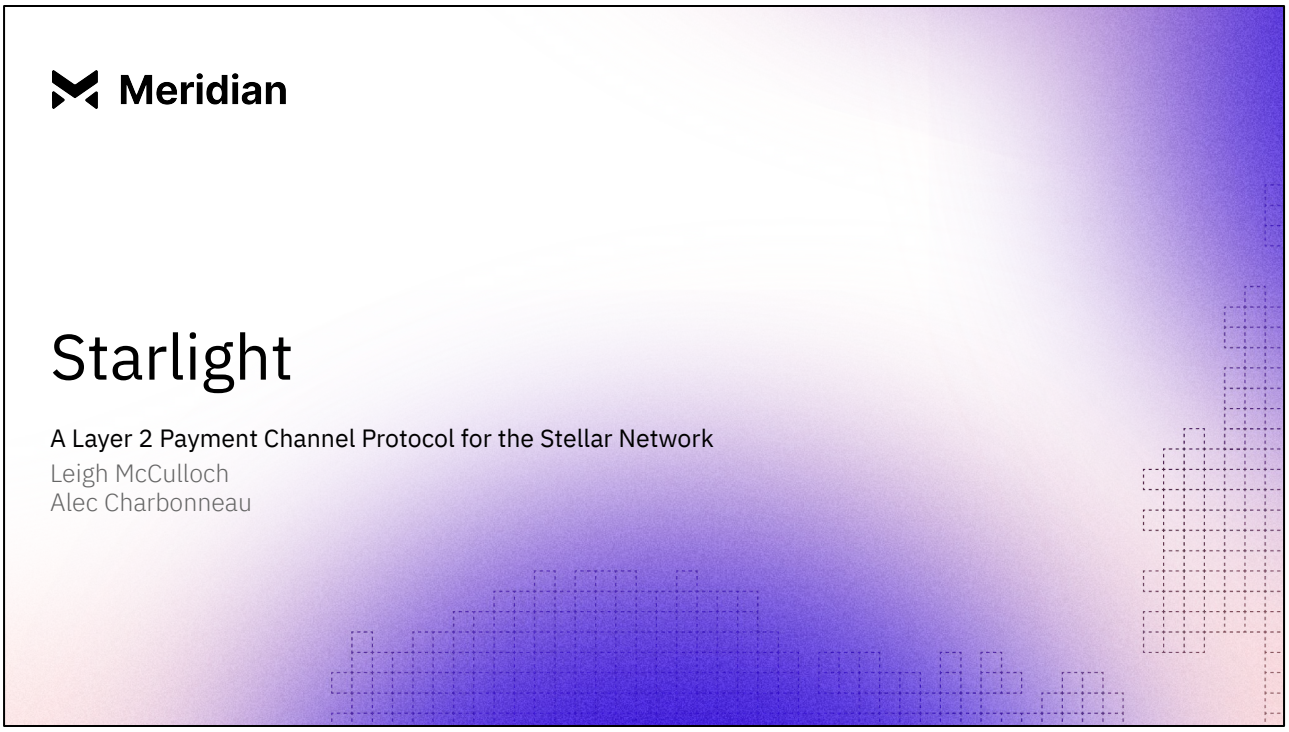


Starlight is a layer 2 payment channel protocol for the Stellar Network.

Welcome

Meridian

MEET LEIGH

Leigh McCulloch is a Principal Software Engineer at the Stellar Development Foundation (SDF), a non-profit organization that supports the development and growth of Stellar, an open-source network that connects the world's financial infrastructure. He leads forward-looking cross-functional engineering projects for Stellar.

I'm Leigh McCulloch.
I'm a principal software engineer at the Stellar Development Foundation. I joined the engineering team in 2019. And I work on ecosystem standards, core advancement proposals, and experimental projects, like Starlight.

Welcome

Meridian

MEET ALEC

Alec is a Software Engineer on the SDF Product Engineering team. Alec has been working on products for SDF for over a year, and has contributed to projects such as Laboratory, Vibrant, and Data. Currently Alec is focused on Stellar's Starlight project.

Hi I'm Alec Charbonneau.
I've been with SDF for about a year and half, and am part of the Product Engineering team. I've worked on a couple different SDF products such as the Lab, Vibrant, and Data projects.

Agenda

Meridian

1. Why Layer 2

2. What are Payment Channels?

3. Demo a Starlight Payment Channel on Stellar

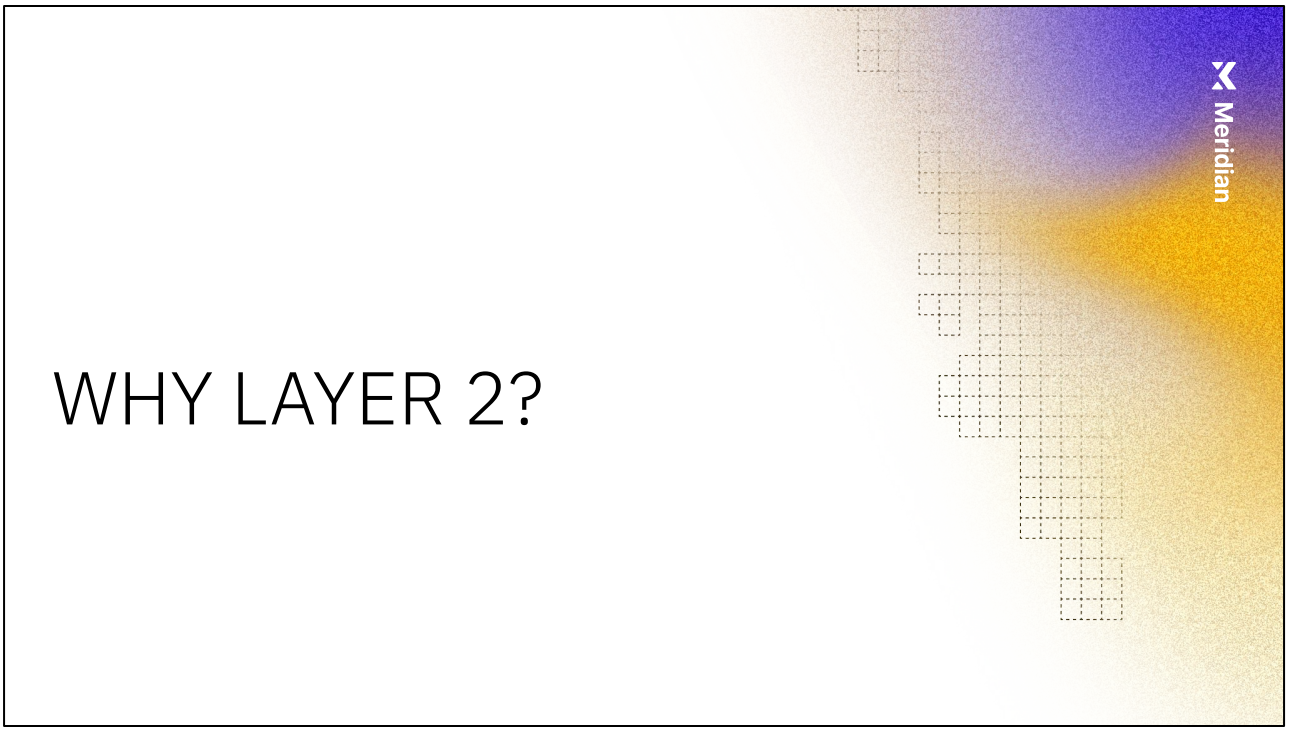
For the agenda,

First we'll explain why layer 2 tech, like Starlight, is relevant to Stellar.

Next we'll explain what payment channels are.

And lastly we'll give a demo of a Starlight payment channel, and explain how it works as we're using it.

Along the way we'll see how two new proposals, CAP-21 and CAP-40, make Starlight possible, simple, and efficient.



Why layer 2.

We've been experimenting with layer 2 to see if there are capabilities we could bring to Stellar users.

Specifically, Stellar users who need to process a large number of payments.

Processing a large number of payments has an upper limit on layer 1 networks that guarantee consensus and provide fast transaction confirmations, like Stellar.

TRANSITION: The Stellar network today is configured with an artificial limit of...1000 operation per ledger.

The Stellar Network

Why Configure With An Artificial Limit?

Meridian

Configured Limit

1,000
operations
per ledger

- Voted on by network validators
- Promotes financial inclusion
- Promotes decentralization
- Preserves cost predictability
- Broadens who can run the network

TRANSITION: The Stellar network today is configured with an artificial limit of...1000 operation per ledger.

This limit is controlled by the network and something that validators on the network can vote on.

What this limit does is prevent more transactions from being distributed when the limit is reached.

It does this even if the network is capable of processing more.

The network sets this limit to promote financial inclusion, and promote decentralization. It ensures that the community running the network, the people who are running validators, or hosting archives, are able to do so with predictable costs, and with affordable and readily available hardware.

This widens the funnel for who can run the network.

This is really important to us because Stellar is a borderless and open network. And our mission at SDF is to use the Stellar network to create equitable access to financial services.

TRANSITION: Ledgers on Stellar occur every...5-10 seconds.

The Stellar Network

Current Configuration

Meridian

Configured Limit

1,000
operations
per ledger

~5-10
seconds
per ledger

TRANSITION: Ledgers on Stellar occur every...5-10 seconds.

If we spread that 1000 operations over 5 seconds.

TRANSITION: We get approximately...200 TPS.

The Stellar Network

Current Configuration

Meridian

Configured Limit

1,000
operations
per ledger

~5-10
seconds
per ledger

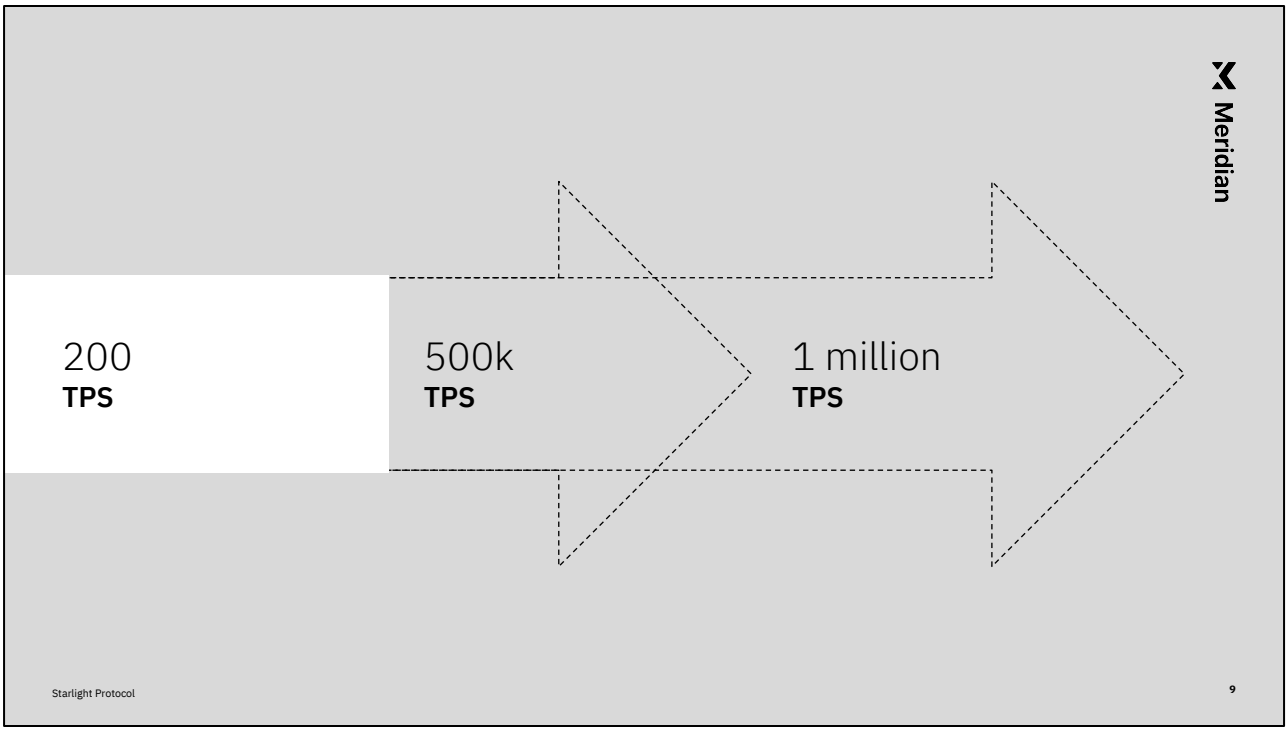
≈ 200
transactions per
second (TPS)

TRANSITION: We get approximately...200 TPS.

We've been exploring what use cases future users of Stellar might have.

Specifically use cases with high-throughput requirements.

TRANSITION: By high-throughput, I mean volumes that exceed these limits by...many orders of magnitude.



TRANSITION: By high-throughput, I mean volumes that exceed these limits by...many orders of magnitude.

We've been exploring how in these use cases, users can secure payments on the Stellar network without performing each individual payment on the Stellar network.

We've been considering many different use cases. But we've focused on the use case where a user needs to move large volumes of payments directly with another user.

TRANSITION: That's why we've been experimenting with...payment channels.

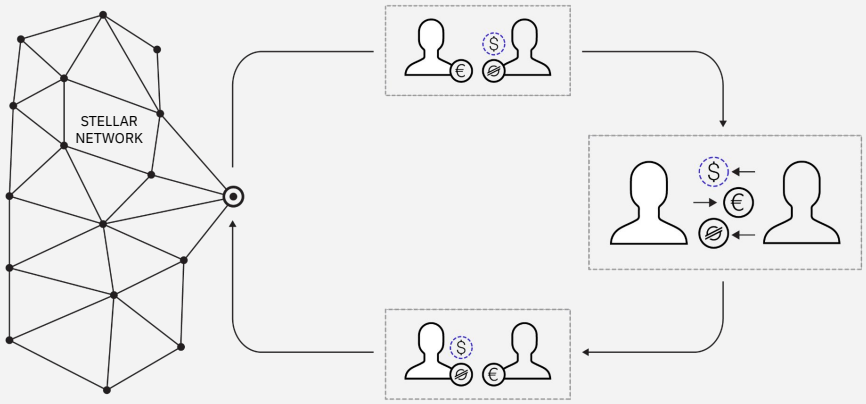
TRANSITION: That's why we've been experimenting with...payment channels.

So, what are payment channels in general?

Payment Channels

Enable Fast Payments Between Two Parties

Meridian



Starlight Protocol

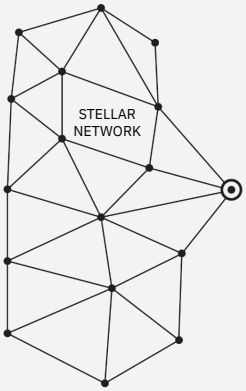
Payment channels are a layer 2 technology that enables fast and instantaneous payments between two parties.

The payments occur off-network, but don’t require either participant to trust each other.

Payment Channels

Enable Fast Payments Between Two Parties

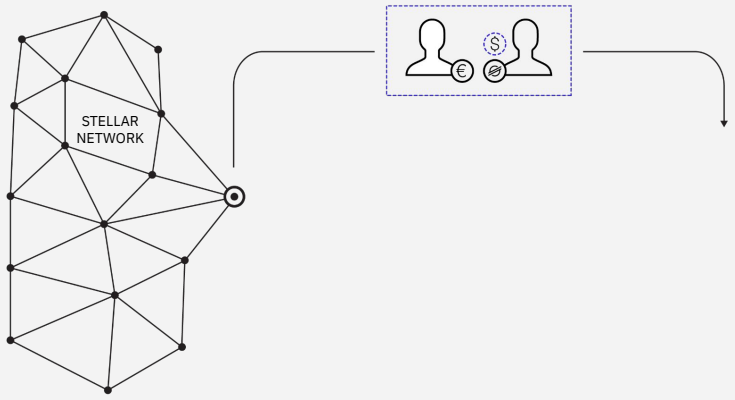
Meridian



Payment channels can do this because of the cryptography and security of the underlying blockchain.

Payment Channels

Enable Fast Payments Between Two Parties



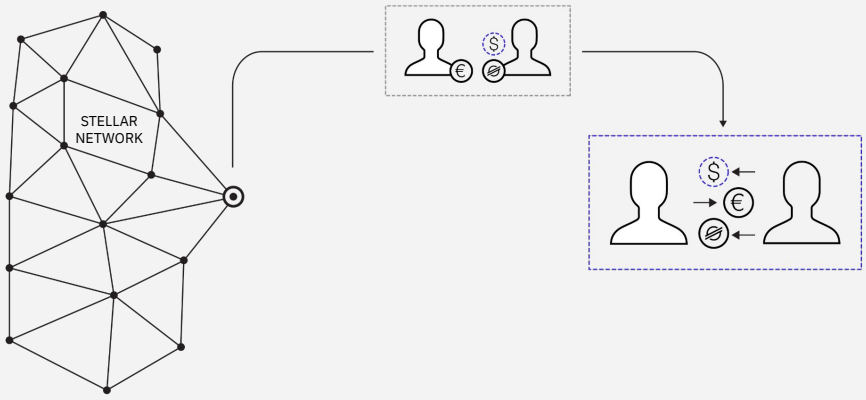
Meridian

Starlight Protocol

To open a channel participants first make an on-chain deposit into a channel.

Payment Channels

Enable Fast Payments Between Two Parties



Starlight Protocol

14

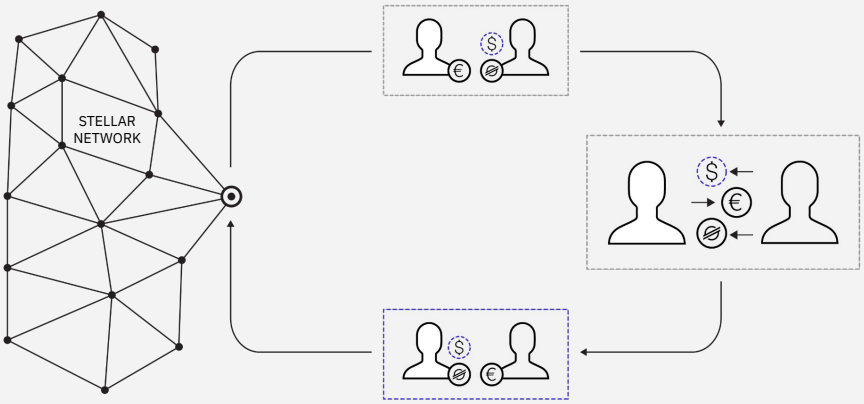
Within the channel, participants can send each other payments. These payments are done off-chain, so the frequency at which the participants can send to one another is much higher than if they were using the layer 1 network alone.

For each payment, the participants are forming a new agreement that gives a new balance each participant will receive when the channel closes.

Payment Channels

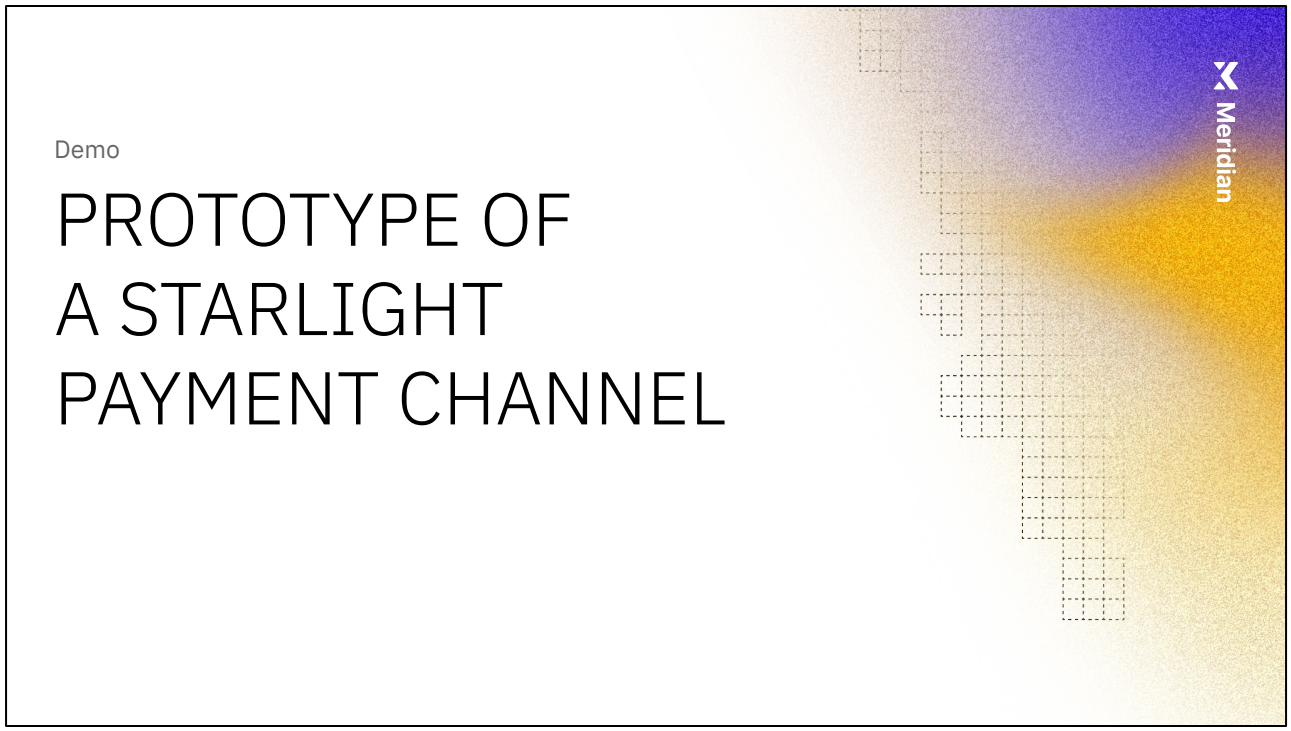
Enable Fast Payments Between Two Parties

Meridian



Starlight Protocol

When the participants want to close the channel, all the payments that happened between them are rolled into a final net-settlement amount, that then gets submitted to the blockchain.



Leigh and I are now going to demo a prototype of a Starlight payment channel.

For the demo, we have a private Stellar network running, that includes prototype implementations of CAP-21 and CAP-40. Which are two new proposals for the Stellar network.

These are core advancement proposals currently in development. And they add new capabilities that we'll discuss as we encounter them.

In this demo we're going to open a channel, send some payments back and forth, and then close it.

While we're doing that we'll explain in detail how Starlight payment channels work.

<DEMO

Alec and I both have a copy of the demo console application running locally.

Our applications are connected to each other over a TCP connection, and will communicate using simple binary messages.

On the left you can see my console where I'll be entering instructions into the application.

Alec is running the same console and he will also be entering instructions at his computer.

On the right we have a dashboard that'll display the state of the channel on the Stellar network, as well as the state of the payment channel off-network.

The graph at the top of the dashboard shows us transactions that happen on layer 1, on the Stellar network. It'll be relatively quiet because we're running this demo using a private test network that's running here on my computer in a Docker container.

If you'd like to replicate this demo and poke around it's as simple as running the Stellar network in Docker, and running our console application. We'll have a link to that at the end of the talk.

The first thing we can see on the right is two charts, one for each Stellar account that

will hold assets for the payment channel.

Let's pause for a second and talk about why we're looking at charts of two Stellar accounts.

The process of opening a Starlight payment channel begins with two Stellar accounts on layer 1.

Each participant in the channel creates and starts out fully controlling an account.

These accounts hold the funds of the channel, which will be redistributed during the close.

The blue and orange charts show us the signers for the two Stellar accounts.

We can see that both accounts have a single signer. I am the sole signer of my account. Leigh is the sole signer of his account.

Now let's open the channel and see what changes.

Ok, I'm running the command to open the channel.

DEMO: open USD

The chart at the very top of the dashboard shows that the open transaction has been confirmed in a ledger on the Stellar network.

Below we see the signers for the two accounts have changed.

The accounts are now both multisig accounts, as both accounts have two signers.

Leigh and I are now joint signers on each account. That means that we both must sign transactions going forward.

PAUSE

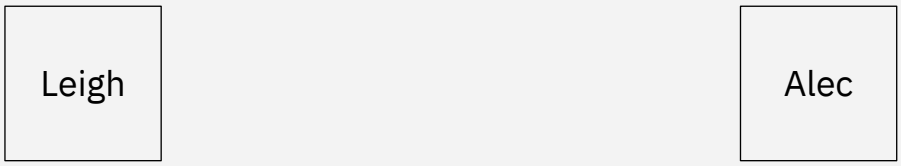
Let's go over how this open process works.

SLIDES>

Opening a Payment Channel

Involves Signing Three Transactions

Meridian



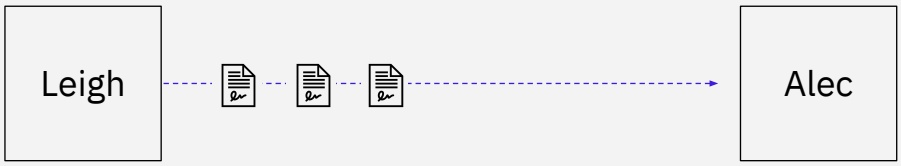
SLIDES>

When Leigh opened the channel his application generated three transactions.

Opening a Payment Channel

Both Parties Sign Each Transaction to Authorize the Agreement

Meridian



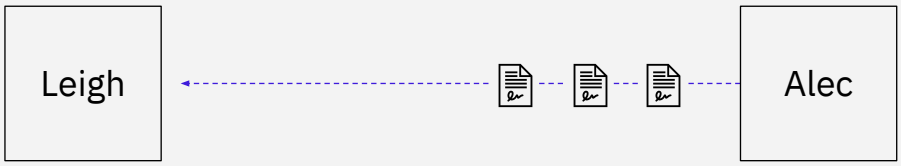
These three transactions form our open agreement.

Leigh and I both sign each transaction to make the agreement valid. During this process we both share our signatures in a single message so that either of us can attach all the signatures to the transactions and submit it to the network.

Opening a Payment Channel

Both Parties Sign Each Transaction to Authorize the Agreement

Meridian



When I received Leigh's signatures, my application validated the agreement, signed the transactions, and sent back my signatures.

At this point we've both just signed the open agreement.

Opening a Payment Channel

Defining the Three Transactions

Meridian

The three transactions I've been referring to are the open, declaration, and close transaction.

The open transaction does what you think it would by its name, it opens the channel. It accomplishes this by changing both Stellar accounts into multi-sig accounts, which we saw on the charts.

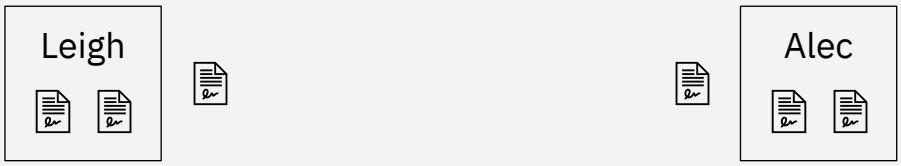
The declaration and close transaction together form a close agreement, which can be used to close the channel.

We'll explain how close agreements work when we do a payment.

Opening a Payment Channel

Submit Open Transaction to the Network

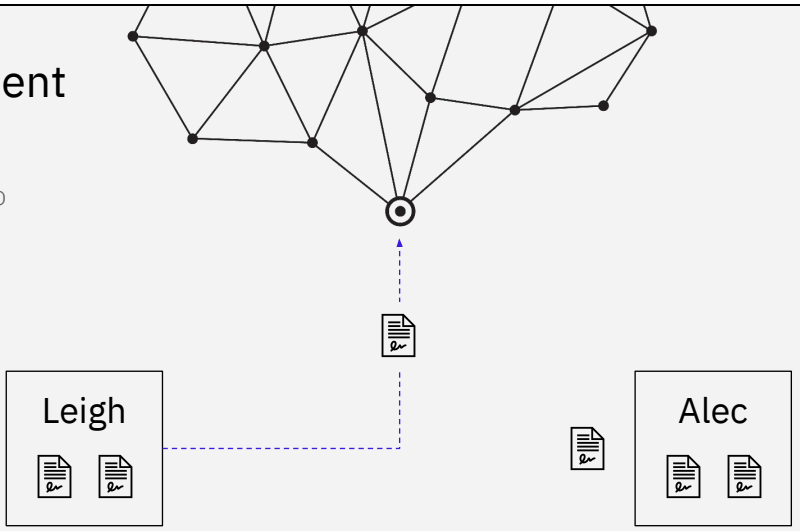
Meridian



The last step in the open process is submitting the open transaction to the network, which is what Leigh did in the demo.

Opening a Payment Channel

Submit Open Transaction to the Network



Meridian

By submitting the open transaction to the network, Leigh effectively opened the channel.

<DEMO

So that's how the open works.

But looking back at the dashboard, we can see neither account has a balance.

So, Leigh will be depositing $10,000 of a US dollar asset into the accounts.

DEMO: deposit 10000
DEMO: deposit 10000 to alec

This a test network, so this isn't real US dollar assets we're using, but it is an issued asset like any other asset you can issue or hold on the Stellar network.

We can see on the left pie chart that the account balances on the Stellar network have updated showing our contributions.

The right pie chart shows our balances of the channel off-network.

To briefly talk about deposits, Starlight channels allow both participants to make deposits to the channel at anytime, without involving the other participant.

They do this with a regular payment to their respective accounts.

No new agreements need signing. And the existing agreements stand.

Any amount I deposit will automatically belong to me when the channel closes, and any amount Leigh deposits will automatically belong to him.

So, it's really easy for me, or anyone really, to top up an existing channel.

And I can use a Stellar anchor, an exchange, or a wallet, to send funds. This makes Starlight channels as simple as a regular Stellar account to integrate with existing payment rails.

Ok, now let's do a payment using the Starlight channel. I'm going to send Leigh $4,000 of the US dollar asset I deposited.

DEMO: pay 4000

We can see on the dashboard that the payment immediately changed the balance of the channel. The payment channel operates as a net settlement mechanism. When Alec paid me $4,000, we updated the channel to pay me $4,000 when it closes.

I'll pay Alec $6,000 of the US dollar asset. Let's watch the dashboard to see what happens.

DEMO: pay 6000

We see the balance swing back the other way. If we were to close the channel now, I would pay Alec $2,000 to settle the account.

The dashboard confirms this. The orange area of the graph has gotten larger, because Alec owns more of the final balance of the channel.

He owns the $10,000 he contributed, plus the $2,000 I owe him after our two payments.

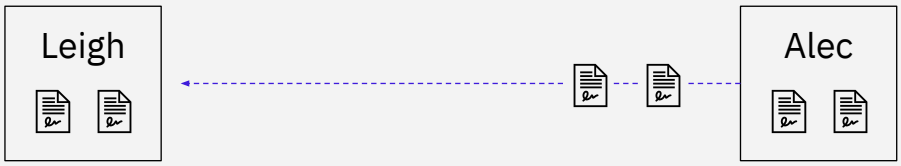
SLIDES>

TRANSITION: Let's have a look at...what's happened in the background when we made the payment.

Making a Payment

Proposes a New Close Agreement

Meridian



Starlight Protocol

25

TRANSITION: Let's have a look at...what's happened in the background when we made the payment.

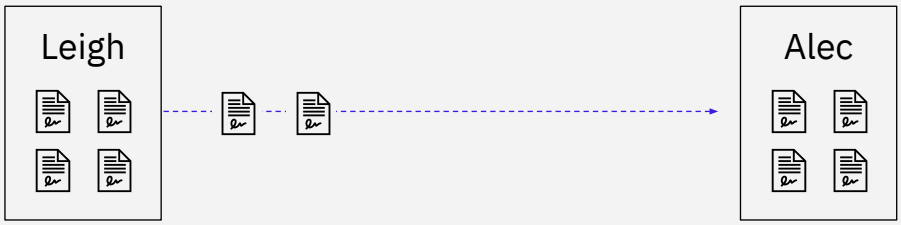
When Alec paid me $4,000, he proposed a new close agreement that superseded the agreement we signed when opening the channel.

TRANSITION: When I received the new agreement, I checked that it increased my balance, and then...signed and returned my signatures to Alec.

Making a Payment

Confirms the New Agreement

Meridian



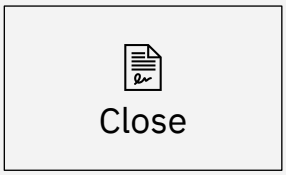
TRANSITION: When I received the new agreement, I checked that it increased my balance, and then...signed and returned my signatures to Alec.

We're exchanging the signatures for these transactions in a single message, which can be unsafe to do in some multi-transaction protocols. We'll come back to this soon and explain why it is safe to do this with Starlight.

TRANSITION: When we sign a close agreement, we're signing two transactions, the...declaration, and the close.

Making a Payment

A Close Agreement Is Two Transactions



Meridian

TRANSITION: When we sign a close agreement, we're signing two transactions, the...declaration, and the close.

These transactions go in pairs.

Together they're used to close the channel.

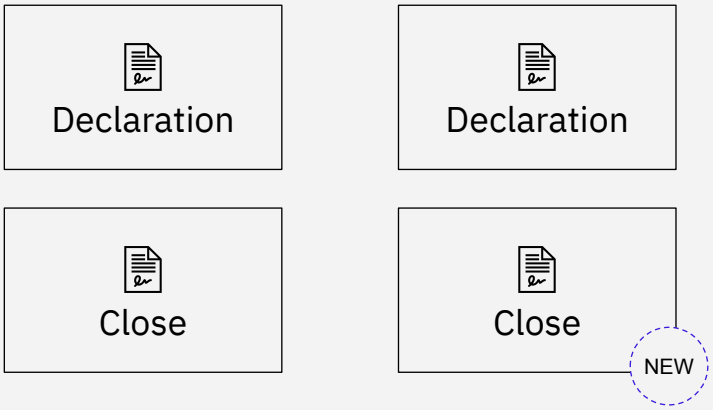
They contain what we're agreeing will be the final state of the channel.

But these transactions probably won't be submitted to the network.

TRANSITION: That's because everytime we send a payment on the channel, we agree to a new...pair of transactions.

Making a Payment

A Close Agreement Is Two Transactions



Starlight Protocol

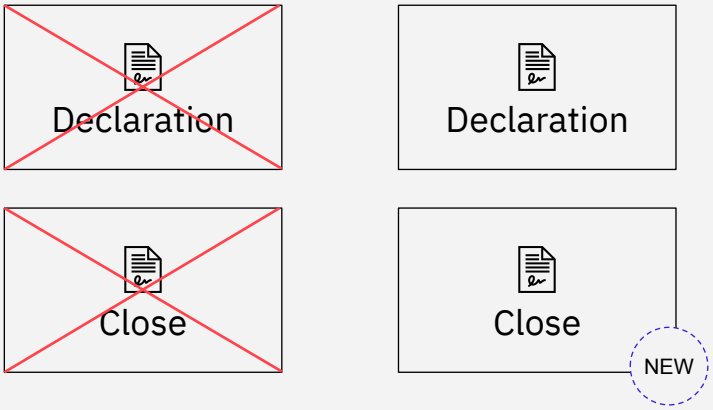
28

TRANSITION: That's because everytime we send a payment on the channel, we be agree to a new...pair of transactions.

TRANSITION: And the new pair of transactions...supersede the previous.

Making a Payment

A Close Agreement Is Two Transactions



Starlight Protocol

29

TRANSITION: And the new pair of transactions...supersede the previous.

TRANSITION: Let's get an understanding of what these transaction do, starting with the...declaration.

The Declaration Transaction

Understanding the Role of the Declaration

Meridian

TRANSITION: Let's get an understanding of what these transaction do, starting with the...declaration.

The declaration is responsible for announcing on-chain that the close process is beginning.

It does that by placing the channel in a state where the close transaction is allowed to execute.

And not just any close transaction, but specifically the close that was paired with the declaration.

It does this by using a very simple feature that's exists on the Stellar network today, as well as an extension to that feature that we're proposing in CAP-21.

TRANSITION: That feature is...sequence numbers.

The Declaration Transaction

Using and Extending Sequence Numbers

Meridian

Sequence Numbers

- Feature of the Stellar Network today
- Provides an order for how transactions must be executed on the network
- Stored in transactions
- Increases by one for every transaction
- Extension proposed in CAP-21

TRANSITION: That feature is...sequence numbers.

An account's transactions have sequence numbers that provide an order to how those transactions must be executed on the network.

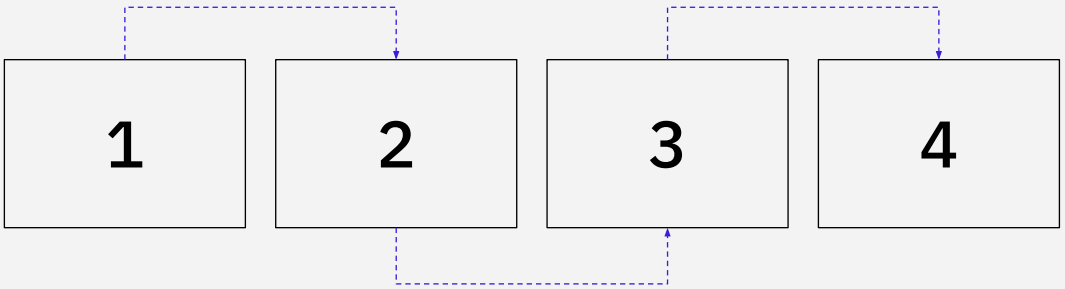
They're a simple number stored in the transaction that must increase by one for every transaction.

TRANSITION: For example...

Sequence Numbers

Traditional Sequence Numbers Require a Series of Transactions to Execute in Order

Meridian



TRANSITION: For example…

If we've created a series of four transactions, 1, 2, 3, and 4.

The transactions must execute in that order.

1 must execute before 2.
2 must execute before 3, and so on.

That's how sequence numbers behave on the Stellar network today.

Now, we're proposing an extension to their capabilities in CAP-21.

TRANSITION: What CAP-21 allows us to do, is to say that a transaction, such as transaction 4, is valid as long as...transaction 1 has executed.

Sequence Numbers

CAP-21 Allows for Sequence Numbers to Be Skipped

Meridian

TRANSITION: What CAP-21 allows us to do, is to say that a transaction, such as transaction 4, is valid as long as...transaction 1 has executed.

We no longer need every transaction between 1 and 4 to execute.

Transactions 2 and 3 can, but they don't have to.

If they are going to be executed, they have to before transaction 4. Because once transaction 4 is confirmed, 2 and 3 are no longer valid.

TRANSITION: Using our transactions so far as an example, Alec and I signed three transactions at the beginning, the...open,...declaration,..and close.

Sequence Numbers

Opening a Payment Channel Created Three Transactions

Meridian

TRANSITION: For example, in the transactions that Alec and I have performed we both initially signed three transactions, the...open,

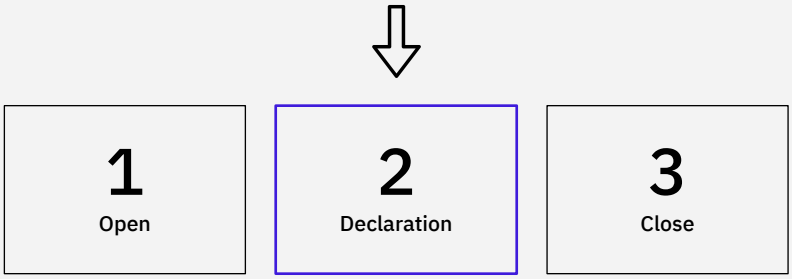
...declaration,

..and close.

Sequence Numbers

Opening a Payment Channel Created Three Transactions

Meridian



...declaration,

..and close.

Sequence Numbers

Opening a Payment Channel Created Three Transactions

Meridian

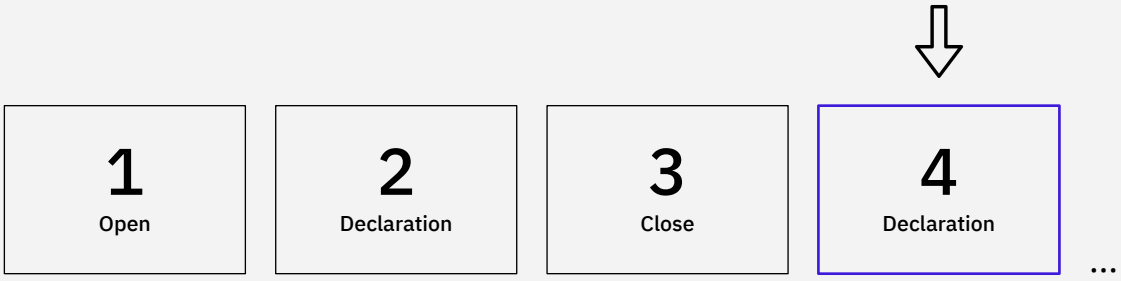
..and close.

TRANSITION: The first payment that Alec sent me, created a new close agreement that occupied sequence numbers,...4 and 5.

Sequence Numbers

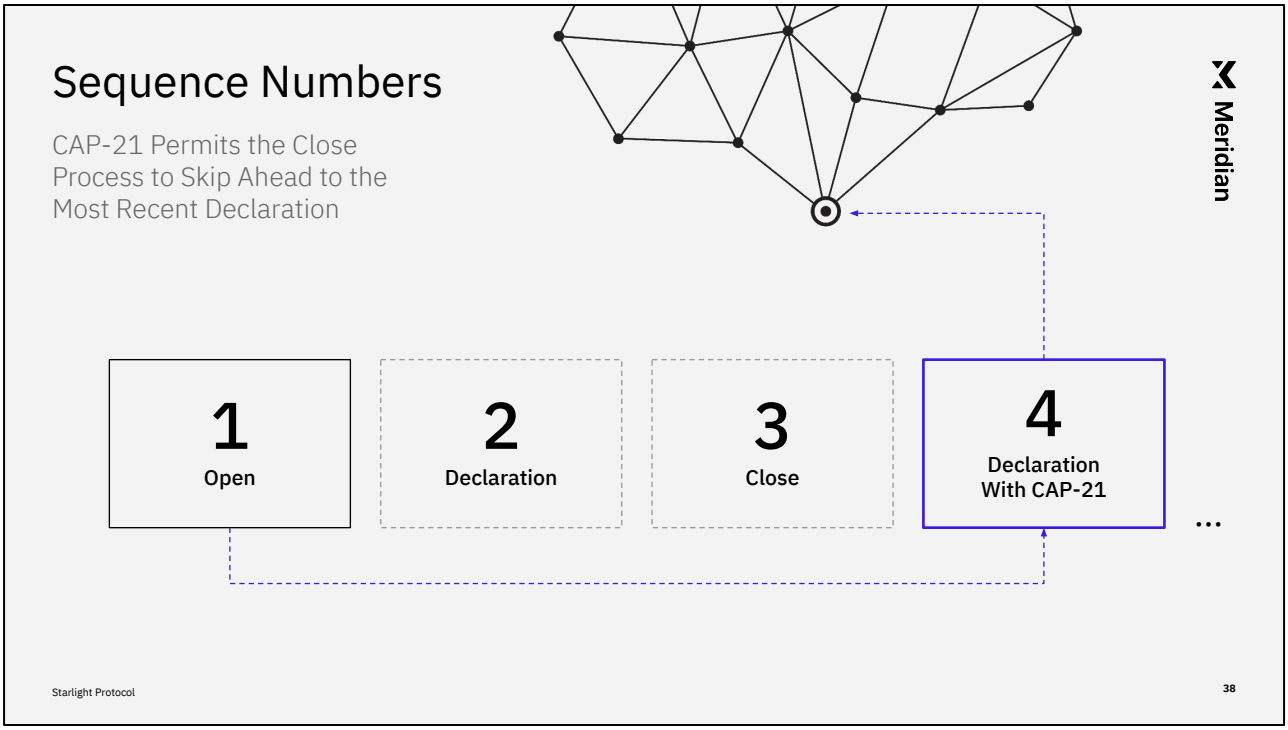
Making a Payment Creates a New Declaration and Close

Meridian



TRANSITION: The first payment that Alec sent me, creates a new close agreement that occupies sequence numbers,...4 and 5.

TRANSITION: If we'd decided to close the channel after that payment, we could simply submit...transaction 4, declaring the start of the close process.



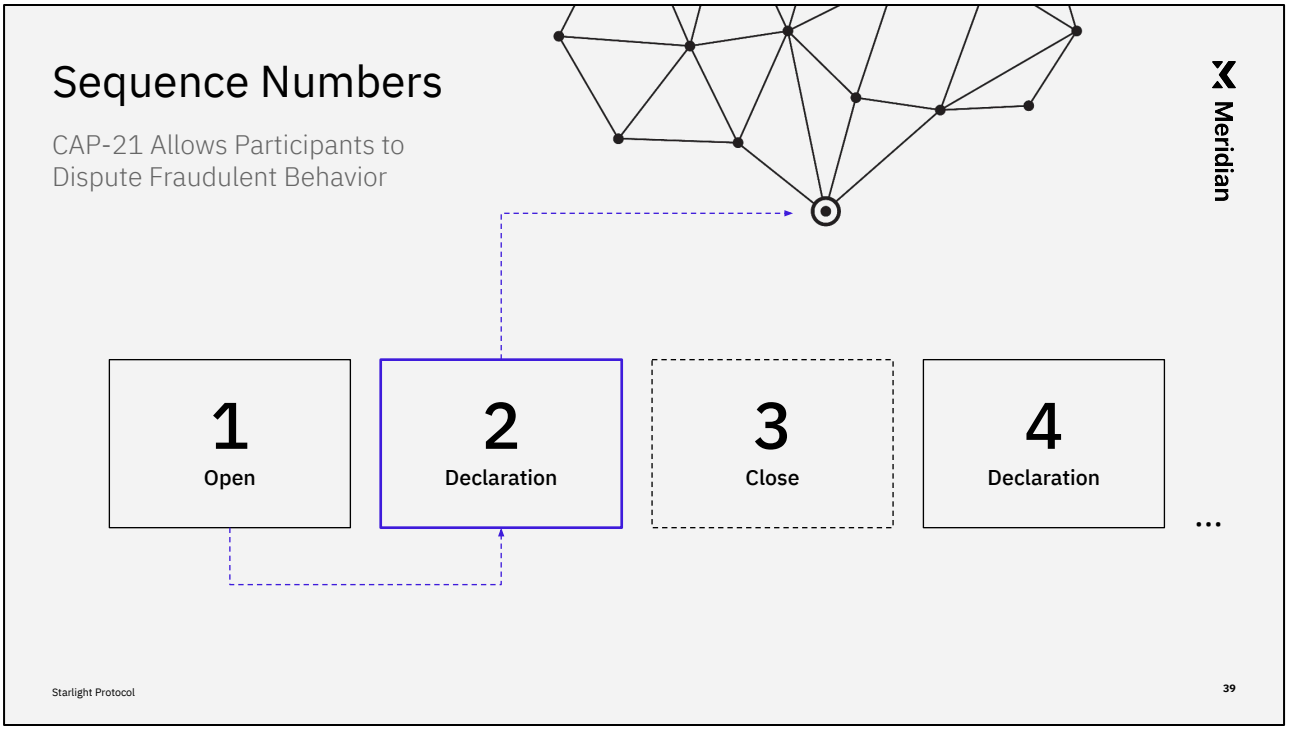
TRANSITION: If we'd decided to close the channel after that payment, we could simply submit...transaction 4, declaring the start of the close process.

Submitting transaction 4 puts the channel into the state where the next transaction, transaction 5, the close transaction, can execute.

PAUSE

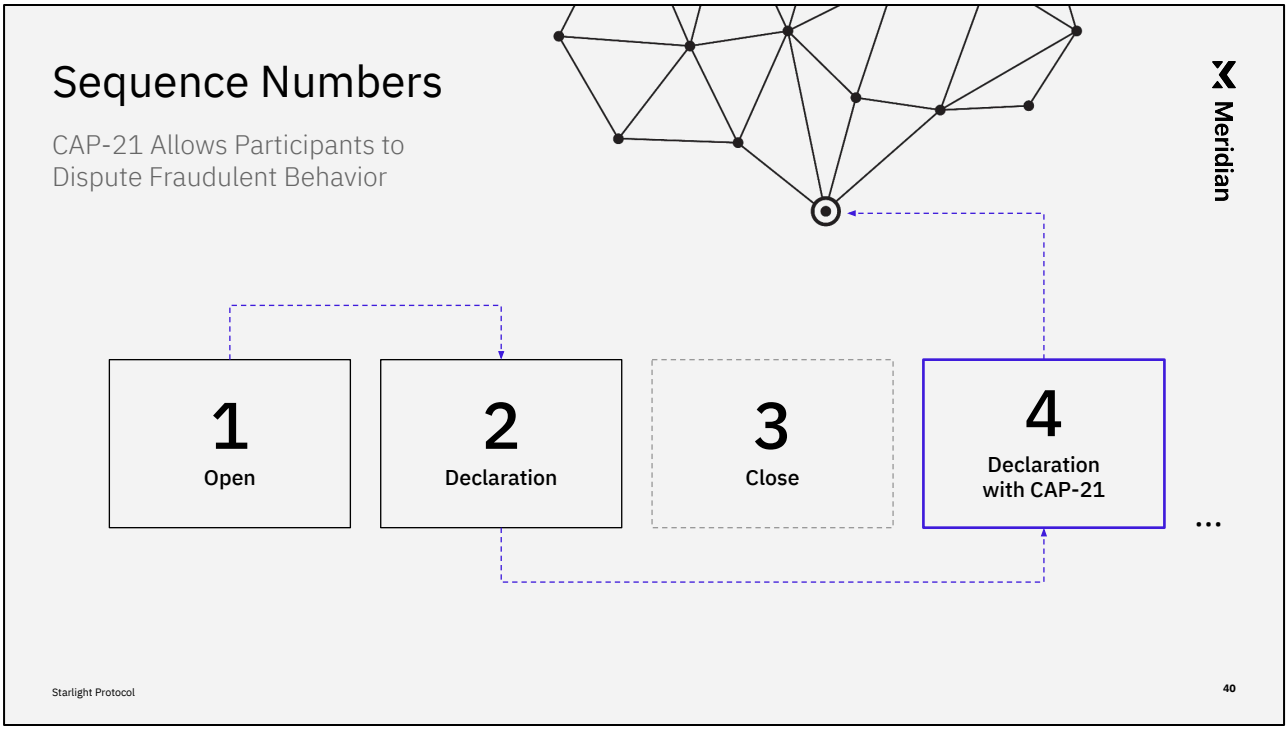
This capability to skip ahead to a more recent declaration is also how a close can be disputed.

TRANSITION: If in our scenario Alec regretted the payment that's captured in transactions 4 and 5 he might try to start the close process with...transaction 2.



TRANSITION: If in our scenario Alec regretted the payment that's in capture in transactions 4 and 5 he might try to start the close process with...transaction 2.

TRANSITION: If I saw that happen, I could dispute the close by submitting...transaction 4.

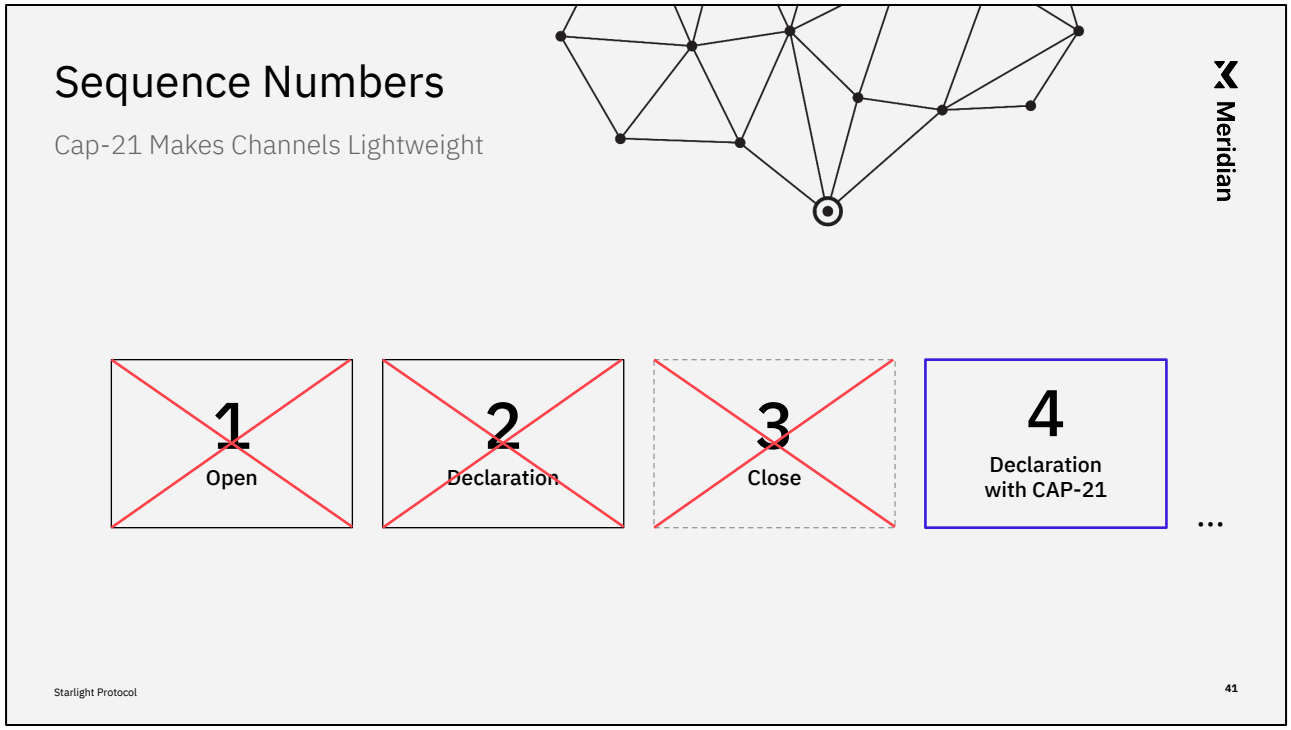


TRANSITION: The other participant can still skip ahead to...transaction 4.

Remember that once transaction 4 is submitted, lower number transactions cannot execute anymore. So this prevents the close from completing with transaction 3 which contains the old state.

Because we can always dispute an old transaction with the latest transaction, there's no need for us to keep old transactions.

And we can delete them.



Participants in a Starlight channel only need to store a tiny amount of data to maintain the safety of the channel, even if the channel is long lived, and even if they do millions of payments.

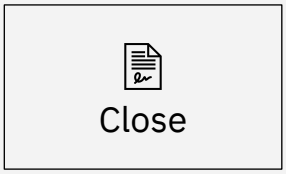
This makes Starlight channels very easy to scale. They don't add a data storage burden to your infrastructure.

Let's take a look at the second transaction in the close agreement.

The Close Transaction

Understanding the Role of the Close

Meridian



The second transaction is the close transaction.

The close transaction is responsible for finalizing the close of the channel.

The Close Transaction

Contains a few operations

Meridian

Operations

- Payment – Distributes the net-settlement
- 2x Set Options – Modifies the signers of the accounts

To do this it contains a few operations. The first is a payment operation.

This payment operation is a net settlement resulting from all the payments that occurred in the channel up to that point in time.

So far we've sent two payments. I paid Leigh $4,000, and Leigh paid me $6,000. If we closed the channel now, the net settlement would be $2,000 owed to me, from Leigh.

In addition to the payment, the transaction also contains two operations that modify the signers of Leigh and my Stellar Accounts. Those operations give Leigh and I full control of our accounts again.

So, together these three operations close the channel.

There's one more attribute of the close transaction that's really important...

The Close Transaction

Introducing a Transaction Time Delay

Meridian

Relative Time Lock

- New capability proposed in CAP-21
- Introduces a configurable delay between two transactions
- Delay is relative to when first transaction executes

And that attribute is a relative time lock.

Relative time locks are a new capability that CAP-21 brings to the Stellar network.

Relative Time Lock

How It Works

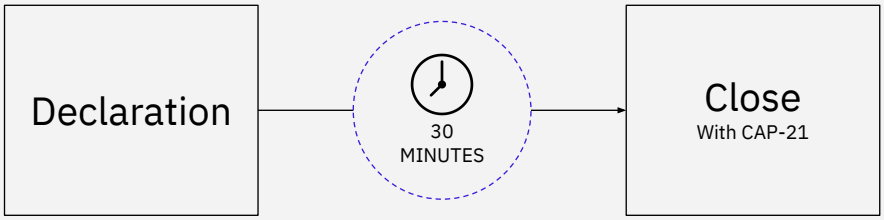
Meridian

Normally transactions from the same account can execute immediately one after another.

Relative Time Lock

How It Works

Meridian



Relative time locks introduce a delay that must occur between two transactions.

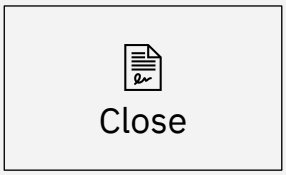
With CAP-21 we configure the close transaction so that there must be a delay between itself and the declaration.

If I submit a declaration, let's say for example an older one like in Leigh's example, this delay is what gives Leigh that time to dispute the closing of the channel.

Making a Payment

A Close Agreement Is Two Transactions

Meridian



That's how the declaration and close transactions come together to close the channel.

We mentioned earlier that one of the challenges with protocols that require participants to sign multiple transactions, is that it can be unsafe to share signatures for both transactions in a single message.

But as we saw, participants in Starlight channels do share all signatures at the same time. The reason we do that is for efficiency. By eliminating network messages we maximize the number of transactions we can perform in a second.

Solving this problem is where a new capability proposed in CAP-40 comes in.

TRANSITION: That capability is...atomic signature disclosure.

Atomic Signature Disclosure

Meridian

A capability in CAP-40 to safely share signatures for multiple transactions in a single message.

TRANSITION: That capability is...atomic signature disclosure.

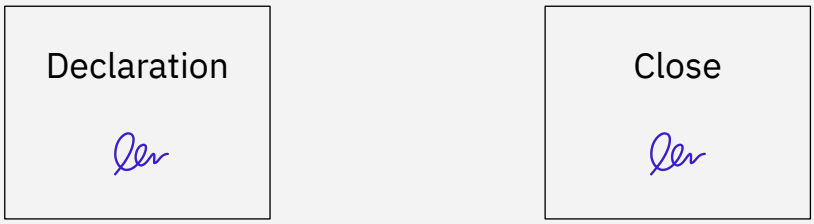
Let's walk through an example to understand why the problem exists, and how atomic signature disclosure solves it.

TRANSITION: Let's assume..Alec signs two transactions assuming that if one executes the other must then be valid.

Atomic Signature Disclosure

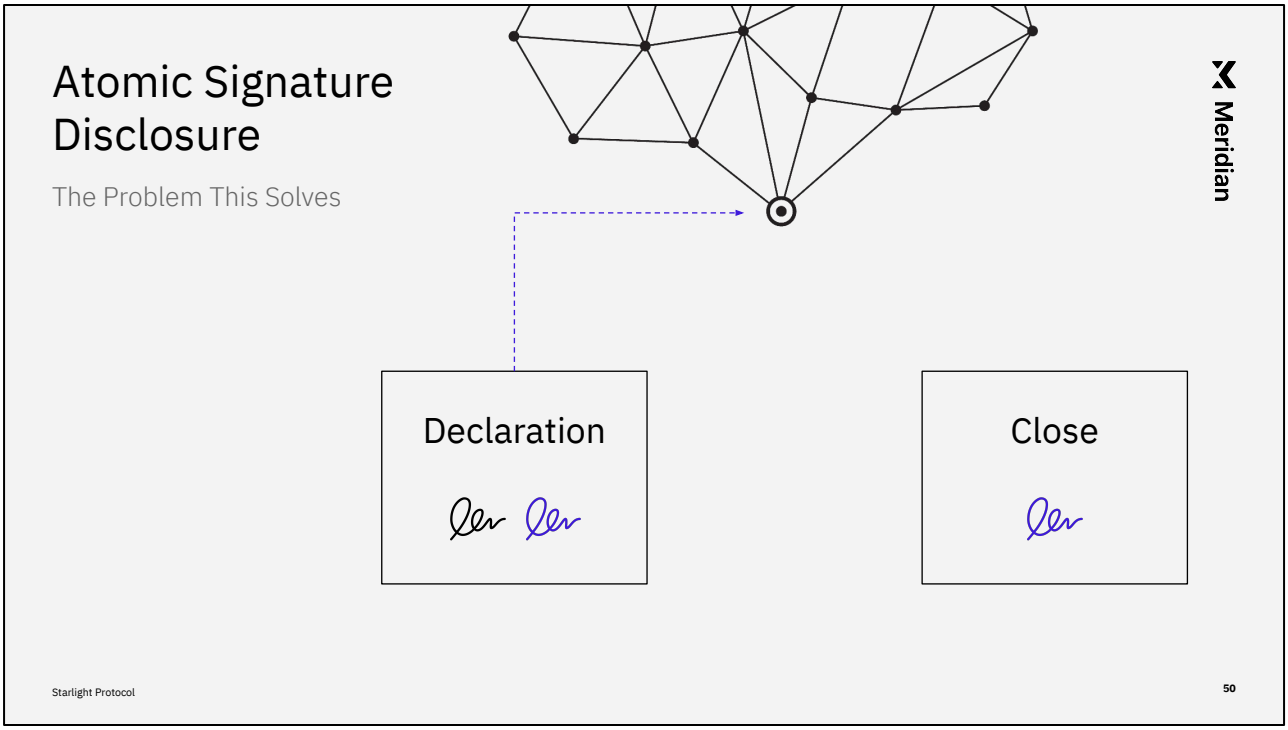
The Problem This Solves

Meridian



TRANSITION: Let's assume..Alec signs two transactions assuming that if one executes the other must then be valid.

TRANSITION: And say I...sign and submit to the network only one of the transactions.



TRANSITION: And say I...sign and submit to the network only one of the transactions.

Alec would have no way of submitting the second transaction, because he doesn't have all the signatures for it.

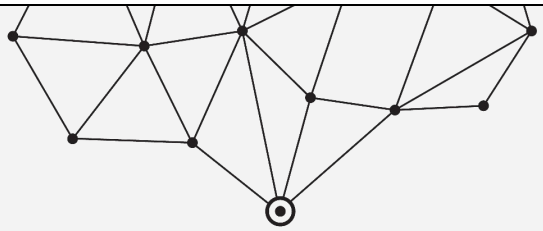
CAP-40 addresses this challenge by introducing a new type of signer to the Stellar network.

That signer, is the signature of another transaction.

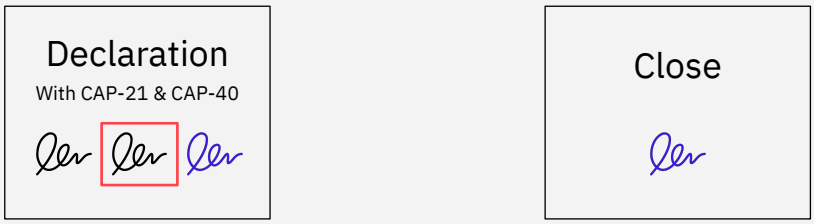
TRANSITION: Atomic signature disclosure then uses CAP-21 to embed that new signer and its signature inside the...first transaction.

Atomic Signature Disclosure

The Solution: CAP-40 and CAP-21 Require the Signature of the Close Within the Declaration



Meridian



TRANSITION: Atomic signature disclosure then uses CAP-21 to embed that new signer and its signature inside the...first transaction.

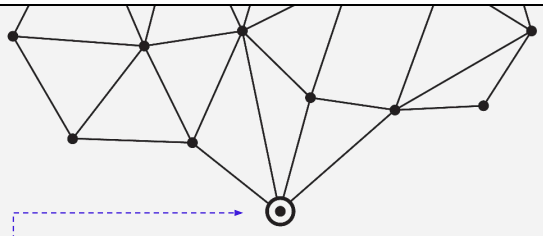
Starlight channels use CAP-40 and CAP-21 to require participants to disclose the signature of the close within the declaration.

This means that the first transaction isn't valid unless the signature for the second is also disclosed.

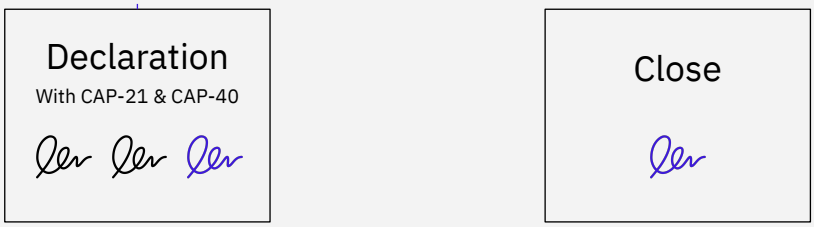
TRANSITION: If we rerun that same scenario using the process defined in CAP-40...

Atomic Signature Disclosure

The Solution: CAP-40 and CAP-21 Require the Signature of the Close Within the Declaration



Meridian



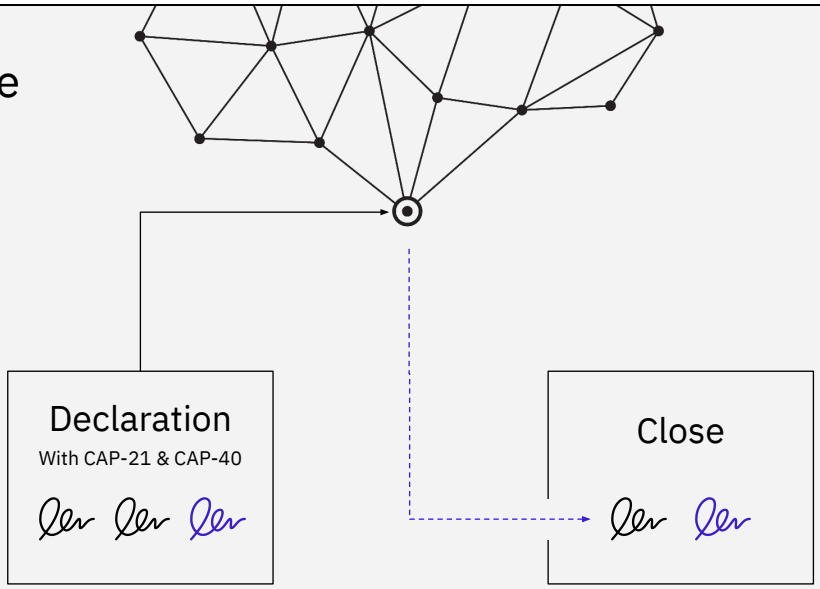
TRANSITION: If we rerun that same scenario using CAP-40...

If I was to completely authorize the declaration, and submit that transaction to the network.

TRANSITION: Alec would see the transaction on the network and be able to...copy the embedded signature and attach it to the close transaction.

Atomic Signature Disclosure

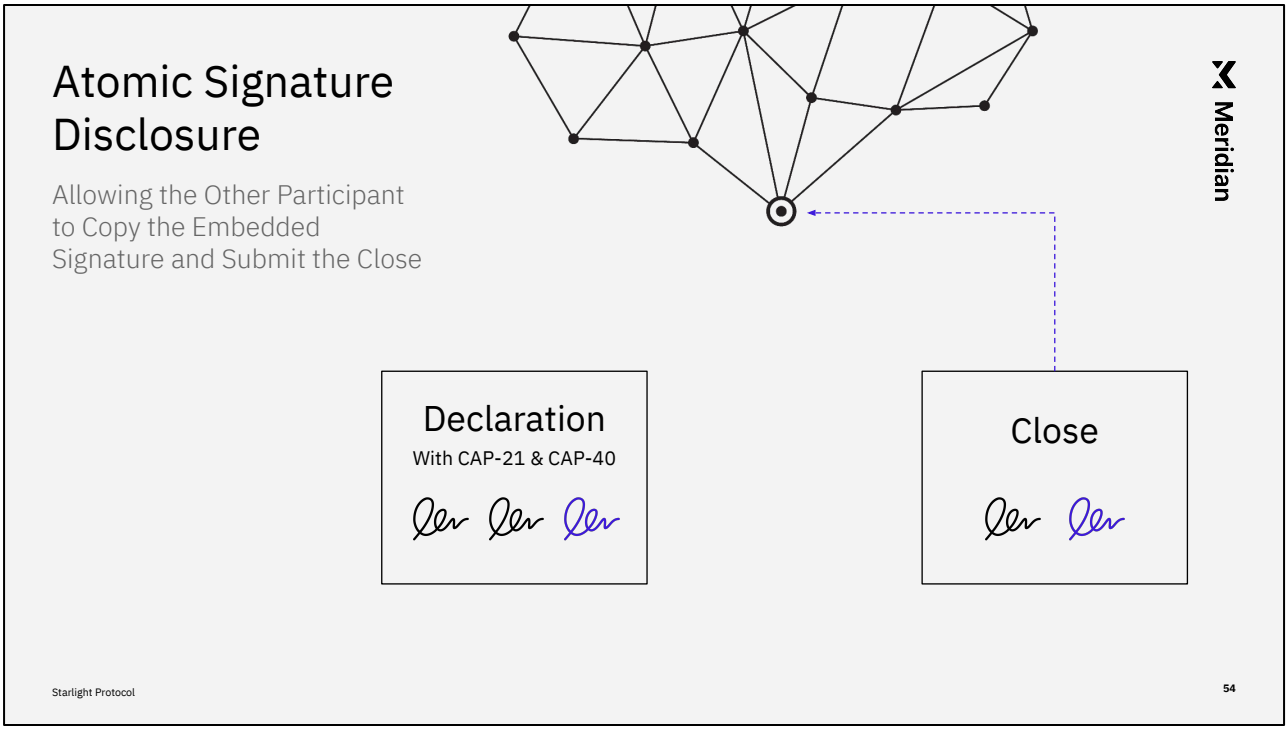
The Solution: CAP-40 and CAP-21 Require the Signature of the Close Within the Declaration



Meridian

TRANSITION: Alec would see the transaction on the network and be able to...copy the embedded signature and attach it to the close.

TRANSITION: That would allow Alec to...submit the close transaction.



TRANSITION: That would allow Alec to...submit the close transaction.

And so, atomic signature disclosure makes it safe to share signatures in a single message.

Because it requires the disclosure of all signatures before any transaction is valid.

PAUSE

You will have noticed that we've mentioned CAP-21 a few times.

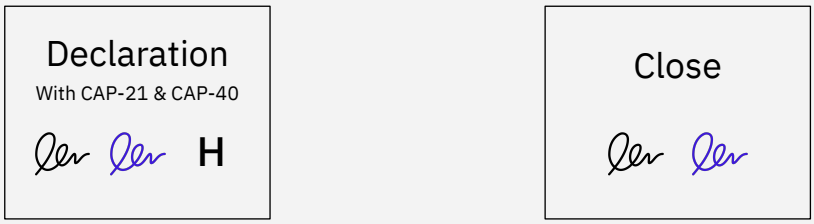
That's because the proposal is full of new features that add new capabilities to the Stellar network.

TRANSITION: One that I want to touch on, because it is relevant to payment channels, is the ability to add...hash signers directly to transactions as a precondition.

Adding Hash Signers Directly to Transactions

Another CAP-21 Capability: Stateless HTLCs

Meridian



Starlight Protocol

55

TRANSITION: One that I want to touch on, because it is relevant to payment channels, is the ability to add...hash signers directly to transactions as a precondition.

This will give us the ability to create the hash component of a HTLC in a stateless way.

The hash will live only within a transaction off-network.

We think this will be useful for anyone who implements multi-hop trustless payment channels in the future.

Let’s jump back to our demo.

<DEMO

<DEMO

So we did some one off payments and we were able to see what effect they had on the balance.

Let's generate a larger number of payments.

I'm going generate a thousand payments. They'll be sent to Alec one at a time. Each payment will wait for previous payment to complete before sending.

Because we're doing these payments serially the speed at which we do the payments will be limited by our network latency.

The reason for this is each payment is a message that must travel from me to Alec, and from Alec back to me.

Our network latency is around 30 milliseconds, and so we can expect to see around 33 transactions per second.

I'm sending the payments now.

DEMO: payx 1 1000 1

Watching the dashboard we can see we're doing 30 TPS.

PAUSE

Not all use cases require that the result of every payment be confirmed before another payment begins.

In Starlight channels we've also been experimenting with buffering payments.

SLIDES>

TRANSITION: Buffering is the process where we...collect payments while we wait for the opportunity to propose the next agreement.

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian

TRANSITION: Buffering is the process where we...collect payments while we wait for the opportunity to propose the next agreement.

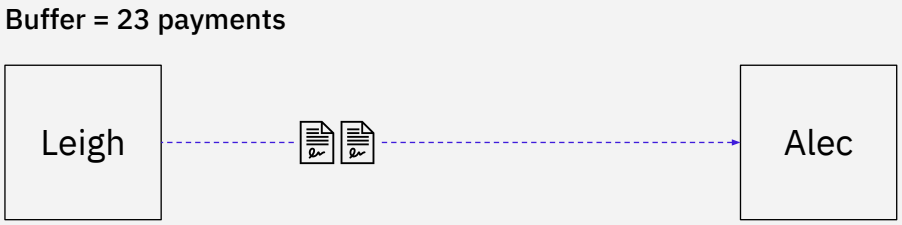
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

1 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



Starlight Protocol

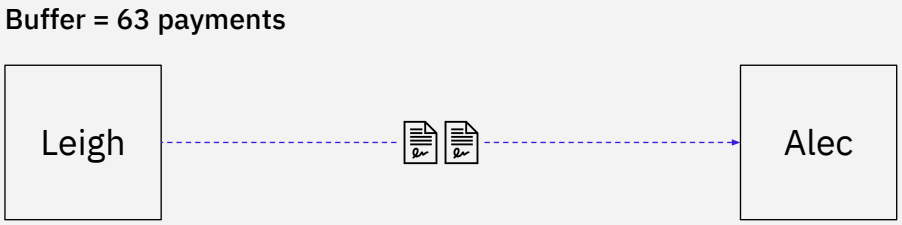
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

2 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



Starlight Protocol

59

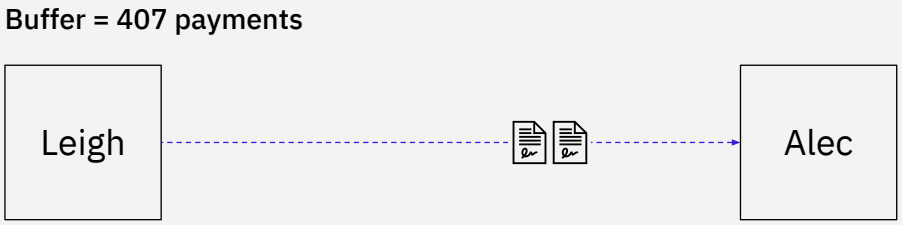
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

3 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



Starlight Protocol

60

While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

4 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian

Starlight Protocol

61

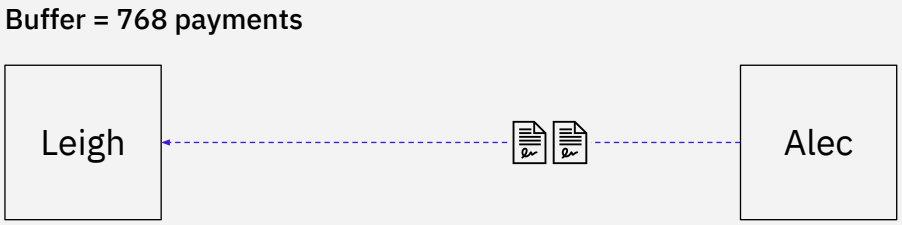
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

5 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



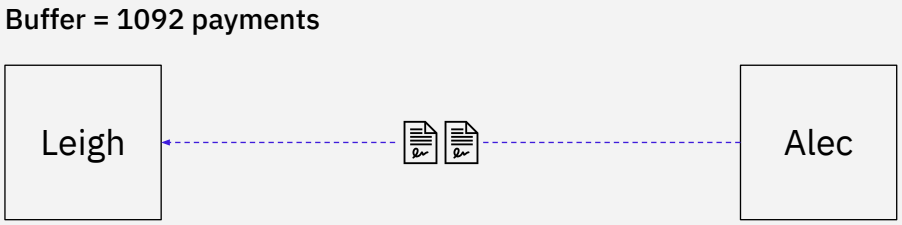
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

6 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



Starlight Protocol

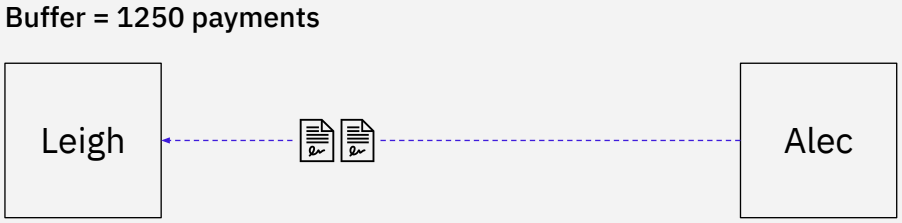
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

7 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



Starlight Protocol

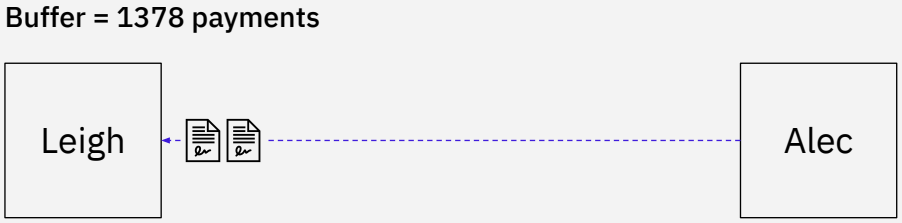
While we're waiting the 30 milliseconds for our agreement to be authorized, we collect payments into a buffer.

8 / 9

Buffering Payments

Collect Payments While an Agreement Is Being Authorized

Meridian



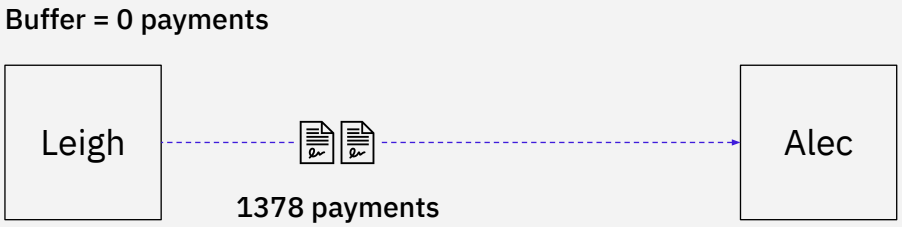
TRANSITION: Once the agreement is authorized, we can...immediately propose the next agreement.

9 / 9

Buffering Payments

Propose a New Agreement With the Payments Previously Buffered

Meridian



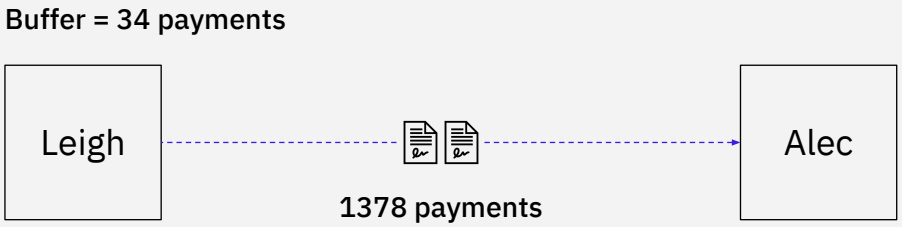
TRANSITION: Once the last transaction set is authorized, we can...immediately propose the next agreement.

1 / 3

Buffering Payments

Propose a New Agreement With the Payments Previously Buffered

Meridian



Starlight Protocol

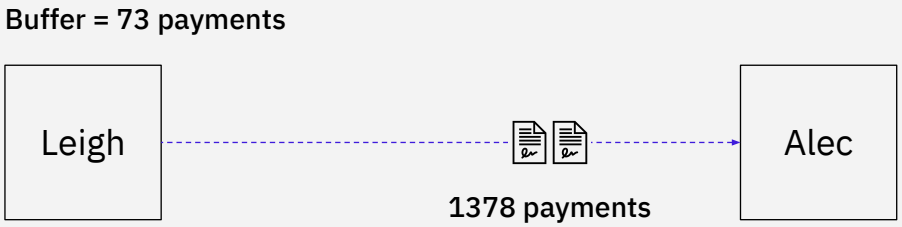
That next agreement will contain all of the payments that have been buffered during that 30 millisecond wait.

2 / 3

Buffering Payments

Each Subsequent Agreement Will Contain All
Payments Buffered During the 30 Millisecond Wait

Meridian



Starlight Protocol

68

We send the details of each individual payment, but we deliver that buffer in a single agreement.

A payment channel is fast, so we can deliver many buffers per second.

We're just limited by how much bandwidth we have to transmit the buffer over the network.

3 / 3

<DEMO

<DEMO

I'm going to send Alec 100k payments. I'm setting our maximum buffer size to 10000. Our payment channel will propose a new agreement as soon as it can, but while it's waiting for that opportunity it'll allow up to 10000 payments to collect in the buffer. It'll then send those 10000 payments at the next opportunity it gets.

I'm starting the payments now.

DEMO: payx 0.01 100000 10000

Watching the dashboard we can see the TPS is much higher, because we're using the bandwidth we have available to us along with the fast finality of the payment channel, to send a large number of payments over the wire in a few milliseconds.

Let's try another.

I'm going to send Alec 10 million payments with larger buffer sizes, up to 95k.

This might seem a bit unreasonable, but at this size our buffers are small when compressed. We should see the buffers be around a couple hundred kilobytes. So as long as we have the bandwidth to move those buffers across the network quickly, we're still gonna get fast finality.

DEMO: payx 0.001 10000000 95000

You might be wondering how buffering over payment channels like Starlight, is different to batching payments, which is a really common pattern in slow traditional payments systems. And the answer is that payment channels aren't slow, they're fast. This difference means buffering payments doesn't have to compromise on time to finality. The payment sender still gets a fast confirmation in milliseconds. But instead of getting fast confirmation of one payment, they get fast confirmation of an agreement that is made up of many payments that occurred in the last few milliseconds. This is really the same principle as blocks in blockchains.

PAUSE

Let's move on to closing the channel.

SLIDES>

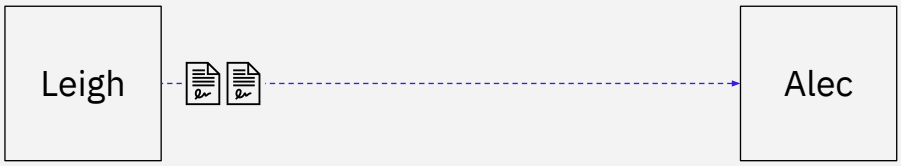
Closing the channel means submitting the declaration, waiting the delay period, then submitting the close.

But when we are both in agreement about closing the channel, waiting the delay time between the declaration and close is unnecessary.

Closing the Payment Channel

Both Parties Sign to Authorize the Agreement

Meridian

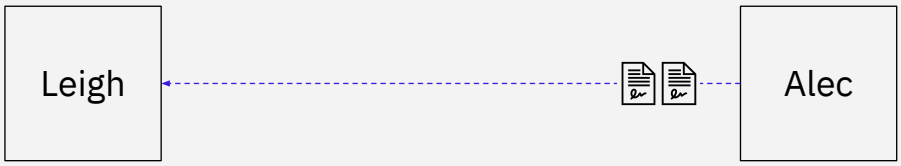


When Leigh kicks off the close process, his application will modify the last agreement removing the delay, resign the transactions, and send the signatures to me.

Closing the Payment Channel

Both Parties Sign to Authorize the Agreement

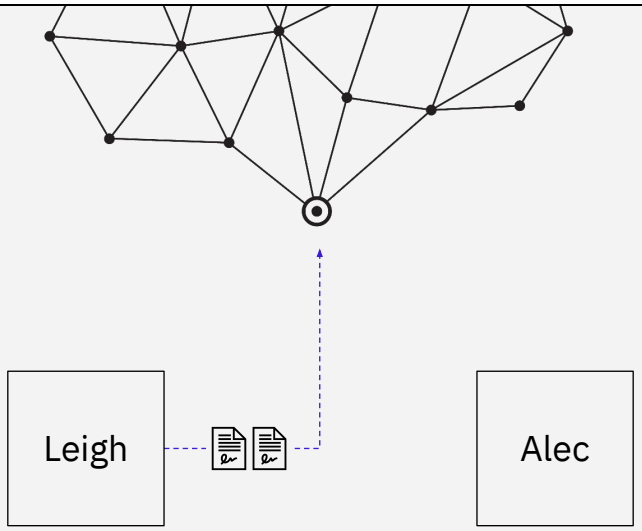
Meridian



Then my application will verify the signatures, make sure it's the same balance as the last agreement, sign it, and send my signatures back to Leigh.

Closing the Payment Channel

Submit to the Network



Meridian

At this point the close agreement, which has both Leigh and my signatures, can be submitted to the network.

<DEMO

Let’s now close our channel in the demo.

Starlight Demo

A Summary

Meridian

1

We Opened and Closed a Channel

- Performed millions of payments
- Closed the channel quickly because the Stellar network is fast

2

We Did It Efficiently

- Stored only the most recent agreement leveraging CAP-21
- Used single message in either direction to exchange signatures using CAP-40

3

We Can Recover

- Able to dispute fraudulent behavior using CAP-21
- Able to recover signatures not shared using CAP-40

<DEMO

I'm running the close command.

DEMO: declareclose

On the right we can see the balance of the two Stellar accounts on network have updated to match the balance of the payment channel off-network.

And the signers of the two accounts have changed. Alec and I each have full control over our own Stellar accounts.

Fantastic. The payment channel worked.

SLIDES>

To reflect on what we've done.

We opened a channel.
Performed millions of payments.
And were able to close the channel quickly.

And we were able to do that efficiently:

We only needed to store the most recent agreement.
Without needing to revoke previous agreements.
Because of CAP-21.
We were able to communicate efficiently with a single message in either direction.
Because we could use CAP-40's atomic signature disclosure.

And we saw we were able to recover in some situations, like where:
We're able to dispute fraudulent behavior using CAP-21.
And able to recover signatures not shared using CAP-40.

SLIDES>

So, what's next for Starlight and payment channels on Stellar?

Check Out Starlight

Public and Open Source

Meridian

Open Source

github.com/stellar/starlight

- Docker image runs a network with prototypes of CAP-21 and CAP-40
- Starlight Go SDK
- Example code

Right now we have a public repo on Github that hosts Starlight for anyone to look at. In that repo you'll find:

our CAP-21 and CAP-40 proposals.

As well as a Docker image for running your own private network with these proposals implemented, which is what we used for our demo today.

And our prototype SDK is also there, written in Go. And there is example code using this SDK, both with and without buffering.

<PAUSE>

In terms of what's next.

We have ideas for how we can improve and build upon what we've already designed.

And we're actively pursuing getting CAP-21 and CAP-40 accepted into an upcoming protocol release.

If payment channels on Stellar interests you at all we invite you to get involved.

You can talk to us on our GitHub repo's discussion board about anything related to Starlight.

And please feel free to experiment with our prototype SDK yourself!

Thank you!