ISSN: 2230-9926

Available online at http://www.journalijdr.com

IJDR

International Journal of Development Research
Vol. 14, Issue, 04, pp. 65512-65516, April, 2024
https://doi.org/10.37118/ijdr.28175.04.2024

RESEARCH ARTICLE OPEN ACCESS

STREAM CONTROL TRANSMISSION PROTOCOL

***Barigat Shaik Sadiya, Suraparaju Harshini and Prof M.P Vani**

Computer Networks, Course-Integrated Mtech Software Engineering, School-Score, Vellore Institute of Technology, Vellore, Tamilnadu, India

ARTICLE INFO

Article History:
Received 17th January, 2024
Received in revised form
26th February, 2024
Accepted 05th March, 2024
Published online 30th April, 2024

Key Words:
Stream Control Transmission Protocol, Message-oriented communication, Multi-homing capability.

Corresponding author: Barigat Shaik Sadiya

ABSTRACT

Stream Control Transmission Protocol (SCTP) is presented as a robust option compared to traditional transport layer protocols like TCP and UDP. It offers various advanced features to meet modern communication needs. This paper provides an overview of SCTP, highlighting its unique attributes and benefits in ensuring reliable, message-oriented communication between hosts. SCTP ensures data integrity through acknowledgment mechanisms and retransmission strategies, even during network disruptions. Its messageoriented nature preserves data unit boundaries, facilitating efficient communication between applications. One notable feature of SCTP is its support for multi-streaming, allowing multiple streams of data to travel independently through a single connection. This enables applications to manage different types of data traffic within the same connection, optimizing resource usage. Additionally, SCTP's multi-homing capability provides redundancy and fault tolerance by supporting multiple network interfaces. Its path management capabilities dynamically monitor network conditions, enhancing fault tolerance, load balancing, and overall performance in diverse network environments. Moreover, SCTP offers flexible ordered and unordered data delivery options, meeting various application requirements. OBJECTIVE: This paper helps readers to understand the Stream Control Transmission Protocol (SCTP). We will include its basic principles and how it actually works. By dissecting SCTP's structure, features, and what it can do, we hope to make it easier for people to grasp how SCTP functions and why it's better than traditional transport layer protocols. We'll look at things like how reliable it is, its focus on handling messages, its ability to handle multiple streams and connections (multi-streaming and multi-homing), and its capability to manage paths adaptively. Ultimately, our goal is to give readers the knowledge they need to see how SCTP improves communication reliability and efficiency in modern networking setups.

Citation*: *Barigat Shaik Sadiya, Suraparaju Harshini and Prof M.P Vani, 2024. *"Stream control transmission protocol". International Journal of Development Research*, 14, (04), 65512-65516.

INTRODUCTION

SCTP (Stream Control Transmission Protocol): Invented in 2000.It's reliable like the careful delivery service, making sure everything gets there correctly, but it also aims to be fast like the motorbike messenger. This is particularly useful for things like phone calls, video chats, and online games where both speed and reliability are important. SCTP can handle these situations smoothly, making communication more efficient and less likely to have hiccups. Beyond Speed and Reliability SCTP goes even further, offering some unique features:

Multihoming: Think of having multiple internet connections. SCTP can use any of them, ensuring smooth communication even if one path gets congested. [4]

Multi-streaming: Imagine sending multiple messages simultaneously. SCTP can handle separate streams within a single connection, perfect for video calls with audio, video, and control signals traveling together.

WHY SCTP?

Reliable Delivery: Think of SCTP like a careful postal worker. It makes sure all the pieces of your video call information reach your friend's device in the right order, just like you sent them. This is similar to how regular mail works.

Faster Speeds: Unlike regular mail, SCTP can also be speedy. It avoids some extra checks, allowing information to travel quicker. This is similar to sending a quick text message, but...

Message Boundaries: SCTP knows where each part of your message (like a video frame) begins and ends. This makes it easier for your friend's device to understand the information. Text messages don't have this, so things can get jumbled sometimes.

Benefits of SCTP

Clear Calls and Video Chats: Because SCTP is both reliable and fast, it's perfect for voice calls, video conferencing, and online gaming where smooth communication is essential. Keeps Working During Outages: If your internet connection goes down for a moment, SCTP can automatically switch to another one, keeping your call going without interruption. Secure for Sensitive Data: SCTP can scramble data like a secret code, making it unreadable to anyone who shouldn't see it. This is useful for online banking or other private communication.

Works with Modern Networks: SCTP is compatible with both older and newer internet systems. In short, SCTP is a versatile tool that helps information travel reliably and quickly across the internet. This makes it a valuable asset for many of the communication tools we use today.

Key Features of SCTP

Reliable Delivery: Makes sure all data arrives correctly.

Faster Speeds: Gets information there quicker than some other methods.

Message Boundaries: Knows where each piece of data begins and ends.

Multi-connection: Can use multiple internet connections for reliability.

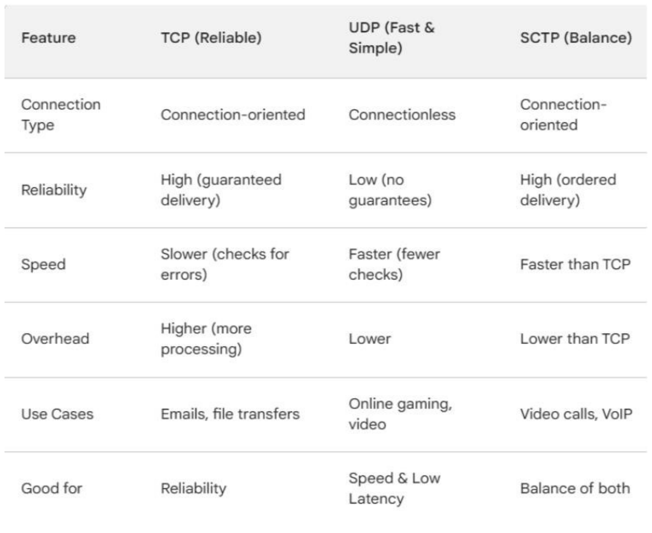
Multi-tasking: Can send different types of data (like video and audio) at once.

Traffic Control: Prevents data overload on the network.

Security Features: Protects data from being tampered with.

Works with Both Networks: Compatible with modern and older internet systems. [4]

Comparison of SCTP with other protocols:



Feature	TCP (Reliable)	UDP (Fast & Simple)	SCTP (Balance)
Connection Type	Connection-oriented	Connectionless	Connection-oriented
Reliability	High (guaranteed delivery)	Low (no guarantees)	High (ordered delivery)
Speed	Slower (checks for errors)	Faster (fewer checks)	Faster than TCP
Overhead	Higher (more processing)	Lower	Lower than TCP
Use Cases	Emails, file transfers	Online gaming, video	Video calls, VoIP
Good for	Reliability	Speed & Low Latency	Balance of both

Packet Structure: Let's see the fundamental structure of an SCTP packet, known as a "chunk": [1][3]

1. **Common Header:** The common header is present at the beginning of each SCTP chunk and contains essential information about the packet.

Key fields include:

- **Source Port Number:** The port number of the sending endpoint.
- ***Destination Port Number:*** The port number of the receiving endpoint.
- ***Verification Tag:*** A 32-bit value used for verification purposes to ensure that the SCTP packet belongs to an association.
- ***Checksum:*** A 32-bit field used for error detection.

2.Chunk Type and Chunk Flags: After the common header, there are two fields: Chunk Type (1 byte) and Chunk Flags (1 byte). These fields specify the type of chunk and provide additional control or information about the chunk.

3.Chunk Length: A 16-bit field indicating the length of the chunk in bytes. This field allows the receiver to accurately determine the size of the incoming chunk and locate the next chunk in the SCTP packet.

4.Chunk-specific Parameters: Following the common header, type, flags, and length fields, there are parameters specific to each type of chunk. These parameters carry various information essential for SCTP's operation. Some common chunk types and their parameters include: -

Data: Carries user message data and sequence numbers.

- **INIT:** Used during association initialization, contains information such as supported features and initial sequence numbers.
- **SACK:** Acknowledges received DATA chunks, includes cumulative acknowledgment and selective acknowledgment information.
- **HEARTBEAT:** Used for detecting the liveliness of a peer, may include heartbeat information.
- **SHUTDOW:** Indicates the intention to terminate an association, may include additional shutdown-related parameters.

5. Padding: Sometimes padding is added to ensure alignment or for other reasons. This padding ensures that the next chunk starts on a 32-bit boundary. It helps maintain the structure of the SCTP packet and facilitates efficient parsing.

6. Chunk Multiplexing: SCTP supports the multiple chunks within a single SCTP packet known as chunk multiplexing, helps reduce packet overhead and improves efficiency by allowing multiple operations to perform in a single packet.

By understanding the structure of SCTP chunks, network engineers and developers can effectively analyse and optimize SCTP-based communication protocols for various applications and network environments.[1]

Association setup and teardown: The setup and teardown of associations in the Stream Control Transmission Protocol (SCTP) involve a multi-step process known as a four-way handshake. This process ensures reliable communication between endpoints and allows for the establishment and termination of SCTP associations. Here are the details of the four-way handshake for establishing and terminating SCTP associations:

Association Setup (Four-Way Handshake)

1. Initialization (INIT): The process starts with the initiator, also known as the client, sending an INIT chunk. This chunk contains important information like the sender's Verification Tag, supported features, initial sequence numbers, and other optional details. The initiator picks an initial sequence number and sends the INIT chunk to the other endpoint.

2. Initialization Acknowledgment (INIT-ACK): When the server receives the INIT chunk, it sends back an INIT-ACK chunk to the initiator. This INIT-ACK chunk confirms receiving the INIT chunk and contains similar information as the INIT chunk. Additionally, the server selects its own initial sequence numbers and includes them in the INIT-ACK chunk.

3.*Cookie Echo (COOKIE-ECHO)* - After getting the INIT-ACK chunk, the initiator sends a COOKIEECHO chunk to the server. This COOKIE-ECHO chunk includes the parameters from the original INIT chunk, along with a new cookie. The cookie acts as a token generated by the initiator to verify the association setup and protect against certain attacks, like SYN flooding.

4.Cookie Acknowledgment (COOKIE-ACK) inSCTP: Imagine two friends setting up a walkietalkie call with a secret code. Here's how SCTP's COOKIE-ACK works in this scenario:

Secret Code Exchange: One friend (initiator) whispers the secret code (COOKIE-ECHO chunk) to the other (responder).

Confirmation: The other friend whispers back "OK" (COOKIE-ACK chunk) to confirm they heard the code correctly and it's valid.

Ready to Talk: Once they both whisper "OK," they can start talking (data transmission) on their walkie-talkies! In SCTP, the "secret code" is a data packet containing a cookie. The COOKIEACK chunk is the responder's way of saying, "I got your secret code and it looks good, let's chat!" Once the initiator receives the COOKIEACK, they know both sides are connected and ready to exchange data.

Association Teardown (Four-Way Handshake)

1. ***Initiation of Shutdown (SHUTDOWN)*** - The association teardown begins when one initiator decides to close the association. The endpoint sends a SHUTDOWN chunk to the peer, indicating its intention to terminate the association gracefully. Imagine you're ending a video call with a friend.
2. ***Here's how SCTP ensures a cleandisconnection Initiating Shutdown (SHUTDOWN chunk):*** You (the initiator) tell your friend it's time to end the call by sending a "goodbye" signal (SHUTDOWN chunk).
3. ***Acknowledging Goodbye (SHUTDOWN-ACKchunk):*** Your friend responds with an "OK, goodbye" (SHUTDOWN-ACK chunk) to confirm they received your signal and agree to end the call.

Confirming Disconnection (SHUTDOWN): COMPLETE chunk): You then send a final "call ended" confirmation (SHUTDOWN-COMPLETE chunk) to your friend.

Aspects Securing SCTP Communications

Securing SCTP Communications: Transport Layer Security (TLS) Extension: For security in SCTP communications we can use TLS extension. It provides encryption, authentication, and integrity protection. SCTP can operate directly over TLS, ensuring endto-end security for applications. TLS employs certificate-based authentication to ensure communication with trusted peers.

Message Authentication Code (MAC): SCTP can include a Message Authentication Code (MAC) mechanism to verify packet integrity. MAC utilizes cryptographic algorithms like HMAC to prevent tampering during transmission.

IPsec Integration: IPsec, operating at the network layer, secures SCTP communications with encryption and authentication for IP packets. Integration of SCTP with IPsec ensures end-to-end security, safeguarding against listen in and tampering.

Selective Forwarding Unit (SFU) for Multihoming Security: SCTP supports multi-homing, but this introduces security risks that SFU extensions mitigate through secure address management and authentication.

Partial Reliability Extension (PR-SCTP): PRSCTP allows for partial reliability, prioritizing critical data transmission while offering flexibility for non-critical data.

Security Policies and Access Control: Enhancing SCTP security involves implementing security policies and access control at the application layer, using measures like access control lists and firewalls.

Encryption of Payload Data: Encrypting payload data between SCTP endpoints ensures confidence flexibility and privacy using application-layer encryption mechanisms. Considering these measures enhances SCTP communications' resilience against security threats, ensuring data integrity, confidentiality, and authenticity. However, specific security requirements should be evaluated for each SCTP deployment to choose appropriate measures.

Security: Keeping Your SCTP Conversations Safe Imagine you are sending messages over a walkie-talkie, but worried someone might be hearing it. SCTP security protects your communication in a few ways:

Encryption (like AES): The data you send, making it unreadable even if someone intercepts it. It's like whispering a secret code only you and the receiver understand.

Authentication (like TLS): This verifies the identity of the person you're talking to. It ensures it's actually your friend and not an imposter trying to eavesdrop. It's like checking their secret handshake to make sure it's them.

Message Integrity (like MACs): This makes sure the message you send delivered without being altered. It is like having a special code at the end of your message to confirm nothing got changed along the way.

Key Management: This securely creates, stores, and distributes the secret keys used for encryption and decryption. It's like keeping the keys to your secret code safe!

Secure Associations: This creates a safe connection between you and your friend, like a private tunnel on the walkie-talkie network.

Extra Security Tools for SCTP

DTLS over SCTP: This is a special encryption and authentication method particularly useful for realtime communication like voice calls (VoIP).

SCTP Authentication Extension (AE): This uses shared secret keys to verify who you're talking to, like a pre-arranged codeword only you and your friend know.

Other Extensions: These tools help SCTP handle different situations securely, like dealing with firewalls (NSIS), having multiple network connections (M-SCTP), or allowing some data loss in certain cases (PR-SCTP). By using these security features, SCTP ensures your conversations are confidential (no eavesdropping!), messages arrive unchanged (no tampering!), and you're talking to the right person (proper authentication!).

Dynamic Address Reconfiguration (DAR): Dynamic Address Reconfiguration (DAR) is an extension to the SCTP protocol that allows endpoints to modify their IP addresses in their associated lifetime. This is useful in scenarios where endpoints need to switch between different network interfaces or change IP addresses due to movability. Here's how DAR works and its benefits:

1. ***Address Addition and Removal:*** DAR allows endpoints to dynamically add or remove IP addresses from the list of addresses corelated with an SCTP association. This can be done

using the ADD_IP_ADDRESS or REMOVE_IP_ADDRESS chunk types.
2. ***Address Replacement:*** DAR also supports in replacing an existing IP address with a new one. This is useful when an endpoint is IP address that changes and it needs to inform the peer about the new address. This is done using the REPLACE_IP_ADDRESS chunk type.
3. ***Smooth Handover:*** DAR enables smooth handover in mobile scenarios. When a mobile moves between different networks and collects a new IP address, it can use DAR to update the peer about the change without disturbing the ongoing communication.
4. ***Redundancy and Load Balancing:*** DAR can also be used for redundancy and load balancing purposes. Endpoints can add multiple IP addresses to their list and use DAR to switch between them based on network conditions or load.
5. ***Improved Resilience:*** By supporting dynamic address reconfiguration, SCTP can improve the flexibility of communications. If one IP address is not available, the endpoint can switch to another address without interrupting the communication.

It's important to note that the DAR extension introduced additional complexity to the SCTP protocol, especially for handling address changes and ensuring that ongoing data transmission is not affected. Proper implementation and testing are crucial for the reliable operation of DAR in SCTP implementations.[2]

Ongoing Standardization and Development: SCTP has been standardized and the people are developing it which will be suitable to the next generation.

1. SCTP in future: there are efforts that make the SCTP suitable to next generation and which increase its performance. The improvements are made in areas like congestion control, multihoming, and security.

2. WebRTC: SCTP is used as a transport protocol in WebRTC (Web Real-Time Communication) in web browsers for communication purposes. The people are trying to make SCTP much better so that it helps in communications (real time communication), web browsers.

3. ***5G Networks:*** as the 5G networks are increasing in the usage, and SCTP is also being used for purposes like network slicing, lowlatency communication, and massive machine-type communication.

4. Security Enhancements: The security of SCTP is also important, like authentication, encryption, and protection against security threats. We need to make SCTP more secure and safe to use.

5. Performance improvements: the performance of SCTP is main of all, like its latency, ability to operate, throughput etc SCTP is a mature protocol, there are efforts for increasing its capabilities, security, performance etc.

Future……..

SCTP (Stream Control Transmission Protocol) is expected to remain important and improve in the future of networking. With more 5G networks and Internet-connected devices, SCTP's ability to handle multiple connections and deliver messages reliably could make it a top choice for applications needing dependable, fast communication. For things like internet calls (VoIP) and online games, SCTP's ability to send multiple data streams and handle data loss without disrupting the whole connection could make these experiences smoother. Security is a big deal, and SCTP's built-in features for keeping data safe make it a good option for secure communication needs. As edge computing becomes more popular, SCTP's abilities could be very useful for devices at the edge to talk to the cloud. The future of SCTP depends on how much it's used and improved to fit with new networking technologies. Its role and impact in the future of networks will be decided by how well it's adopted and integrated into these new technologies.

5G and IoT: As 5G networks become more common and the number of connected devices in the Internet of Things (IoT) keeps growing, there will be a greater need for communication protocols that are reliable and fast. SCTP, because it can handle multiple connections, send messages efficiently, and manage traffic well, might become more important in these situations.

Real-time Communication: As real-time communication apps like VoIP, video calls, and online games become more popular, there's a need for protocols that can deliver messages quickly and reliably. SCTP's abilities to send multiple streams of data and handle lost data make it a good fit for these apps.

Security: Security is a big worry in today's networks, so protocols must ensure secure communication. SCTP's ability to use encryption and authentication makes it a good choice for secure communication needs.

Edge Computing: As edge computing becomes more common, protocols that can help edge devices communicate with the cloud are important. SCTP's ability to handle multiple connections and support mobility could be very useful in these situations.

Standardization and Adoption: SCTP's future will also rely on how widely it's standardized and used. Continued efforts to make the protocol a standard and to get more people and companies to use it, along with integrating it into new and existing networking tech, will be crucial for its future.

Implementation: Implementing SCTP (Stream Control

Transmission Protocol) involves using the SCTP API provided by the operating system or a networking library. Steps involved are:

1. ***Choose a Programming Language:*** You can implement SCTP using a programming language that provides access to socket programming APIs, such as C, C++, Java, or Python.
2. ***Include the SCTP Header File:*** In C/C++, include the header file that defines the SCTP API. For example, in Linux, you would include '<netinet/sctp.h>'.
3. ***Create a Socket:*** Use the \`socket()\` function to create an SCTP socket. Specify the address family (\`AF_INET\` for IPv4 or \`AF_INET6\` for IPv6) and the socket type (\`SOCK_STREAM\` for a stream socket or \`SOCK_SEQPACKET\` for a message-oriented socket).
4. ***Bind the Socket:*** Use the \`bind()\` function to bind the socket to a specific IP address and port number.
5. ***Listen for Incoming Connections (Optional):*** If you're implementing a server, use the \`listen()\` function to listen for incoming connections.
6. ***Accept Connections (Optional):*** If you're implementing a server, use the \`accept()\` function to accept incoming connections and create a new socket for each connection.
7. ***Connect to a Server (Client Only):*** If you're implementing a client, use the \`connect()\` function to connect to a server.
8. ***Send and Receive Data:*** Use the \`sendmsg()\` and \`recvmsg()\` functions to send and receive data over the SCTP socket. These functions allow you to specify the stream number for message-oriented communication.
9. ***Handle Events:*** Use the \`sctp_recvmsg()\` function to handle incoming messages and events on the SCTP socket.
10. ***Close the Socket:*** Use the \`close()\` function to close the SCTP socket when you're done with it.

This is a basic explanation of how SCTP can be implemented. The exact steps might be different based on the programming language and the specific API provided by your operating system or networking library. Check the documentation for your environment for detailed instructions on how to implement SCTP. [3]

Implementations and Supported Features: Different operating systems offer SCTP implementations, each with unique supported features. Here's an overview of SCTP implementations on various systems:

Linux: Linux operating system has been in use for many years, it does use SCTP in it which controls mechanisms like multi-homing, message-oriented communication, and advanced congestion control.

Windows: Windows OS are the widely used operating system. It has been supporting SCTP like from its Windows Vista and Windows Server 2008 time. The implementation is not better than linux.

FreeBSD: FreeBSD supports the SCTP and its advanced features. It is known for its stability and performance in SCTP-based applications.

macOS: macOS is also one of the known OS. It supports the SCTP from its macOS 10.7 (Lion) version. Its implementation provides basic SCTP functionality but doesn't support some advanced features found in other OS.

Solaris: Solaris supports the SCTP and make it suitable for high-performance networking applications.

Other Operating Systems: Other operating systems like AIX, HP-UX, and OpenVMS also support the SCTP in their own ways The availability and features of SCTP implementations changes across operating systems. Before using we need check the suitable SCTP version to use.

CONCLUSION

SCTP (Stream Control Transmission Protocol) is a modern way for devices to talk to each other online. It's like a smarter version of older methods like TCP and UDP. It can do more things, like making sure messages arrive in order, handling different types of data, and keeping information safe from unauthorized access. SCTP is a good choice for many different kinds of internet communication because it's reliable, secure, and can work well with new technology. SCTP is not as widely used as TCP or UDP. But it has many unique features compared to the other protocols. As you see the network communications have been evolving and SCTP still remains reliable.

REFERENCES

Daniel Wallace; T. Abdallah ShamiDepartment of Electrical and Computer Engineering, University of Western Ontario, London, ONT, Canada, Publishedin: IEEE Communications Surveys & Tutorials (Volume: 14, Issue: 2, Second Quarter 2012). https://ieeexplore.ieee.org/document/5875919

Dierks T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Engineering Task Force, RFC 8446, Aug. 2018. https://dl.acm.org/doi/pdf/10.17487/RFC8446

Shruti Saini School of Computing, Information and Mathematical Sciences, The University of the South Pacific Suva, Fiji; Ansgar Fehnker Department of Computer Science University of Twente Enschede, the Netherlands; "Evaluating The Stream Control Transmission Protocol Using Uppaal". https://arxiv.org/pdf/1703.06568.pdf

Stewart, R. M. Ramalho, Q. Xie, M. Tuexen, P. Conrad, and L. Ong, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration," Internet Engineering Task Force, RFC 5061. https://www.researchgate.net/publication/247654732_Stream_Control_Transmission_Protocol_SCTP_Dynamic_Address_Reconfiguration

Stewart, R. M. Tüxen, "Stream Control Transmission Protocol (SCTP) Chunk Flags," Internet Engineering Task Force, RFC 4960, Sep. 2007. https://www.semanticscholar.org/paper/Stream-Control-Transmission-Protocol-(SCTP)-Chunk-T%C3%BCxen-Stewart/d8d6307f8e4181a219472c7f949377e8ccd8b27b

****