Demonstrating Advantages of SCTP over TCP

Vikas Kedia Nishit Desai Dheren Gala

November 28, 2004

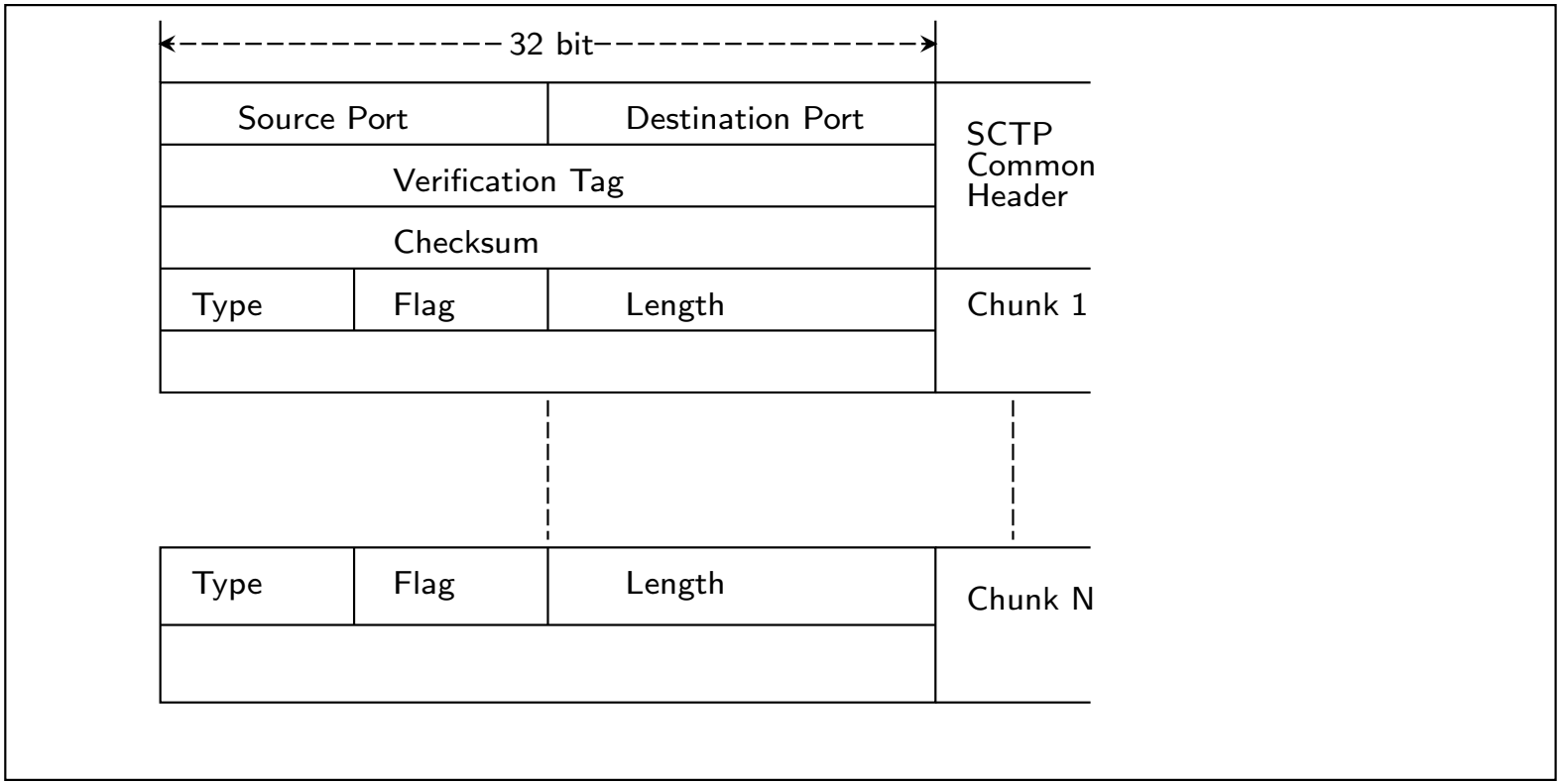
SCTP:General Features

- Equivalent level as TCP and UDP
- Reliable transport service
  - error-free
  - in sequence
- Session-oriented mechanism

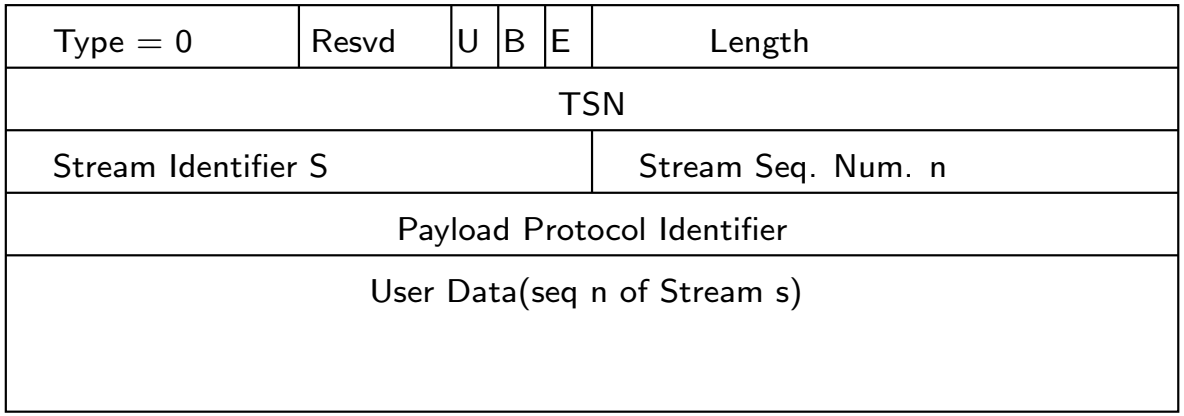
SCTP:Additional Capabilities

Comparison of TCP and SCTP		
Features	**TCP**	**SCTP**
multi-streams support	No	Yes
multi-homing support	No	Yes
preservation of message boundaries	No	Yes
unordered reliable message delivery	No	Yes

SCTP Header


32 bit
Source Port
Destination Port
Verification Tag
Checksum
SCTP Common Header
Type
Flag
Length
Chunk 1
Type
Flag
Length
Chunk N

SCTP Chunk



U: Unordered bit

B: Beginning segment bit

E: Ending segment bit

TSN: Transmission Sequence Number

Problem Definition

- To write a client-server or peer-to-peer application demonstrating features of SCTP
- To compare the performance of SCTP and TCP over various network conditions

Problem Definition

- To write a client-server or peer-to-peer application demonstrating features of SCTP
- To compare the performance of SCTP and TCP over various network conditions

The Application

- Client-sever application
- Transferring files between two computers
- Uses multi-streaming capability of SCTP
- Allows simultaneous transfer of multiple files

Protocol between Client and Server

Typical Sequence of operation

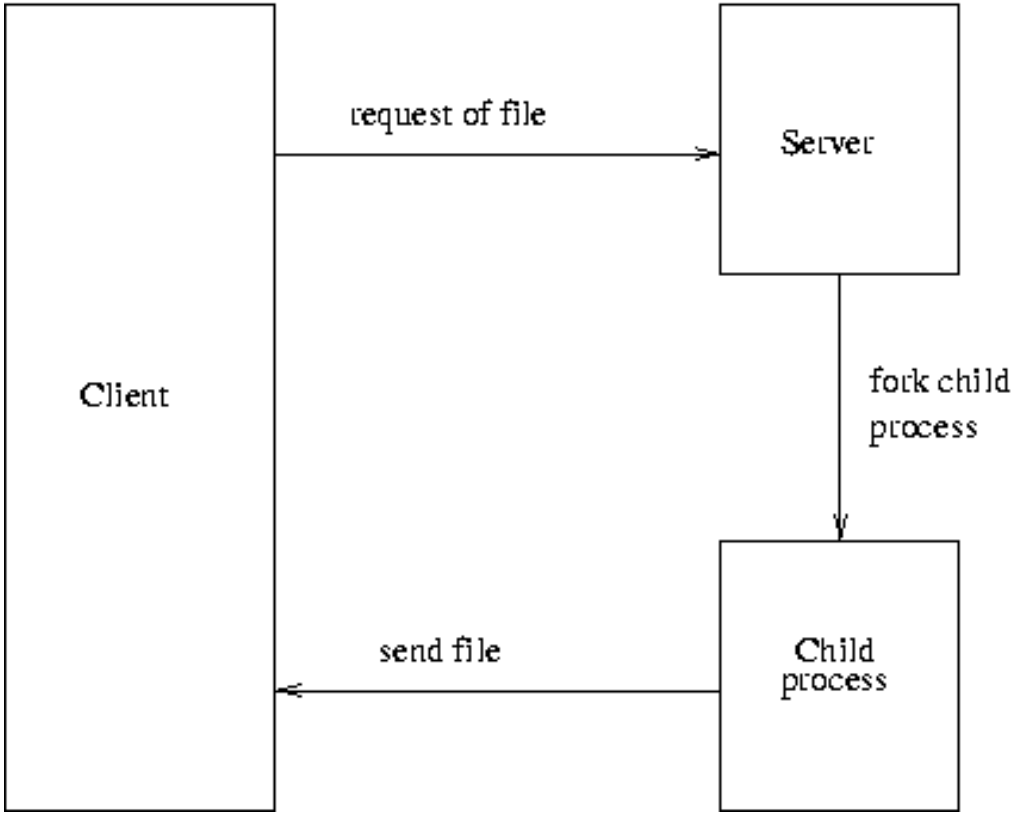
- Client sends user name to server
- Client sends password to server
- Change directory if necessary
- List content of directory
- Client requests for files and server sends the requested files.

Protocol between Client and Server

Protocol will be Command Response based. Client will send commands to the server and server will respond with appropriate response.

- USER $<$ *username* $>$
- PASS $<$ *password* $>$
- CDIR $<$ *directory* $>$
- LIST
- GETFILE $<$ *filename* $>$
- GETFILES $<$ *list of files* | *wild card expression* $>$

Single File Transfer : TCP and SCTP



GETFILES command

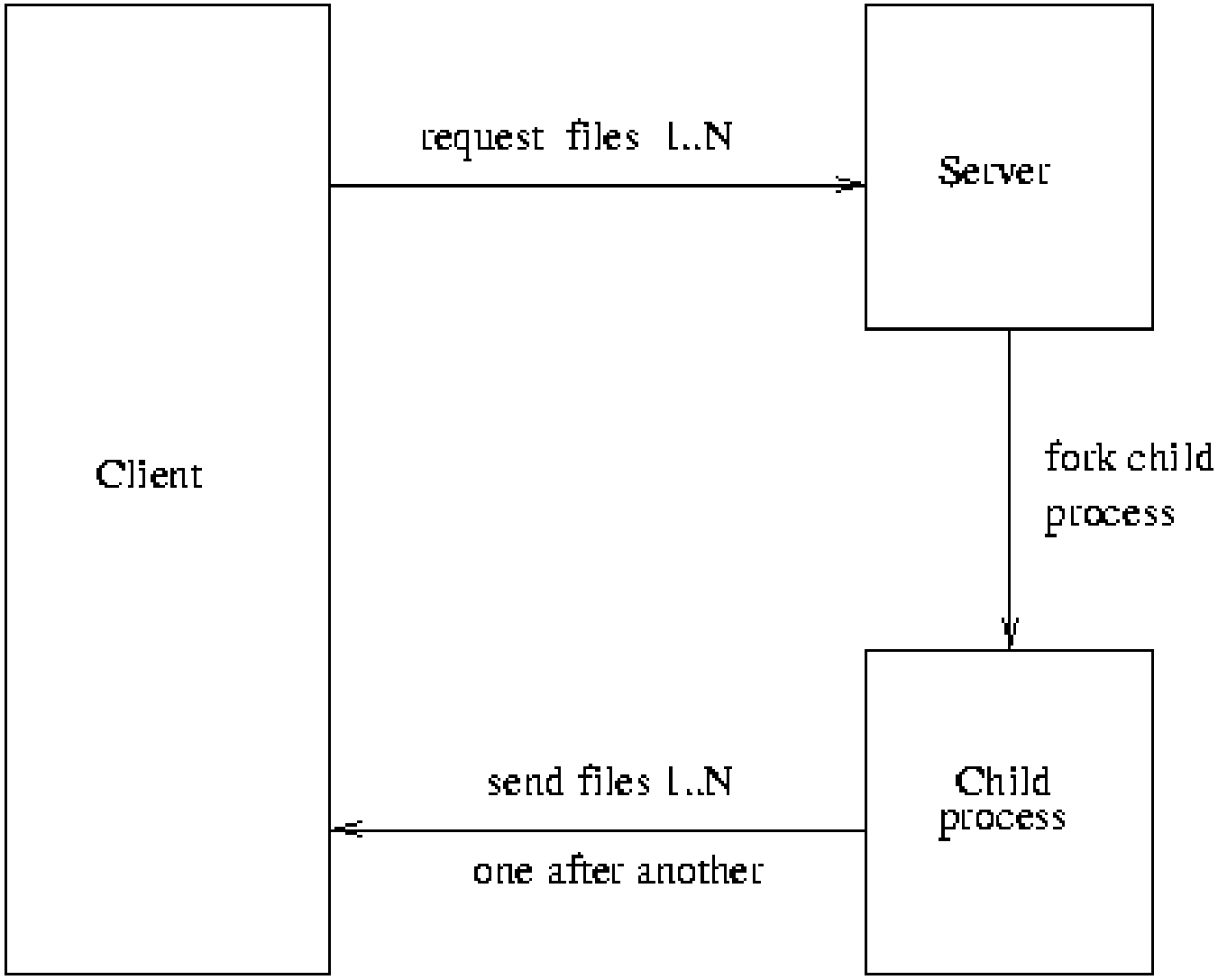
GETFILES $<$ *list of files* | *wild card expression* $>$

Server will evaluate the wild card expression to get the number and list of files.

Implementation over TCP

- Send the number of files to client
- Open a data connection and send all the files one after the other on the same connection

Multiple File Transfer : TCP

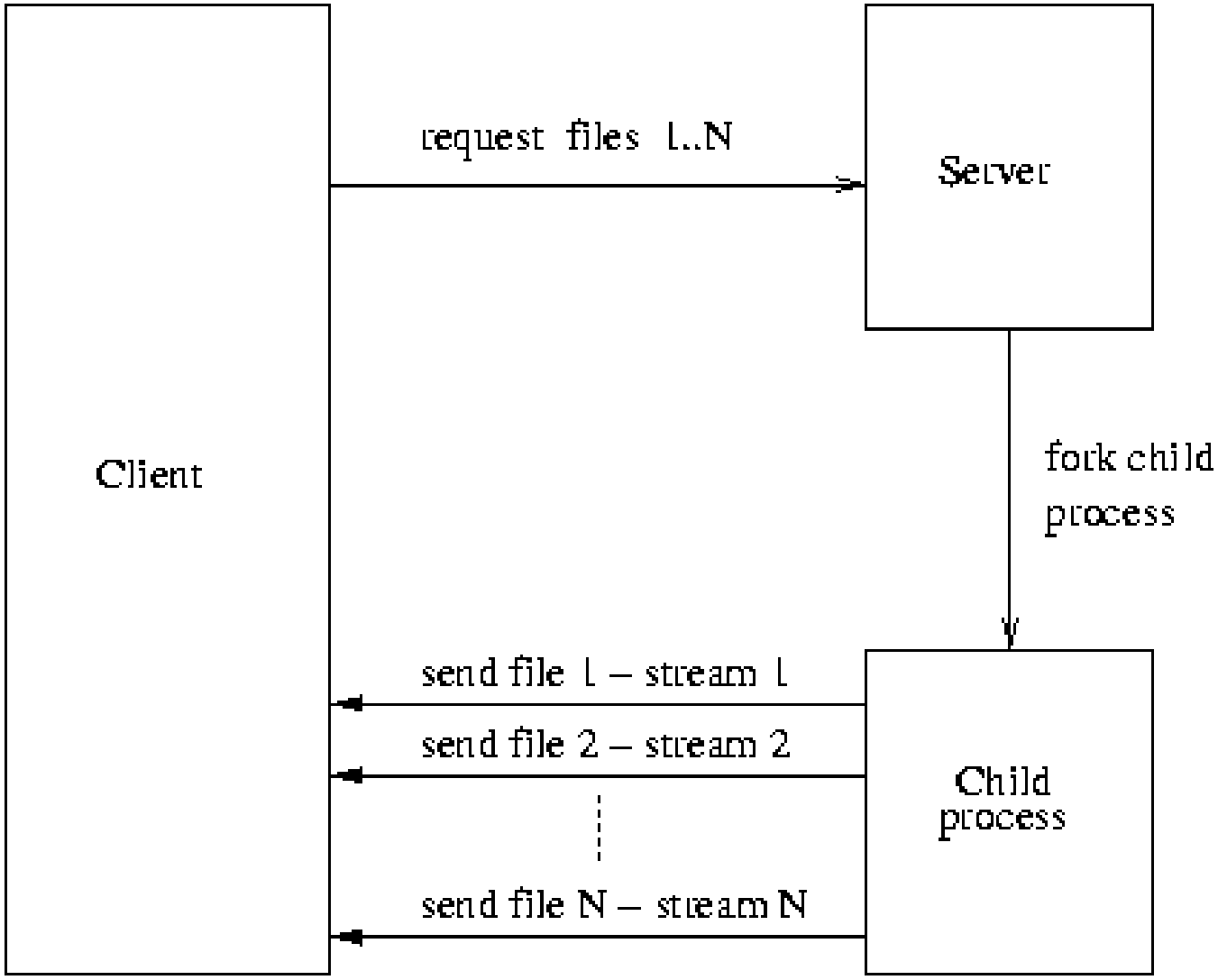


GETFILES command

GETFILES $<$ *list of files* | *wild card expression* $>$
Implementation over SCTP

- Send the number of files to client
- Open a data connection and negotiate with the client for number of streams.
- Send files over different streams
- If number of streams is less than the number of files then reuse the stream.

Multiple File Transfer : SCTP



SCTP Socket APIs

- Kernel imlementation of SCTP protocol available as a module
- Consistent with existing Socket APIs
- Protocol Type IPPROTO_SCTP
- Some differences to accomodate extra features of SCTP

SCTP Socket APIs

- *bindx()* to bind to multiple addresses
- *connect()* can be called multple times on a single socket
- *sendmsg(int sd, const struct msghdr *message, int flags)*
- *recvmsg(int sd, struct msghdr *message, int flags)*
- *msghdr* contains user message and ancilliary data containing SCTP specific paramameters such as SSN, Seq ID etc.
- SCTP stack also sends notifications which can be recieved by *recvmsg*

Negotiating Number of Streams

- Instead of *connect()* use *sendmsg()*
- In ancilliary data set SCTP_INIT as the mesg type and and set number of outbound streams equal to number of files
- Recieve SCTP_ASSOC_CHANGE notification, ancilliary data will contain the number of negotiated streams
- Start sending as many files as number of streams and then reuse streams if necessary

Sending and Recieving files over SCTP

- One to One or One to Many relationship between streams and files
- At a particular time each stream associated with a single file
- Server will send data on all the streams from the files associated with those streams
- First message sent on each stream will contain the list of files associated with the stream in sending order
- On recieving a message, Client will check the stream number and store data in appropriate file

Performance Measures

Latency Elapsed clock time between when a particular file is requested and when the file is fully received

Throughput Elapsed clock time between when all the files are requested and when they are fully received

Hypothesis

We believe that in packet-loss environment, both latency and throughput should be better in case of SCTP

Performance Measures

Latency Elapsed clock time between when a particular file is requested and when the file is fully received

Throughput Elapsed clock time between when all the files are requested and when they are fully received

Hypothesis

We believe that in packet-loss environment, both latency and throughput should be better in case of SCTP

Experimental Setup

- Three Linux machines
  1. client
  2. server
  3. router
- Router is used to control the packet loss rate between two endpoints

Emulating Packet Loss

- Kernel module to hook into the Kernel netfilter
- Will drop some of the incoming packets
- Drop percentage can be varied
- Bandwidth limitation can be simulated using iptables

Performance Evaluation

Single file transfer

- Compare latency of file transfer using TCP and SCTP for files of different sizes
- Compare latency in packet-loss environment for varying loss percentages

Multiple file transfer

- Compare latency and throughput in transferring multiple files for varying number of files
- Compare latency and throughput in packet-loss environment for varying loss percentages

Conclusion

- The designed application for multiple file transfer may be used to demonstrate multi-streaming feature of SCTP
- By simulating a lossy environment, performance improvement in SCTP can be verified

References


Mark Carson and Darrin Santay.
Nist net: a linux-based network emulation tool.
SIGCOMM Comput. Commun. Rev., 33(3):111–126, 2003.


Siemens L. Coene.
Stream Control Transmission Protocol Applicability Statement (RFC 3257)., April 2002.


Q. Xie R. Stewart.
Stream Control Transmission Protocol (RFC 2960)., October 2000.


Rajesh Rajamani, Sumit Kumar, and Nikhil Gupta.
SCTP versus TCP: Comparing the performance of transport protocols for web traffic, May 2002.

Dummy graph using Gnuplot