SCTP for the Application Developer

Michael Tüxen
tuexen@fh-muenster.de

Why use SCTP?

- Which services does the application require from the transport layer?
- Services provided by UDP and TCP are extreme:
  - Either totally unordered and unreliable.
  - Or totally ordered and reliable.
- Not doing more than required improves the performance.
- Some services are provided by SCTP only.

Basic SCTP features

- Connection oriented (SCTP association).
- Message oriented.
- Supports fragmentation and reassembly of large messages.
- Provides congestion and flow control.
- Most protocol parameters are configurable.
- Runs on top of IPv4 and IPv6.

Support of Multihoming

- An SCTP end-point has one port number and one or more IP-addresses.
- It supports IPv4 and IPv6.
- Addresses are negotiated during association setup.
- Addresses can dynamically be changed during the live time of an association when using the ADD-IP extension.
- Multiple addresses are used for redundancy. Load-sharing is possible but not yet standardized.

Message Ordering

- An SCTP association has in each direction a number of streams.
- Streams are uni-directional message channels.
- The number is limited by 64K and negotiated during SCTP association setup.
- Message ordering is preserved only for messages sent on the same stream.
- Upon user request, messages can be sent without ordering constraints.
- There is an ID about resetting and adding streams during the association live time.

Message Reliability

- All messages are transferred reliably per default.
- Using PR-SCTP (partial reliability extension) the sender can stop sending a message.
- Policies include:
  - Limiting the live time of a message.
  - Limiting the number of retransmissions.
  - Discarding low priority messages when the send buffer is full.

Availability of Implementations

- Supported by standard kernels of
  - FreeBSD
  - Linux
  - Solaris
- A loadable kernel extension for Mac OS X (unfortunately not from Apple).
- A user-land stack (which also supports Windows).
- All kernel implementations use a socket based API. Programs are portable.

Socket API

- Defined in draft-ietf-tsvwg-sctpsocket-19.txt.
- It is very easy to port TCP or UDP based applications.
- Using SCTP specific features requires the use of socket options.
- The SCTP stack can inform the application about network events, if interested in.

A Simple Server (UDP-like)

```
int main() {
    int fd;
    struct sockaddr_in addr;
    char buffer[SIZE];

    fd = socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);

    addr.sin_family = AF_INET;
    addr.sin_len = sizeof(struct sockaddr_in);
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr(ADDR);
    bind(fd, (const struct sockaddr *)&addr, sizeof(struct sockaddr_in));

    listen(fd, 1);

    while (1) {
        recv(fd, buffer, SIZE, 0);
    }

    close(fd);
    return 0;
}
```

A Simple Client (TCP-like)

```
int main() {
    int fd,
    struct sockaddr_in addr;
    char buffer[SIZE];

    fd = socket(AF_INET, SOCK_STREAM, IPPROTO_SCTP);

    addr.sin_family = AF_INET;
    addr.sin_len = sizeof(struct sockaddr_in);
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr(ADDR);
    connect(fd, (const sockaddr *)&addr, sizeof(struct sockaddr_in));

    memset((void *)buffer, 'A', SIZE):
    send(fd, buffer, SIZE, 0);

    close(fd);
    return 0;
}
```

Another Simple Client

```
int main() {
    int fd;
    struct sockaddr_in addr;
    char buffer[SIZE];

    fd = socket(AF_INET, SOCK_SEQPACKET, IPPROTO_SCTP);

    addr.sin_family = AF_INET;
    addr.sin_len = sizeof(struct sockaddr_in);
    addr.sin_port = htons(PORT);
    addr.sin_addr.s_addr = inet_addr(ADDR);

    memset((void *)buffer, 'A', SIZE);
    sctp_sendmsg(fd,
                 (const void *)buffer, SIZE,
                 (const struct sockaddr *)&addr, sizeof(struct sockaddr_in),
                 htonl(PPID),
                 SCTP_EOF|SCTP_UNORDERED, SID,
                 TIMETOLIVE, CONTEXT);
    close(fd);
    return 0;
}
```

Deployment Considerations

- Transport layer security: DTLS/SCTP
  draft-ietf-tsvwg-dtls-for-sctp-01.txt
- NAT traversal:
  - UDP Tunneling
    draft-tuexen-sctp-udp-encaps-02.txt
  - SCTP aware NAT
    draft-ietf-behave-sctpnat-01.txt

HTTP over SCTP
draft-natarajan-http-over-sctp-02.txt

Fred Baker, Preethi Natarajan
{prenatar,fred}@cisco.com
Presenting: Michael Tüxen
tuexen@fh-muenster.de

A Case Study: HTTP/SCTP

- No change to the HTTP protocol.
- The client tries to use different outgoing streams for different requests.
- The server sends responses on the outgoing stream corresponding to the incoming stream where the request was received.
- Using a combination of the stream reset feature and PR-SCTP you can also abort transmitting data from the server to the client when the client does not need it anymore.

A Demo

- University of Delaware added support for SCTP to Apache and Firefox.
- The emulated path had a 56Kbps bandwidth and a 1080ms RTT, intended to represent an typical communication using satellites from a developing nation.
- More results are available via http://www.cis.udel.edu/~leighton/

HTTP/SCTP or HTTP/TCP

- SRV based solution.
- New URI for SCTP: “http-sctp://”.
- Try both. Use whichever connection is established first. Discussed in draft-wing-http-new-tech-00.txt

Questions

- … can also be sent to prenatar@cisco.com.