SCTP for Gophers

Why You Should Try SCTP In Your Next Golang Back-end Project

Apr 2019

Alexey Naidyonov
CSA, ITooLabs

What is SCTP?

A transport layer IP protocol (like TCP or UDP), used mostly to deliver telephone network signalling over IP.

```
SCTP is designed to transport Public Switched Telephone Network (PSTN)
signaling messages over IP networks, but is capable of broader applications.
```

RFC 4960 "Stream Control Transmission Protocol" (https://tools.ietf.org/html/rfc4960)

- Message-oriented
- Reliable
- Multi-stream
- Multi-home

Disclaimer

This talk does not cover:

- SCTP over UDP
- WebRTC Data Channels
- usrsctp
- ... or anything alike

SCTP Support

- Linux/FreeBSD/Solaris
- No (native) Windows support (usrsctp)
- No (native) Darwin support (usrsctp, kext)
- Weak NAT traversal support on consumer devices

Suitable for server side, not so for client applications

SCTP Features (1/2)

- Message-oriented
- Message boundary preservation
- Reliable data transfer w/SACK
- Unordered data delivery option
- Multi-stream support
- Message bundling
- Path MTU discovery, message fragmentation

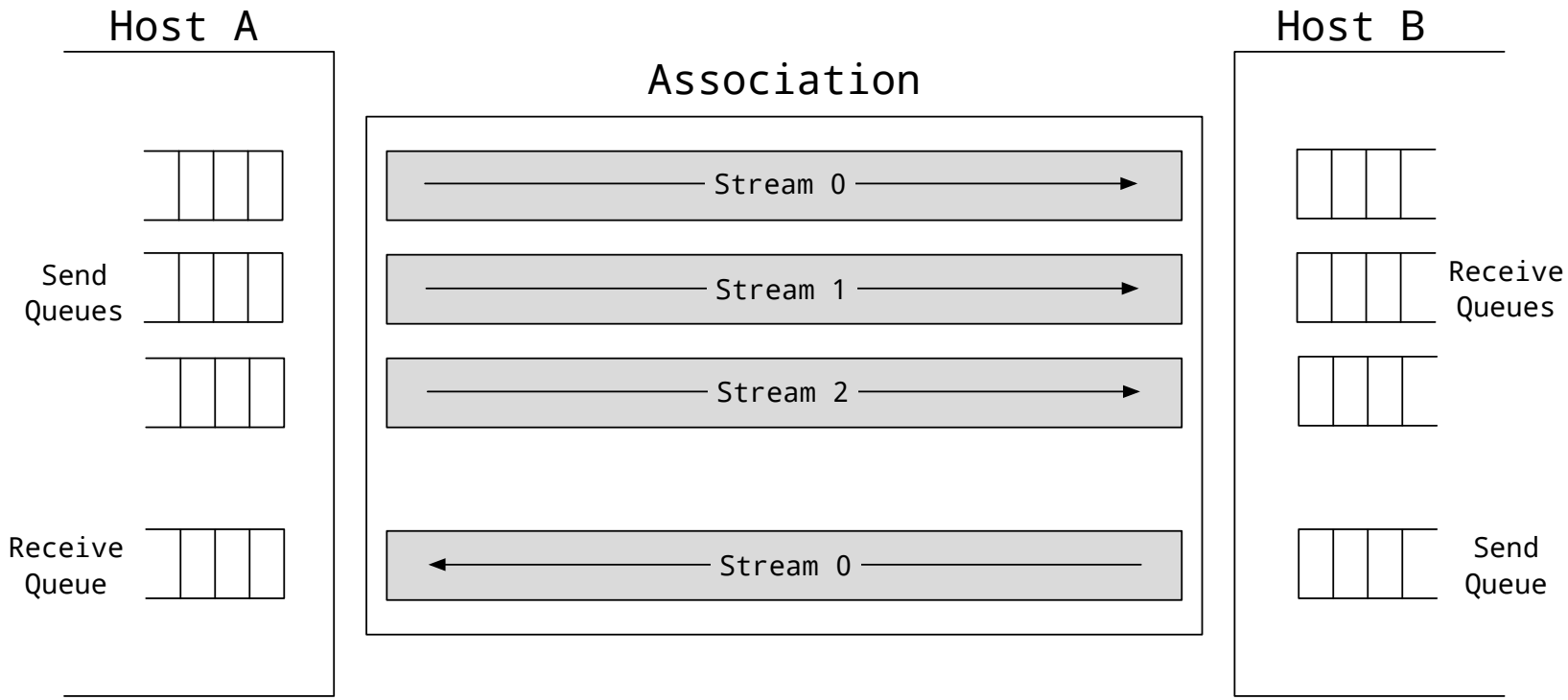
SCTP Features (2/2)

- Congestion control and avoidance
- Multi-homing support
- Built-in heartbeat (reachability check)
- 4-way handshake with security cookies to prevent connection flood attach
- 3-way shutdown
- One-to-one and one-to-many connections
- Extensible

SCTP Model

- **Endpoint**: an addressable logical endpoint, represented by single port number and one or more IP addresses
- **Association**: a logical relationship between two endpoints (≈ connection)
- **Stream**: a logical unidirectional channel transporting applications messages.
- **Message**: a user message delivered over stream
- **Packet**: PDU comprises of header and multiple **chunks**
- **Chunk**: a basic data unit containing either protocol control information (control chunk) or user data (data chunk)
- **Transmission Sequence Number** (TSN): 32-bit sequence number attached to every data chunk
- **Stream Sequence Number** (SSN): 16-bit sequence number attached to every message of a stream

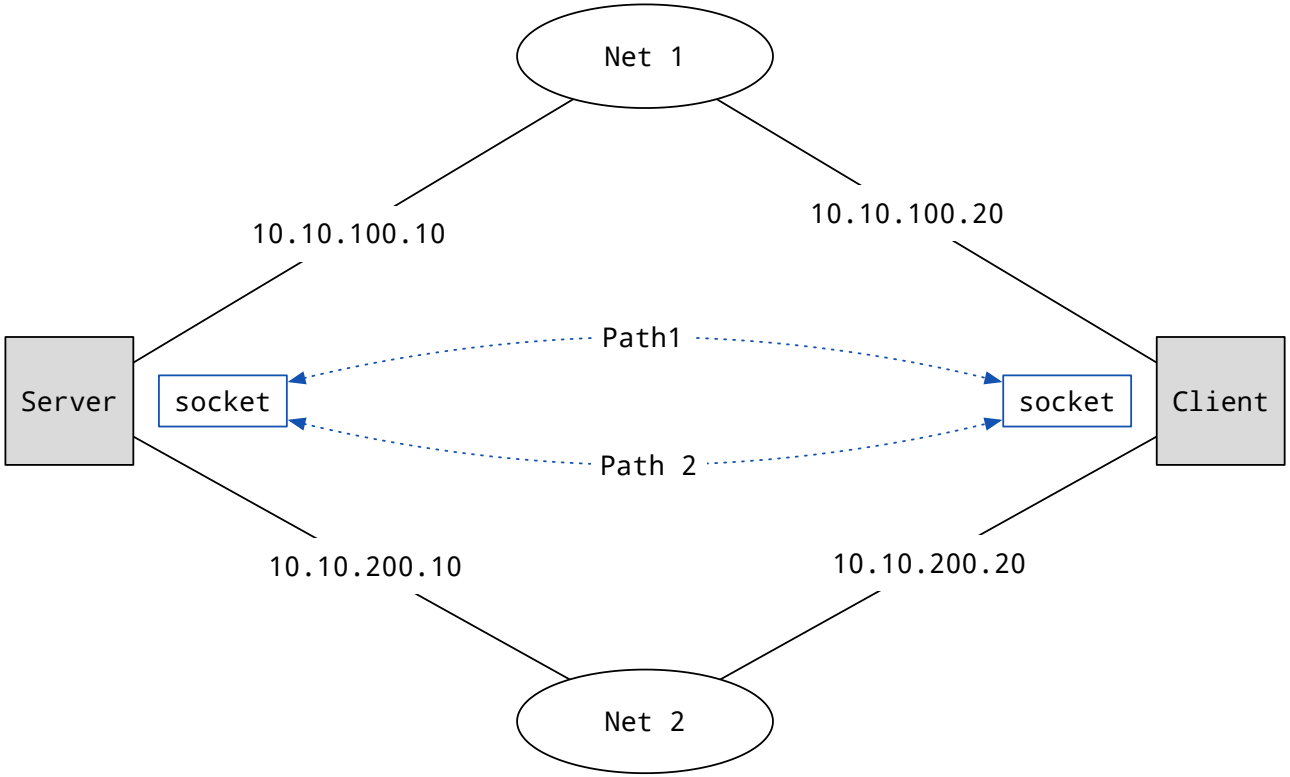
SCTP Multi-stream support



SCTP Multi-stream support

- Every stream is independent (no head-of-line blocking)
- Messages within a stream are ordered using the SSN
- Message fragments have the same SSN
- Message might be unordered
- Number of streams are negotiated at INIT
- Number of outbound streams: how many streams we plan to send?
- Max number of inbound streams: how many streams we wish to accept?
- Number of streams might be re-negotiated

SCTP Multi-homing support

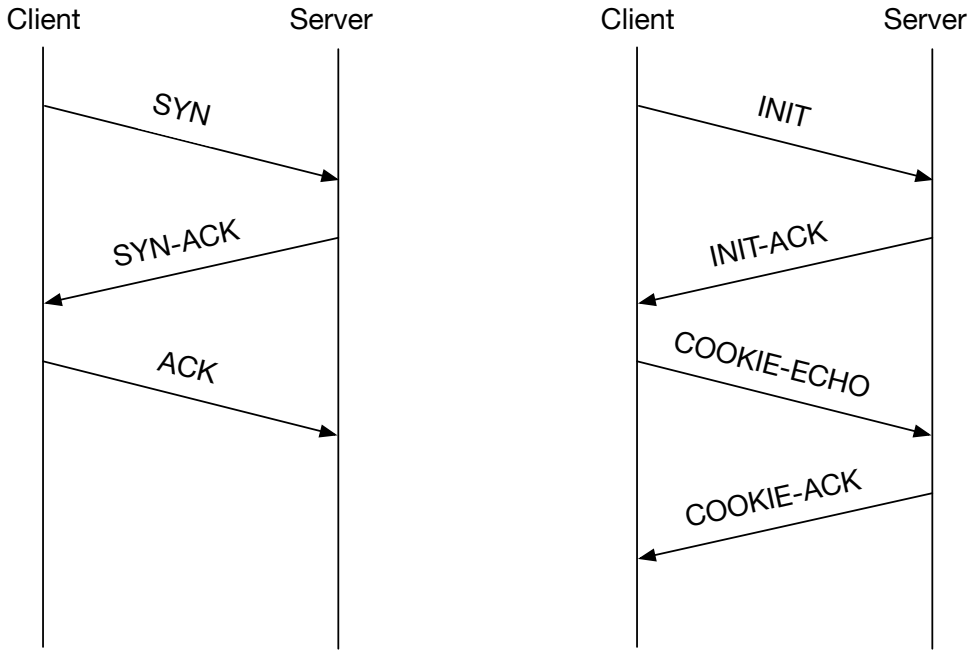


SCTP Multi-homing support

- Multi-homing means multiple transport addresses
- Makes sense when different IP addresses are bound to different network paths
- One path is *primary*, in case of failure SCTP falls back to *alternate* path
- API allows to send data to a specific transport address
- Heartbeat checks performed over all paths
- Addresses might be added or removed on the fly (RFC5061 (https://tools.ietf.org/html/rfc5061))

SCTP Association init and shutdown

- 4-way handshake



- 3-way shutdown
- No half-closed state

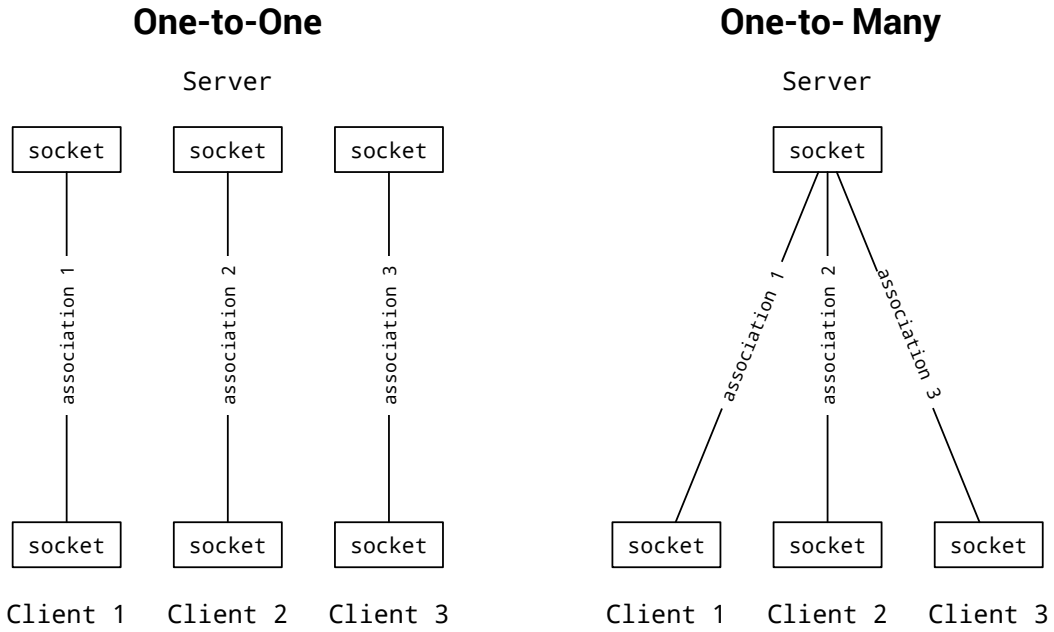
SCTP Flow control

- Guarantees receiver buffer will never overflow
- Acknowledges successful receipt of DATA chunks (using SACK chunks)
- Reports Gaps and Duplicated TSNs
- Guarantees fast recovery

SCTP API

- Not trivial due to model complexity
- Defined by RFC 6458 "Socket API Extensions for SCTP" (https://tools.ietf.org/html/rfc6458)
- One-to-one socket
- One-to-many socket
- Event notifications

SCTP API



- sctp_peeloff API branches a single one-to-many association to one-to-one socket
- SCTP_SENDALL flag forces broadcast to all one-to-many associations

SCTP Go API

- Multiple cgo
- Native github.com/nperez-messagebird/sctp (https://github.com/nperez-messagebird/sctp)
- Native ITooLabs, not open (yet)

SCTP Go API problems

- Go native networking is way too native
- Go `syscall.RawConn` (starting go 1.9)

```
type RawConn interface {
    // Control invokes f on the underlying connection's file
    // descriptor or handle.
    Control(f func(fd uintptr)) error

    // Read invokes f on the underlying connection's file
    // descriptor or handle; f is expected to try to read from the
    // file descriptor.
    Read(f func(fd uintptr) (done bool)) error

    // Write is like Read but for writing.
    Write(f func(fd uintptr) (done bool)) error
}
```

- (still not enough)

SCTP Go API: the minimal set of runtime calls required

```
//go:linkname sysSocket net.sysSocket
func sysSocket(family, sotype, proto int) (int, error)

//go:linkname newNetFD net.newFD
func newNetFD(sysfd, family, sotype int, net string) (unsafe.Pointer, error)

//go:linkname initNetFD net.(*netFD).init
func initNetFD(netfd unsafe.Pointer) error

//go:linkname setDeadline net.(*netFD).SetDeadline
func setDeadline(netfd unsafe.Pointer, t time.Time) error

//go:linkname seReadDeadline net.(*netFD).SetReadDeadline
func setReadDeadline(netfd unsafe.Pointer, t time.Time) error

//go:linkname setWriteDeadline net.(*netFD).SetWriteDeadline
func setWriteDeadline(netfd unsafe.Pointer, t time.Time) error

//go:linkname setsockopt syscall.setsockopt
func setsockopt(s int, level int, name int, val unsafe.Pointer, vallen uintptr) (err error)

//go:linkname getsockopt syscall.getsockopt
func getsockopt(s int, level int, name int, val unsafe.Pointer, vallen *_Socklen) (err error)
```

SCTP Go API:

Waiting for #15021 (https://github.com/golang/go/issues/15021) "Allow registrations for new socket type API"

SCTP Go API (github.com/nperez-messagebird/sctp)

```
// Addr represents SCTP end point address
//
// Single SCTP end point might be bound to multiple IP addresses.
//
type SCTPAddr struct {
    IPAddrs       []net.IPAddr
    Port          int
}

// Resolves SCTP Address. Multiple IP addresses are separated by "/"
// (i.e. 10.50.1.1/10.50.2.1:2904, 10.50.1.1/[2001:db8::1]:2904)
//
func ResolveSCTPAddr(addressFamily SCTPAddressFamily, addrs string) (*SCTPAddr, error)
```

SCTP Go API (github.com/nperez-messagebird/sctp)

```
type SCTPSocketMode int

const (
    OneToOne = SCTPSocketMode(iota)
    OneToMany
)

type InitMsg struct {
    NumOstreams    uint16
    MaxInstreams   uint16
    MaxAttempts    uint16
    MaxInitTimeout uint16
}

func NewSCTPListener(laddr *SCTPAddr, init InitMsg, mode SCTPSocketMode) (*SCTPListener, error)
func NewSCTPConnection(laddr, raddr *SCTPAddr, options InitMsg, mode SCTPSocketMode) (*SCTPConn, error)
```

SCTP Go API (github.com/nperez-messagebird/sctp)

```
type SndRcvInfo struct {
    Stream  uint16 // stream id
    SSN     uint16 // stream sequence number
    Flags   uint16 // flags (i.e. SCTP_UNORDERDED)
    _       uint16
    PPID    uint32 // supplied by peer app
    Context uint32 // message context (supplied by app)
    TTL     uint32 // time to live (ms)
    TSN     uint32 // TSN
    CumTSN  uint32 // Cumulative TSN
    AssocID int32  // association id
}

func SCTPRead(b []byte) (int, *OOBMessage, int, error)
func (o *OOBMessage) GetSndRcvInfo() *SndRcvInfo

func SCTPWrite(b []byte, info *SndRcvInfo) (int, error)
```

SCTP Go API (github.com/nperez-messagebird/sctp)

```
// read (one-to-one)
func (ln *SCTPConn) SCTPRead(b []byte) (n int, oob *OOBMessage, flags int, err error)

// write (one-to-one)
func (ln *SCTPConn) SCTPWrite(b []byte, info *SndRcvInfo) (n int, err error)
```

SCTP Go API (github.com/nperez-messagebird/sctp)

```
// Accept (one-to-one mode)
func (ln *SCTPListener) AcceptSCTP() (*SCTPConn, error)

// read (one-to-many mode)
func (ln *SCTPListener) SCTPRead(b []byte) (n int, oob *OOBMessage, flags int, err error)

// write (one-to-many mode)
func (ln *SCTPListener) SCTPWrite(b []byte, info *SndRcvInfo) (n int, err error)
```

SCTP Go API (github.com/nperez-messagebird/sctp)

```
func SubscribeEvents(int evts) error

const (
    SCTP_EVENT_DATA_IO = 1 << iota
    SCTP_EVENT_ASSOCIATION
    SCTP_EVENT_ADDRESS
    SCTP_EVENT_SEND_FAILURE
    SCTP_EVENT_PEER_ERROR
    SCTP_EVENT_SHUTDOWN
    SCTP_EVENT_PARTIAL_DELIVERY
    SCTP_EVENT_ADAPTATION_LAYER
    SCTP_EVENT_AUTHENTICATION
    SCTP_EVENT_SENDER_DRY
)
```

SCTP Go API (ITooLabs)

```
func NewSCTPSocket(network string, mode SocketMode) (*Socket, error)

func NewSCTPListener(network string, laddr string, mode SocketMode) (*Listener, error)

// Conn is an association, not the socket itself
func DialSCTP(network string, laddr *Addr, mode SocketMode) (*Conn, error)

func (l *Listener) Accept() (*Conn, error)

func (c *Conn) PeelOff() error

// Stream returns a stream handler
func (c *Conn) IStream(sid uint16) (*IStream, error) {...}
func (c *Conn) OStream(sid uint16) (*OStream, error) {...}

func (s *OStream) WriteMsg([]byte) error
func (s *IStream) RecvMsg([]byte) error
```

SCTP Go API (Events)

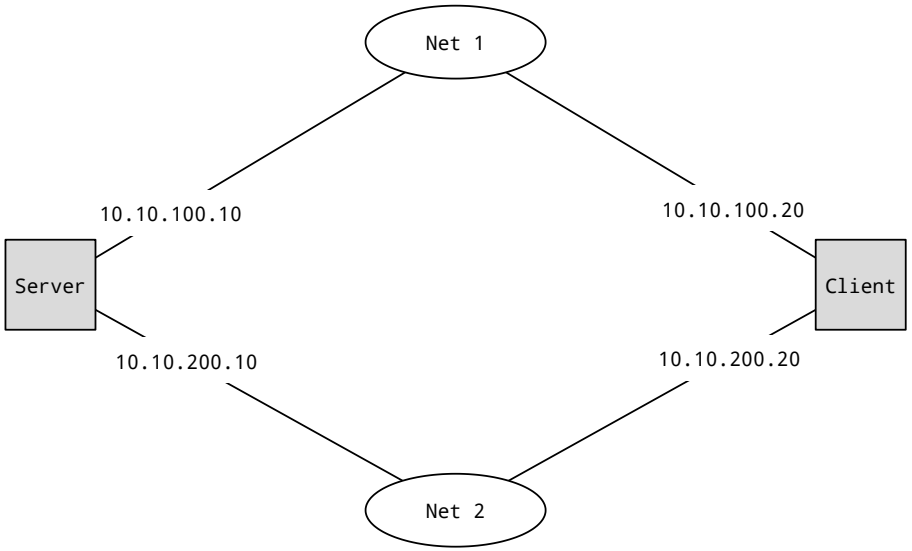
```
// Represents SCTP notification event
type Event interface {
    Type()  uint16 // SCTP event type
    Flags() uint16 // SCTP event flags
    Len()   uint32 // SCTP event length, including header
}

// EventPeerAddrChange
// EventAssocChange
// AuthenticationEvent
// ShutdownEvent
// SenderDryEvent
// PartialDeliveryEvent
// SendFailureEvent
// ...

func (s *Socket) SubscribeEvents(events EventKind...)
func (s *Socket) Events() <-chan Event
```

Multi-homing demo

- Environment: VirtualBox + Docker + OpenVSWitch
- 2 virtual networks (10.10.100/24, 10.10.200/24)
- 2 containers (client, server)



SCTP: Where to use

- A distributed multi-node multi-service application (seem to be ~80% of all Go applications out there)
- Strict requirements for high availability and high resiliency
- Controlled multi-path networking environment

Thank you

Apr 2019
Tags: SCTP, networking (#ZgotmplZ)

Alexey Naidyonov
CSA, ITooLabs
anaidyonov@itoolabs.com (mailto:anaidyonov@itoolabs.com)
https://itoolabs.com/ (https://itoolabs.com/)
@growler (http://twitter.com/growler)