Discussion: Messaging

Michael Welzl

neat

TAPS @ IETF 98
Chicago, 28.3.2017

From draft-gjessing-taps-minset-04

- Transport features that require app knowledge + allow fall-back to TCP

- Sending
  - Reliably transfer data, with congestion control
  - Reliably transfer a **message**, with congestion control
  - Unreliably transfer a **message**
  - Configurable **Message** Reliability
  - Choice between unordered (potentially faster) or ordered delivery of **messages**
  - Request not to bundle **messages**
  - Specifying a key id to be used to authenticate a **message**
  - Request not to delay the acknowledgement (SACK) of a **message**

- Receiving
  - Receive **data (with no message delineation)**
  - Information about partial message arrival

Makes no sense when we don't get messages.

Sending messages, receiving a bytestream

- Can we make this combination work?
  - Be compatible to TCP but still benefit from messages?

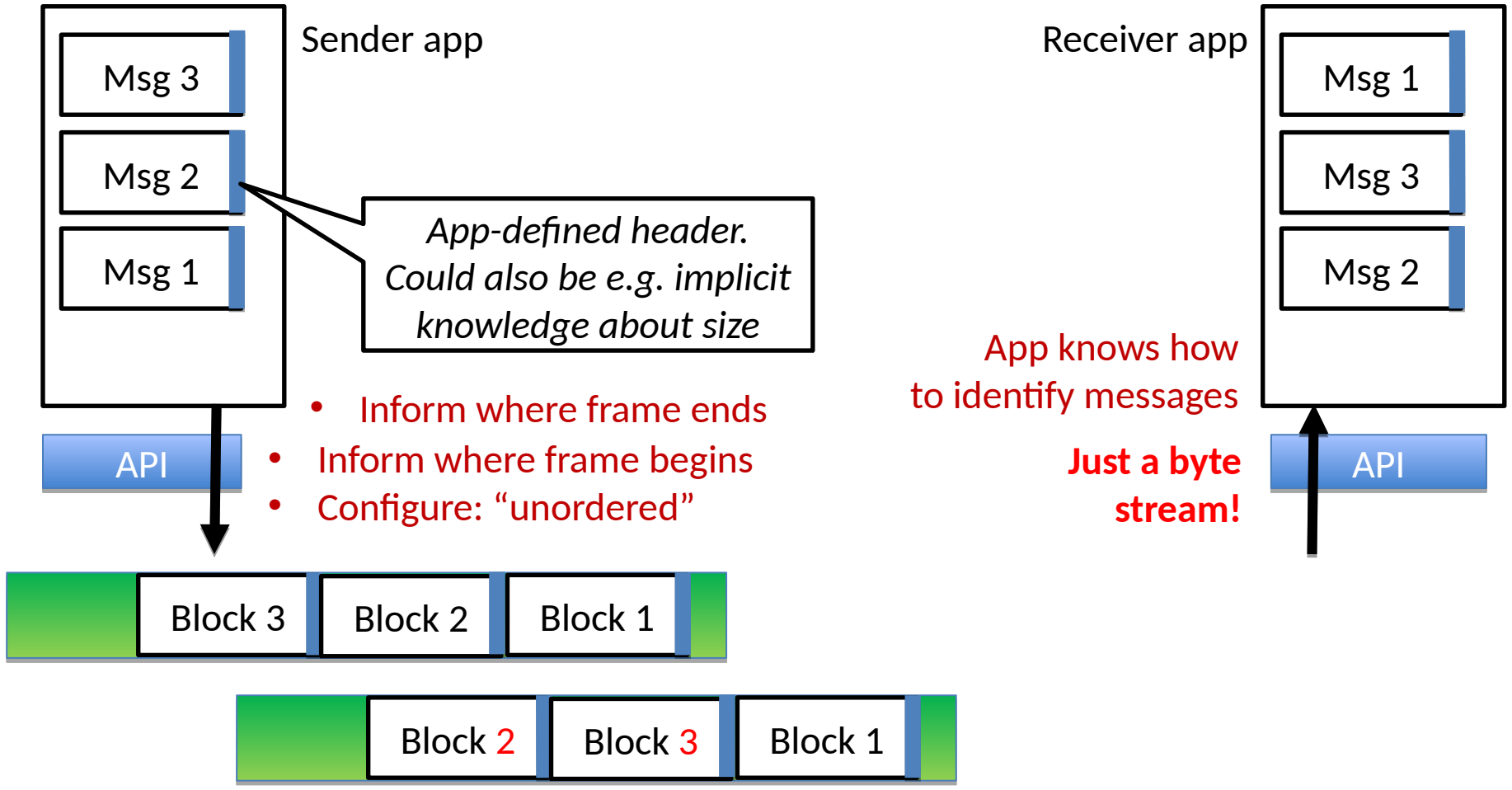
- Alternative not very attractive: always telling an application “sorry, you only get a stream here” is not much different than saying “sorry, use TCP instead”
  - Let’s minimize # hoops an app developer has to jump through

- Message-oriented TCP apps already frame their data
  - Unnecessary to repeat this in transport layer
  - Requirement to tell receiver app “*here is your complete message*” creates a major limitation and is often unnecessary

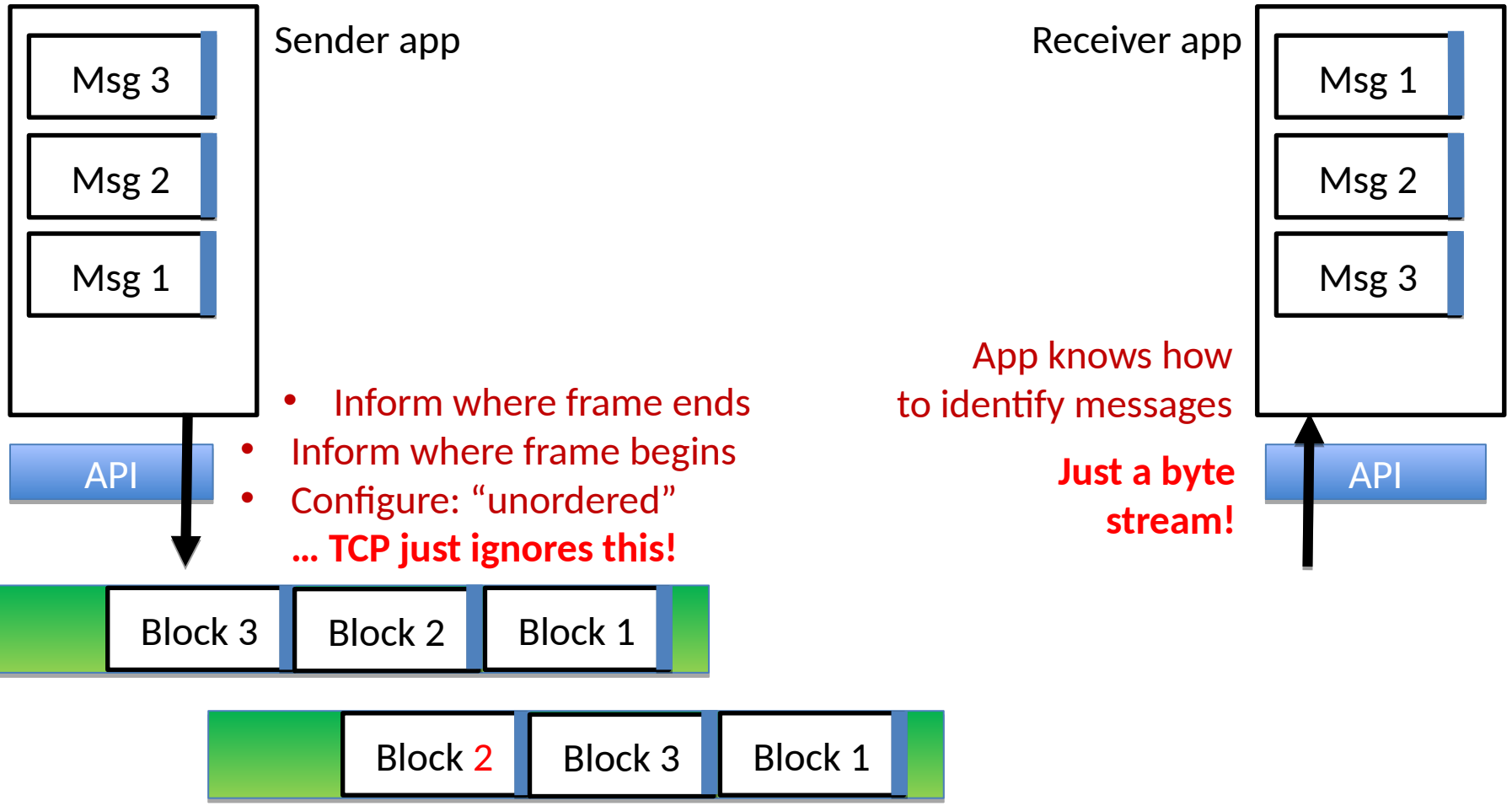
Application-Framed (AFra-)Bytestream

- Normal TCP-like bytestream API
  - Optional: some additional information provided by sender app
- Sender app: hands over a stream of bytes, informs transport about frame boundaries and requirements (order, reliability, ..)
  - Delimited **frames** stay intact, in order
  - More relaxed rules possible between frames
  - Delimiters assumed to be known by application
- Receiver app: receives stream of bytes
  - App-level delimiters turn it into messages
- TCP = special case: no delimiters used
  - Can talk to “normal” TCP applications on both sides

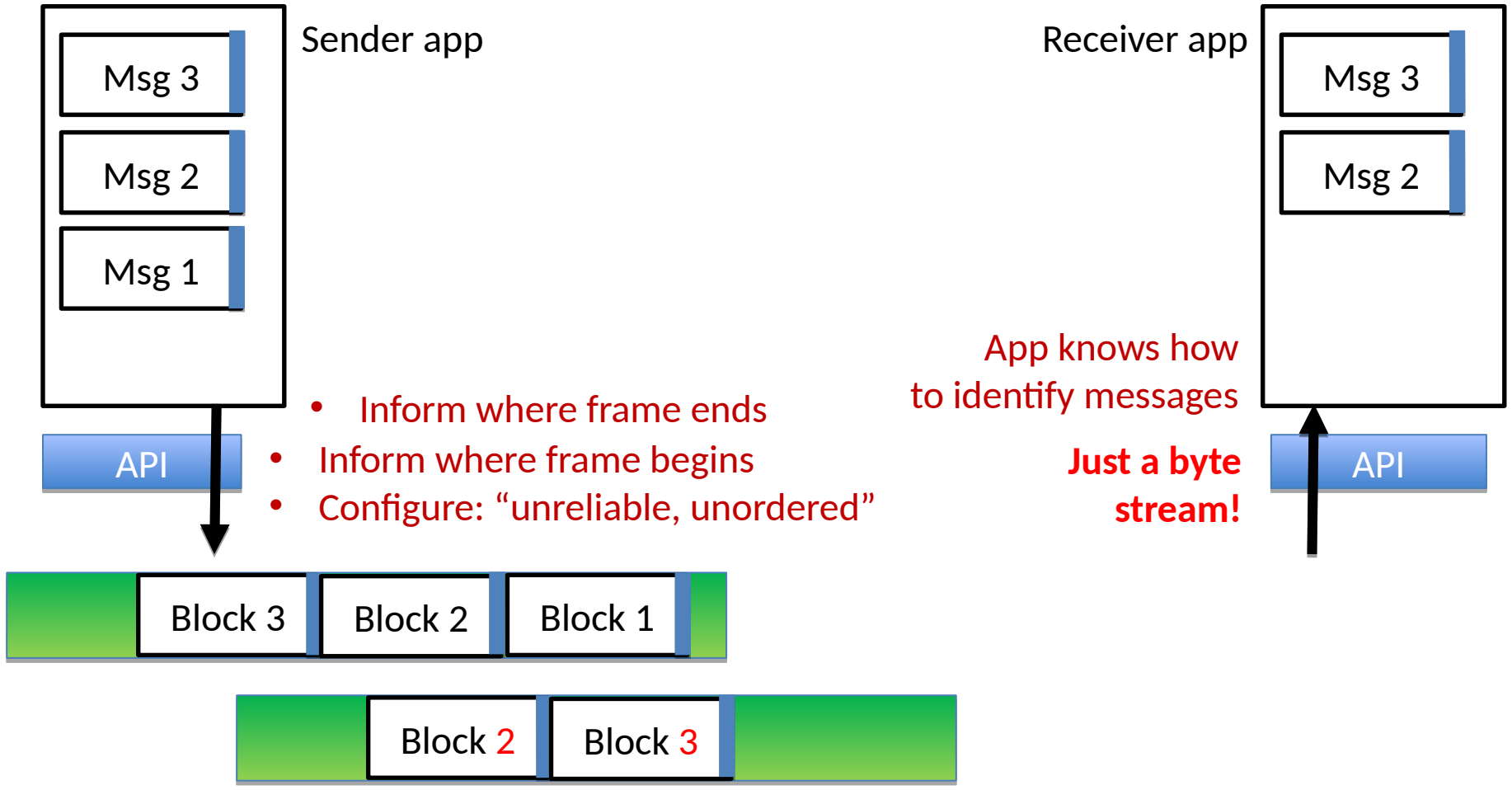
Unordered message delivery: SCTP



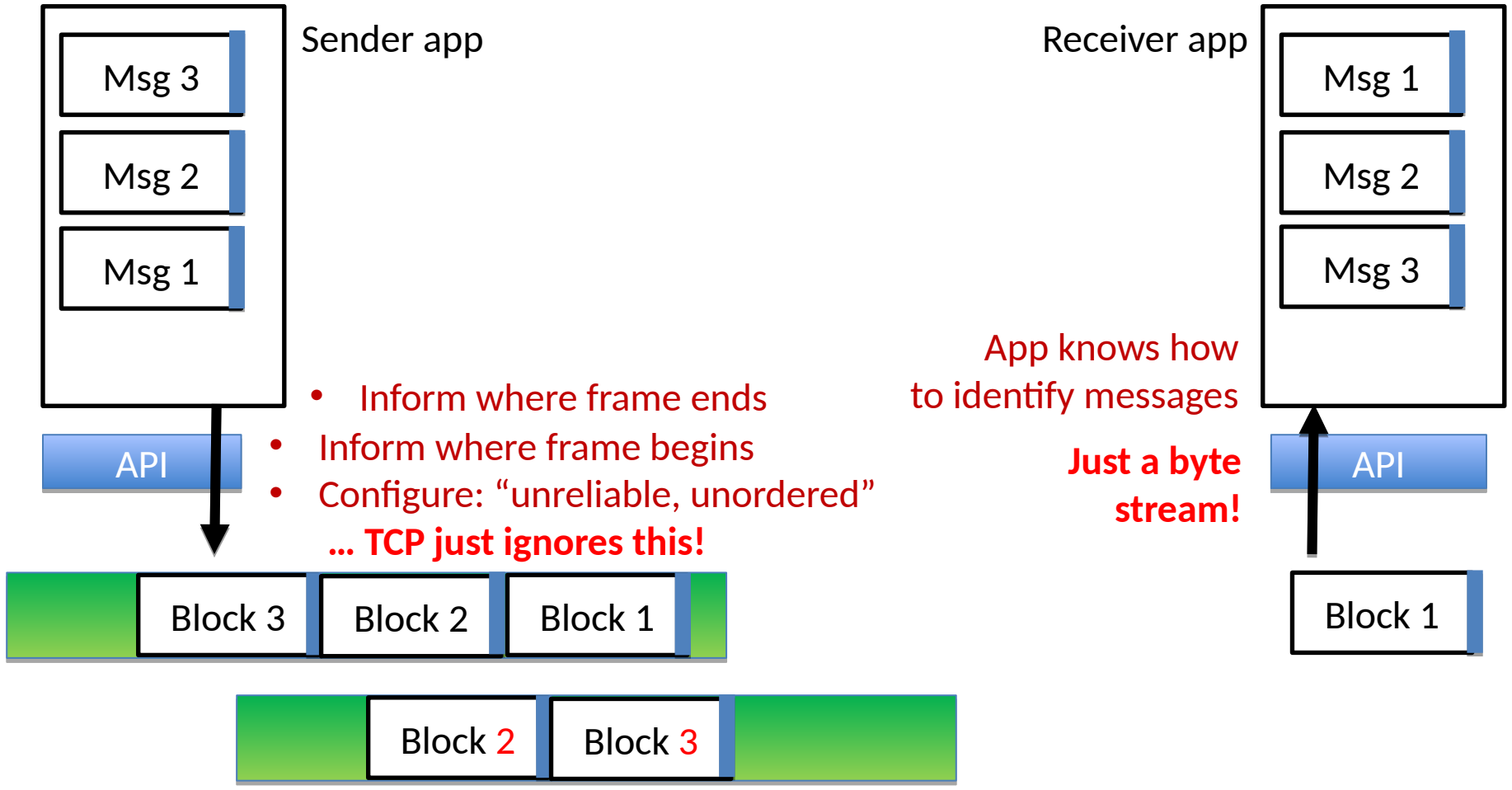
Unordered message delivery: TCP



Unreliable unordered msg delivery: SCTP

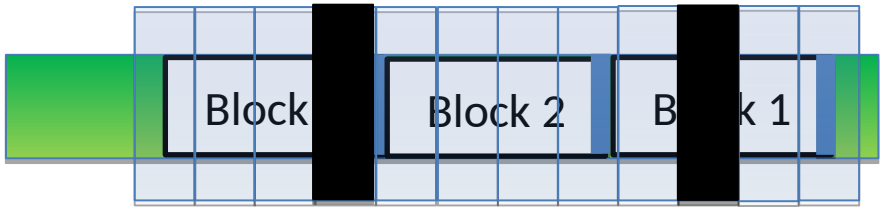


Unreliable unordered msg delivery: TCP



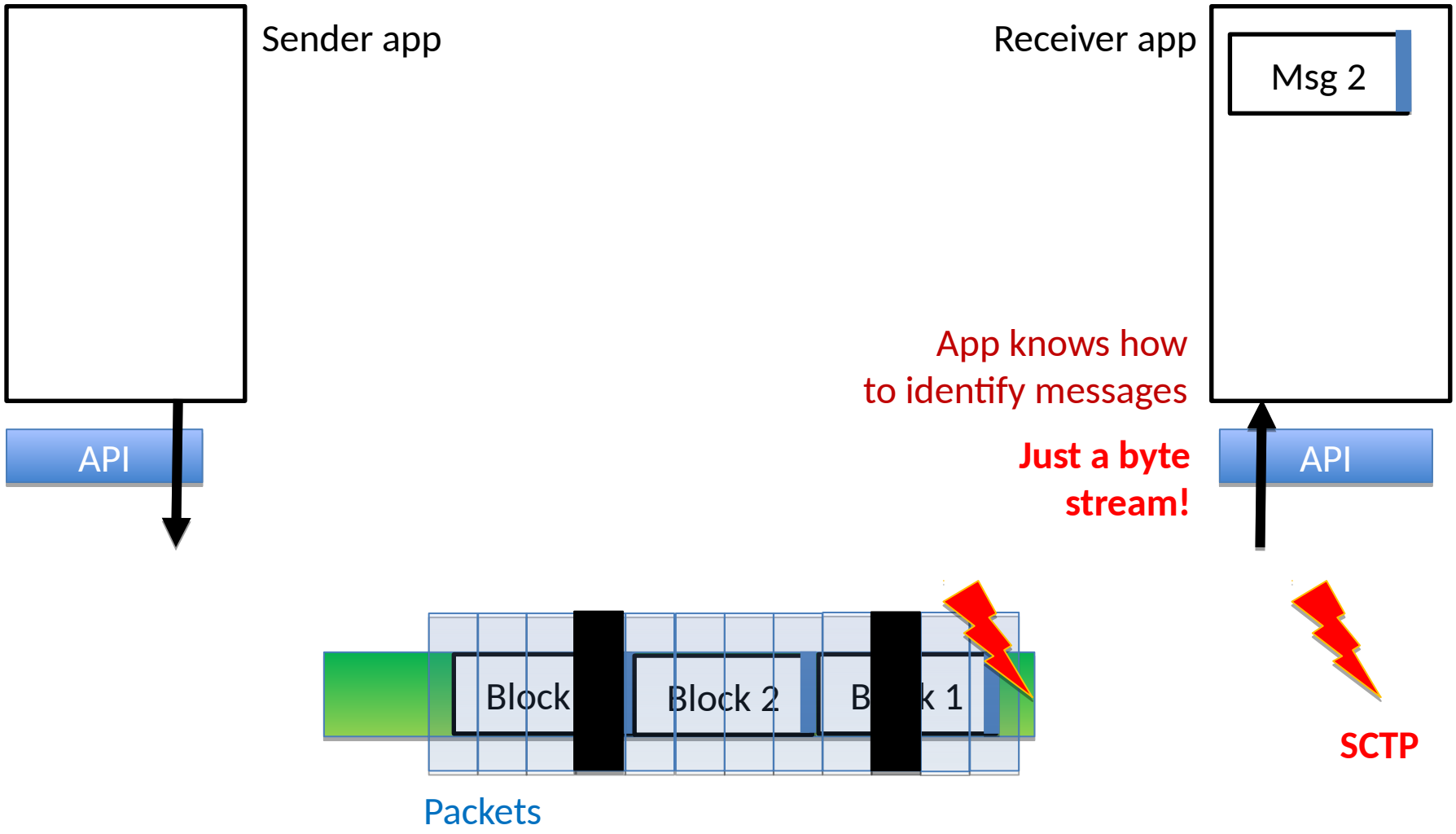
Unreliable message delivery:
SCTP, large messages

- Inform where frame ends
- Inform where block begins
- Configure: “Unreliable”



Packets

Unreliable message delivery:
SCTP, large messages



Questions, comments?

Discussion:
Early data transmission

Michael Welzl

neat

TAPS @ IETF 98
Chicago, 28.3.2017

From draft-gjessing-taps-minset-04

Transport features that require app knowledge + allow fall-back to TCP

1. Hand over a message to transfer (possibly multiple times) before connection establishment
   - This is **TCP** (TFO)

2. Hand over a message to transfer during connection establishment
   - This is **SCTP** sending data together with Cookie-Echo, or **TCP** sending data on SYN without TFO
   - no duplication

Proposal in draft-gjessing-taps-minset-04

- Flow is created before connecting or listening
  - Allows for some early configuration
  - At this stage, deal with early data

- App can…
  1. hand over a message
  2. say whether it prefers “before” (case 1) or “during” (case 2) establishment
  3. query for the maximum amount of data that it can possibly expect to have transmitted before or during connection establishment

Questions, comments?