did some really interesting projects with real estate, with soccer, or football as we call it in Europe. And we even built a streaming media network over satellite, that was a really exciting time. And over time I grew more and more excited about streaming media and about its possibilities. And it even went this far that in 2012 or no, in 2008, I decided to make it my full time job. And that's where I found the company Raskenlund.

At Raskenlund, we are a team of experienced and excited people who really want to help you with your streaming media solutions. But our focus is that we want to make this technology work for your business. We don't want to throw a lot of software or hardware at you and put that together to a cool solution, but we want to make sure that whatever technology and software we choose really supports your business case. And therefore it's also that we very consciously have decided to be and stay an independent consultancy company, we just want to make sure that we can offer whatever is the best in class technology for your situation.



Let's go on to the agenda for today then. First of all, I would like to give you an introduction about the SRT protocol, what it is, and what you can do with it. Then I will give you some comparison of SRT with some of the well-known protocols. After that we'll go over to a live demo where I will show you how you can configure a few devices and software to use SRT. And finally we'll wrap up and hopefully have some time for questions if any questions are coming in.

If you have any questions, please use the chat window, and we'll try to pick a few of these questions in the end and answer them. If we don't have time for all the questions, we'll do our best to get back to you and give you an answer. At any time you may also drop us a message or contact us via the special offer and get some free consultations from us.



So over to the history of SRT. So what is it? Why is it required? And when is it a good fit? So SRT was invented by Haivision back in 2012. And I would say SRT was actually born out of a clear need to replace the existing protocols. The protocols that were in the market at that time were coping or were trying really hard to keep up with the technology changes in the industry but clearly designed and defined for older methods and older technologies, and we'll see that as well when we're going to do comparisons between the protocols. So that's why Haivision, back in 2012, invented SRT and found this new way to transport streaming media over the internet.

In 2017, SRT was made open source. And that really put SRT on the map and at the same time, the SRT alliance was formed by Haivision and Wowza, initially. By today there are more than 350 members of the SRT alliance and at Raskenlund we are very proud to be a member of the alliance as well. And what the alliance does is we're a community that has focused on further adoption and recognition of the protocol but also further development.

The protocol itself is open source as I already said, which means that the source code is available on GitHub. If you're a developer, you can check the source code, you're free to contribute, and if you have some ideas or if you have developed something that may be interesting, then you can put it up for approval. At this moment in this webinar, I don't want to go too much in detail about exactly how the SRT protocol works under the hood, but if you're interested in that, I can recommend a webinar that was held just last week by Haivision as part of their SRT Tuesday series. The name of the webinar is broadcast reimaged SRT. And I'm sure you'll find that when you do some Googling on the internet.



What I would like to tell you about SRT, it's mostly about its functionality and its features. And the most important features are already covered in the name, namely the S and the R. The S is for security, which is implemented by AES encryption. So that makes it pretty much possible to, pretty much impossible, sorry, to steal the data and make anything useful out of it. The reliability of the protocol is mostly because of the use of the ARQ protocol. The ARQ protocol stands for automatic repeat request. Which means that lost packets can be requested over again and then sent over again so that you can get a perfect row of packets without any loss on the receiver side.

A third feature and interesting thing of the SRT protocol is that it's really low latency. Now, the use of ARQ, typically requires a buffer, that makes sense. The receiver side, of course, you must keep a small queue so that you can wedge the lost packets back in its place. And on the sender side, logically, you must keep a small buffer so that you can resend the packets that were lost along the way and that are requested by the receiver.

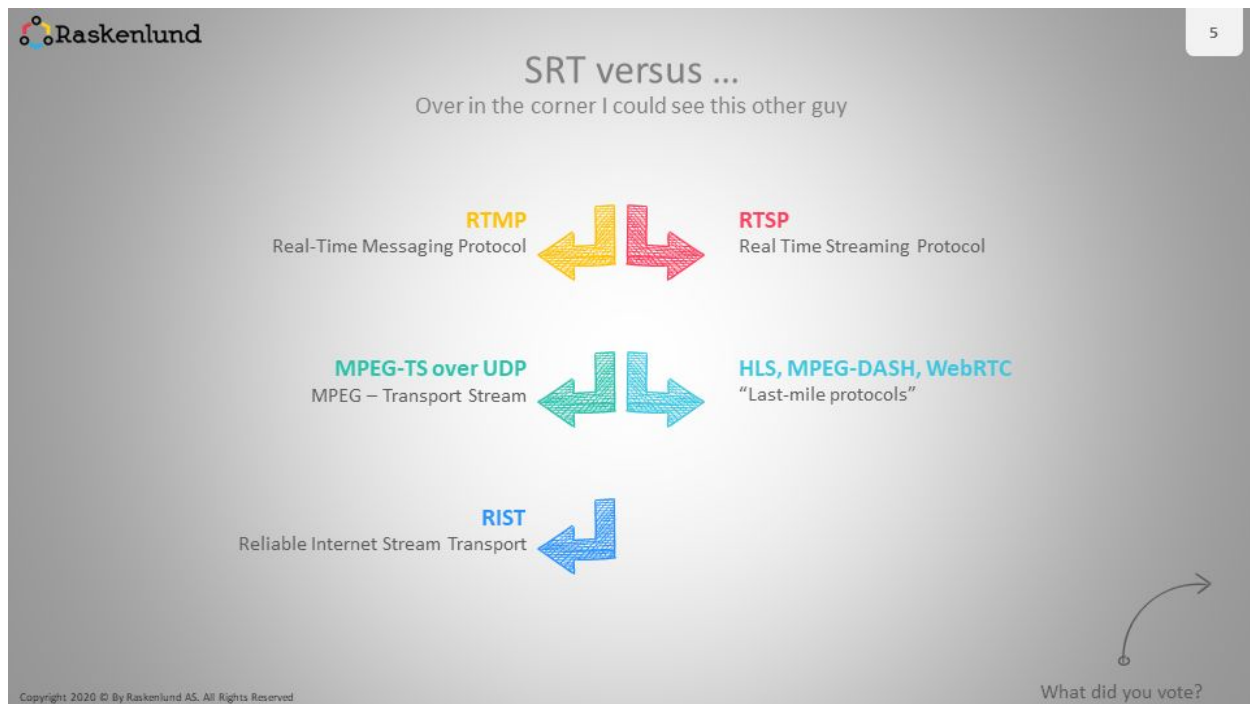
A buffer and low latency typically sound a bit contradictory, and that's true. If you have a large buffer, logically, you can't do low latency. But in this case the buffer is rather small. Typically the recommended size of this buffer is one and half times the round trip time between sender and receiver. And if you think that the round trip time between two servers in Europe is typically less than 50 milliseconds, or the round trip time between Europe and US is typically less than 200 milliseconds, then you can see why this is rightfully called a low latency protocol. Because we're always talking about sub-second latency. And that even counts for really long distances.

In a recent test that Haivision did they sent a stream from Germany all the way to Sydney in Australia, they received a round trip time of approximately 360-400 milliseconds and that means that they set a buffer of about 600 milliseconds. And that is still well below the second. There are some other interesting features that have recently been added and that soon will be added to the SRT protocol. One of which is that multiplexing has been introduced in version 1.4. And that means that you can send multiple streams or multiple packets over the same port instead of having to open a separate port for every stream. And that may of course sound like music in the ears of the IT department because they will have less to do.

Some things that will be introduced in the next version and you can find these in the experimental branch in the GitHub repository are for instance load balancing, connection bonding and seamless switching. And I think these are some very exciting features. I'm personally especially interested in the connection bonding to see how we can send a stream over multiple connections. So, if we have listed all these features, I think it could become clear why SRT is really such a great protocol.

When is SRT interesting to be used? Well, mostly on the first mile site and on the backhaul site. And the first mile is between the fuel production and the broadcast facility. Or between your encoder and your origin server. And the backhaul is between your servers or between your data centers. This is really where the SRT excels.

At this moment, SRT is not so much interesting for a last mile protocol, there are other protocols that we know very well there, like HLS, MPEG-DASH, most lately WebRTC has become really popular. SRT is not really a good fit yet there. It may be in the future, but for instance, because there is still very limited support for SRT in web browsers. It's not a good match to use SRT to the players. If you still wonder why SRT would be a good fit, then I would almost ask why would it not be a good fit? Especially if you look at the other protocols. These advantages that the protocols have and the advantages that SRT has. So let's take a look at these.



We start with probably the most known and the most used protocol for encoder to server and server to server communication and until about five to 10 years ago, also probably the most-used protocol for a client playback when we still had Flash in our browsers. And that is RTMP. RTMP is by now just over 20 years old. It was prototyped by Macromedia which later was acquired by Adobe back in 1999. And one of the advantages is that because it's such an old protocol, it's also very mature, it's a very well established protocol and not at least it's widely supported. I think that pretty much every media server, every encoder can do RTMP. And that of course makes that it's a logical choice or at least it has been a logical choice for many years for a lot of streaming solutions. However, as I already told you in the start, the old protocols can't really keep up with the technology changes and one of these technology changes is the new video and audio codecs that are being launched. RTMP is really tightly connected to the video and audio format that you can use. So when newer formats are coming, you may not be able to use RTMP for that. And there have been some attempts and some implementations that make RTMP suitable for transmission of HEVC et cetera. But that's not a part of the standard. And for your transport protocol, you really want a protocol that is codec agnostic that can send anything



over the line and that makes it future proof so when new video formats and audio formats are coming, you won't have to change your transport protocol.

Another issue with RTMP is that it works quite well for server to server communications over shorter distances, but if you are trying to send a stream from Europe to Asia pacific, or try to send a stream from up in the north to down south, you will see that first of all, there is a huge latency, and secondly, the RTMP protocol, because of that it's built on TCP, and requires all these acknowledge packets, it will become so slow that eventually it may even fail. SRT does not have these problems. Because the use of UDP, because of the use of ARQ, it will perform just fine also over longer distances. And even keep this very low latency as I've talked about.

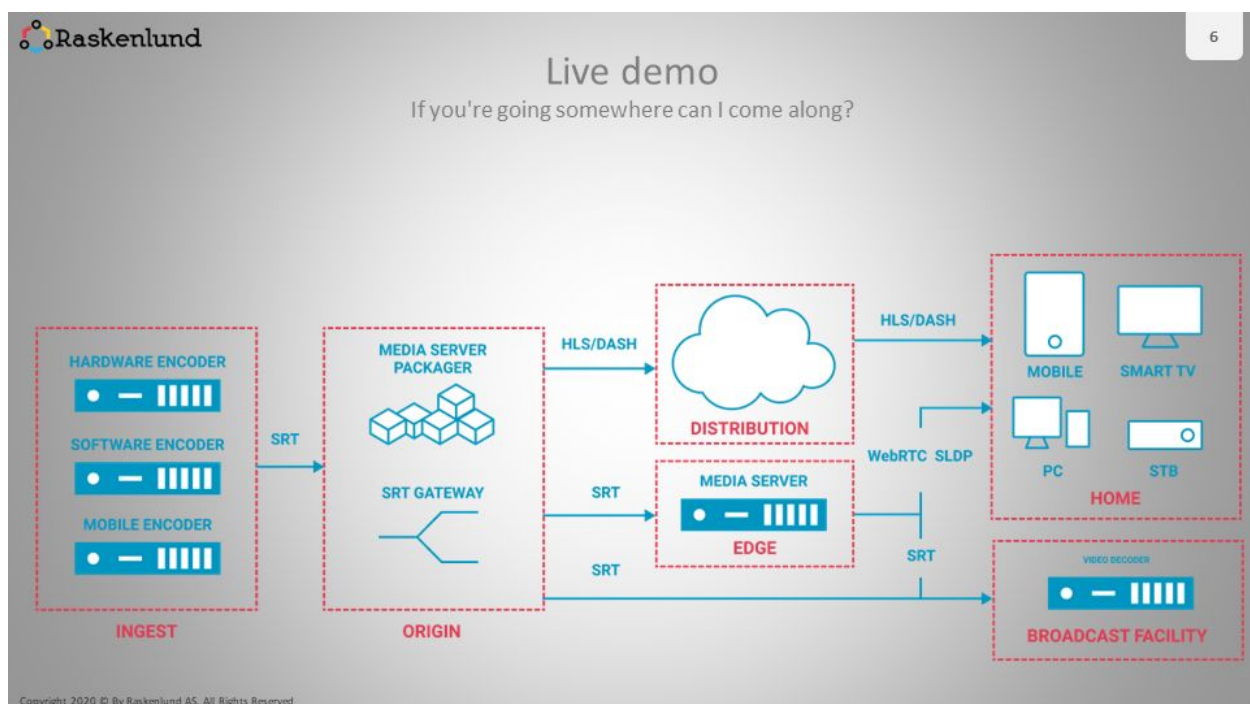
So the next protocol that I would like to talk about is RTSP. RTSP is actually even older than RTMP. The first draft was submitted to IETF in 1996. And RTSP is mostly use in IP cameras, that's where we see a lot of RTSP. RTSP actually uses RTP to transfer the data. And one of the advantages of RTP is that you can run it both over TCP and UDP. However, that is also its disadvantage. Because if you want to use TCP over very long distances in real time, you'll have this problem of the acknowledge packets that must be sent back all the time, which makes the protocol slow. Whereas, if you want to use UDP, you'll have no control over where the bytes are going and if there is any packet loss and there is no packet recovery. So, you'll always have this trade off where you have to choose for either of them and neither of them is really a good solution.

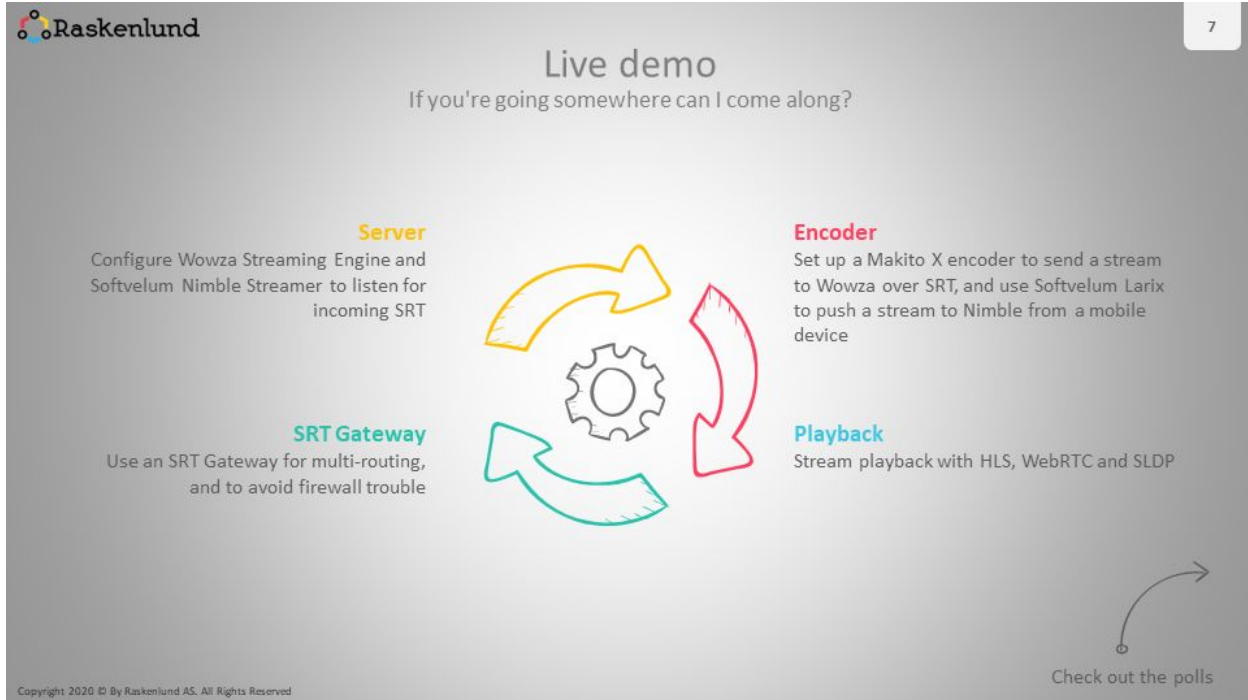
The third protocol or solution that I would like to compare is the MPEG-TS over UDP. And I would say probably MPEG-TS over UDP is the most straightforward solution that you can use for streaming. You can just let any solution put the bytes on the UDP port and send it to the receiver. However, the fact that it's so straightforward, also makes it so vulnerable. And because it's using UDP and as I tend to say, UDP is a protocol that doesn't care, it just throws the bytes out on the internet and it really doesn't care if somebody gets it. It really doesn't care about packet loss. And if there is any packet loss, there is no possibility to retrieve the packets. So, that makes MPEG-TS over UDP a really unreliable solution and not at least when you are using it over the internet. It may work well in your local network or in a controlled environment, but internet is of course a very unreliable and unpredictable infrastructure where you really should go for a more reliable protocol.

As for HLS, MPEG-DASH, WebRTC and a whole lot of other protocols, I have bundled them together under the name last mile protocols. As I already mentioned, SRT is not really a last mile protocol, not yet at least, we don't know what the future will bring. But if you want to do last mile and you don't care about latency, then HLS and MPEG-DASH, HTTP-based protocols are really a good choice because it's so easy to transport and the possibility to cache.

For low latency, as I mentioned, WebRTC has really become popular, and there are some other low latency implementations of course, actually in the demo we'll show Softvelum's SLDP protocol, and there are many companies who can offer you low latency solutions.

Then over to RIST and RIST is really what you could call a new kid on the block. It's a younger protocol than SRT. And it's developed under the Video Service Forum Activity Group. Interestingly enough, the inventor of SRT, Haivision, is also a member of this video service forum. And is also actively involved in the development of RIST. So why invent a new protocol if you already have SRT? Well, this was mostly because the RIST protocol really is aiming towards becoming a standard. And already now the technical recommendation has been submitted and is up for review and the RIST protocol is based on standards from for instance, SMPTE and IETF. That can both be an advantage and a disadvantage. The advantage is that once it's approved as a standard, of course it will be a really safe choice for a lot of companies, especially maybe for those who want to build it in their hardware. However, since it takes time to approve the standard and since it takes time to approve the features, the development of RIST will most likely be much slower than the development of SRT. And we see for instance that RIST lacks some important features that SRT has for today. For instance, RIST can only push the stream out to the receiver. Whereas with SRT, you can either push the stream or you can pull the stream. Or as we call it in SRT, you can have a caller or a listener. And RIST does not support encryption yet. RIST does not support multiplexing, and RIST only supports RTP for data. And these are features that are currently lacking in RIST and I expect that they will come, but as I said, it will just take time before these features can be implemented. Whereas with SRT, first of all, many of these features already are implemented, and secondly, because of the SRT community, because of the SRT Alliance, and because of SRT being open source, it's possible to implement features and functionality much faster. Now that we have seen and heard a lot about the different protocols and about the different possibilities, let's move on to a live demo.





What I would like to show in this demo is how you can build your flow from the encoder to or through the origin to an edge server and then eventually play it back on a few devices. So, first of all, we'll configure the server and make sure that they can listen for incoming SRT. And then we'll configure encoders to push a stream to the video server, and then we'll do a quick playback just to have proof that it actually works. And eventually I will also show you the high efficient SRT gateway and why it can be useful. So let me share my screen. And let's dive into the demo.

Here I have a Windows media server that has been set up with a really default set up. In Windows Streaming Engine, if you want to use SRT, the first thing that you have to do is that you have to define the configuration. An SRT definition is created as a stream file. So what I can do here is I can add a new stream file. Give it a name. And provide the URI. In this case I want the SRT to listen on all interfaces. On port 10,002. I can set some of the properties. Of course, it's interesting to edit the encryption properties. Set a key length of 16 characters and then fill out a new key. And I can add some other properties. I can set this recovery buffer that we talked about. The default is 400 milliseconds, if your servers are rather close, then you can make that a lower number. If your servers are on a large geographic distance, you can increase that number. And when you go back to an overview of our stream files, we now have this configured and I can start that. And choose SRT as the MediaCaster type. Click okay. And when I go over to the incoming streams, you will see that there is right now an active stream with a status waiting for stream because there is currently no encoder that is sending the stream.

So let's go ahead and do that. I'll right now take OBS studio which is a free product and surprisingly or interestingly supports SRT. So I'll go over to settings in my OBS studio. Stream.



Write the name of my Wowza server. Port 10002. And at the passphrase as a parameter. Then I click okay. And I start streaming. After a second or two we will see the green button here which shows that the stream is on. And if I now go back to my Wowza server and click refresh, you will see that I have an active stream. For the sake of this test, I have actually enabled the Wowza transcoder. So that we can transcode the audio to Opus, which allows me to give a quick demo of playback with WebRTC. So here I have set up the WebRTC playback with Wowza. And I've reconfigured it already, the only thing I need to do is add the stream name with the Opus suffix to make sure that I get a transcoded output. When I click play I should get the stream in just a second.

Of course I chose the wrong stream name, it's stream 2.stream. Let's try once more. There we go. So, without further specific optimization or tuning, you can see that the latency currently is approximately three seconds. And it's possible to get that further down, that's outside the scope of this webinar. But we can see that it's decent low latency. And again, of these three second latency, the SRT protocol itself only stands for 400 milliseconds. So I have not optimized my WebRTC in Wowza.

Let's go back to the Wowza server and stop this stream for a second. Now I would like to show you the same thing just with some different software. Over here I have installed a Nimble streamer which also supports SRT. And when I go to the Nimble streamer live stream settings, I can choose MPACT TSN and add a SRT stream. And here I can choose the receive mode, either listen or pull or caller as we call it in SRT. Or rendezvous. Which means that whoever makes the first call will be the caller and the other one will be the listener. I'll set it to listen on all interfaces. And at the local port, 10003. I'll call this one stream three. And since I want to add an AES encryption to this as well, I'll add a custom parameter that is called passphrase and I will add the key there. As you see, as soon as I set the custom parameter passphrase, the key length is automatically added and set to a default of 16.

I also want an output stream application live and I'll call this first stream three. It takes a few seconds for this stream to be synchronized with my Nimble stream server, because I am using Softvelum's online service, WMSPanel to configure my Nimble streamer. It's also possible to configure Nimble streamer directly in the console via the configuration files. But using WMSPanel is a really easy way and shows exactly the status of the stream. So it will take a few seconds

In the meantime I'll go over to my iPad and just start an app there that I have installed. Unfortunately, I have not managed to find a way to mirror the screen of my iPad so I'll just tell you what's going on over here. I have installed an app that is called Larix. Larix is an app that is developed by Softvelum and you can use it to send a stream to a media server and it supports SRT. So let me just quickly show you here, may be a bit small characters but this is the settings of the configuration in Larix. I'll go over to my iPad and just verify the connection. One second please. Let's test. Okay, that shoots back to the Nimble server. And right now I'm streaming a live stream from my iPad with Larix.

When I go to the overview of the live streams in Nimble, I see that the stream three that I've just configured has a green check mark, which means that it is online. I can check the outgoing stream and see that it has recognized the audio codec to video codec, et cetera. So, the next thing I have done is I have set up a test page with the SLDP protocol that is developed by Softvelum, which is a low latency protocol as well. When I play back we will see that this is live from my iPad with approximately two seconds of delay. I'll just make sure that I click play, there we go. So that is from my iPad over SRT to the Nimble server. And from there with SLDP to the player.


Another possibility is to use Haivision's own hardware. And I've been so lucky that Haivision has let me borrow one of their Makito X encoders where I have configured the stream to be sent directly to the Wowza server. So as you can see here, I have set up the stream to use TS over SRT. I have set up this as a caller. The address of my Wowza server and the destination port. And when I switch back to my Wowza server, of course I have to make sure that I have this stream file started, so the name of the stream is in this case srt10001. I'll start this one, choose SRT. And immediately we can see that it is already active because the Makito encoder has been configured to send the stream to it.

Now of course as I already told that SRT uses UDP which requires ports to be opened, multiplexing can be a nice solution if you have limited availability of ports that you can open in the firewall. But another possibility, because of SRT supports this listener and caller system, you can choose to put an SRT gateway in the middle. And I have installed an SRT gateway here and an SRT gateway allows you to listen for both the sender and receiver, and to route the streams to multiple destinations, one or multiple destinations. So as you can see here, in this gateway, I have set up a single encoder in this route and then two destinations. One of which is the Wowza server that we already know. And the other one is the player pro app, which is a free app that runs on iOS, available from Haivision. Which is a really great app to monitor your SRT streams.


So, let me configure my Makito to send a stream to the Haivision gateway instead of to the Wowza server. And what I need is the address of the gateway. I'll go to the Makito and change that address and I'll stick with port 10001. And that's all. Now I'll go back to the Haivision gateway and it may take a few seconds before, and there we see, before the dock turns green is what I wanted to say, but it went even faster than I expected. And at this moment, I have the Makito X encoder who sends the stream to my gateway and then the output is both to the Wowza server and the player pro app. So when I go back to my Wowza server I should still have an active stream, which I indeed do have, except now it comes from the Makito gateway, and from the Haivision SRT gateway instead of directly from the Makito. That opens for a lot of possibilities especially if you want to send the stream to multiple destinations or if you have these firewall issues.

SRT Hub

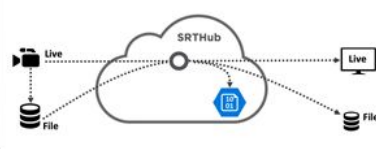
Media Routing in the Cloud




Secure, Reliable Global Routing



Live- & File-based Workflows




Built for Broadcasters




Haivision

Hublets | Connected Open Partner Ecosystem



APIs for
Broadcast Cloud
Workflows




Cloud-native
Micro-services
Architecture

Extensible
3rd-party Services


Input Hublets

Are those that generate the content.




Output Hublets

Are the output components of a route.



Processing Hublets

Those are used to make transformations to the content.

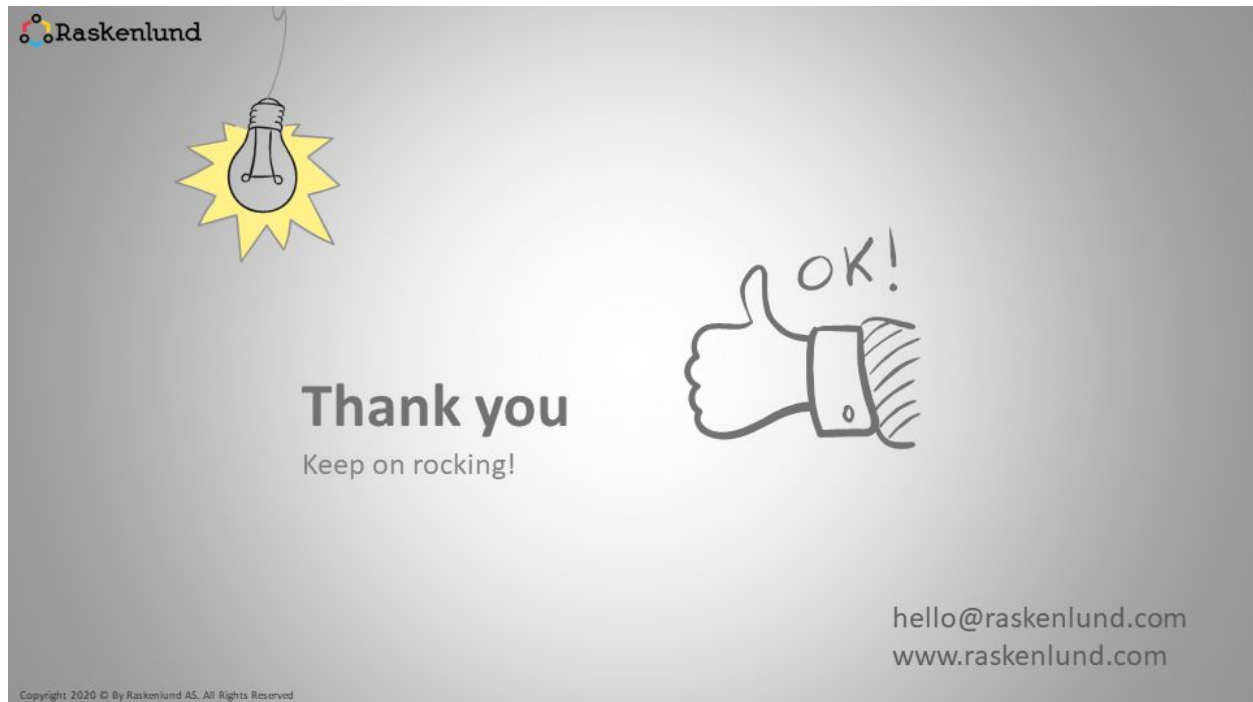


Haivision

One last thing that I want to mention that it's not possible to hold a demo of yet is the SRT hub. The SRT hub is, I would say with my words that it's an infrastructure as a service. It's powered by Microsoft Azure and uses their network and allows you to define your SRT connections in the SRT hub. And also allows you to process the stream in the hub. And I have received a few slides from Haivision about SRT hub that I'd like to share with you that really makes it clear what the SRT hub does.



It's a possibility to define and design your workflows in a really easy GUI. And to connect your end points and to connect any media or streaming processing in between. And as you can see it has three kinds of hublets, as they call it. The input, the output and the processing. So you imagine that maybe you want to do some transcoding in between and you might have all kinds of input hardware or software, and you can connect that together in the hub, and the Azure infrastructure makes sure that your stream is transported safely and sound over the internet.



And with that, I would like to see if there have come in any questions. There have indeed and I'll just have to pick a few random and as I said, we'll go through all the questions that have come in and we'll try to answer them one by one afterwards.

So one questions from Tyler is what are my thoughts on the SRT Live transmit app that is available from the GitHub repository? And personally, I think it's a great tool. But mainly used to show how you can transmit data if it's SRT. The nice thing with that app is that it really doesn't care what type of data you send over the line, so you can use it for file transfer, or you can use it for streams, of course. But I would say that, I would personally not use it in a production environment because it's really meant to show the capabilities and possibilities of SRT and it needs some more work if you want to use it in production. I hope that answers your question.

Let me see if I have a few other questions that I can answer. So there's a question, if the websites and services that I demoed here are available on a month to month subscription? Most of them are. Wowza is available as a month to month subscription. A Nimble server in itself actually is a free product from Softvelum. As is Larix, which you can download for free from the app store or from the Google Play store. However, the configuration from WMSPanel is a



monthly subscription. And for any information about the Haivision products, I would like to refer you to Haivision, they are the ones who know best about that. However what I do know is that the Haivision SRT gateway is available for instance in Amazon Web Services in the marketplace where you can use it as a pay-as-you-go subscription. Which may be really interesting.

A question from Patrice who asked if there is an existing open source streaming server implementing SRT? The one that I have been following lately is a media server called Oven Media Engine, which I think is a really important development. And that one does support SRT, but only partial, yeah, so you can't use all of the features from SRT. But I really find it an interesting development. Apart from that, I personally don't know any media servers that support open source media servers that support SRT. Right now I'm thinking that Ant Media Server may do it. But I'm sure that if you do some Googling, you'll find a lot of interesting projects.

One last question that I'll handle and then the rest I'll answer later. The question from Thomas who says, "is it possible to reach the WebRTC low latency with SRT?" and yes it is. The fact that I showed you three seconds latency in my demo is simply because I have set it up relatively quickly and not spent a lot of time on trying to tune this really for the lowest possible latency, but WebRTC in itself is also a protocol that is capable of doing sub-second latency. And so it's a matter of tuning simply. Then I get a nice point or a nice comment from Mark who says that MistServer may support SRT, so that's also one that you can look into.

Okay. With that I would like to remind you once more about the possibility to get some free consultancy and thank you for attending. Please allow me to mention Ruben from Geezer Agency, he is my marketing expert and has done a fantastic job in helping me. And I would like to thank Selwyn Jans from Haivision who has provided me with the Haivision equipment, and who has provided me with a lot of useful information. But most of all of course, I would like to thank everyone of you for attending. And if you have any questions, please feel free to keep chatting for a while, or send me an email at hello@raskenlund.com.