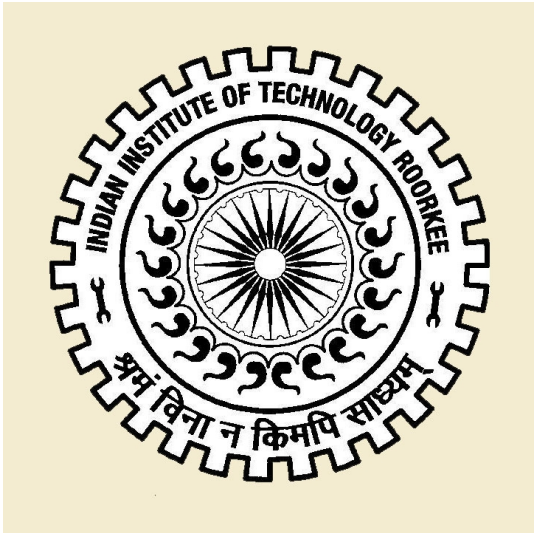
Realizing the aims of Transport-Next Generation (TNG): COBS and SSL Minion

Presented By:

Dishant Ailawadi



Outline

- Motivation: Ad-hoc evolution of internet
- Brief Introduction to the TNG Project
- Minions
- Problem Introduction
- COBS Minion
- SSL Minion
- Network Simulator
- Future Work
- References

Ad-hoc evloution.

- The current Internet architecture tightly bundles several functions into the Transport Layer, all of which were originally intended to operate "end-to-end" between hosts.

We are still using the TCP/IP settings evolved in 1974 (Cerf et al) when user count was merely in few thousands!

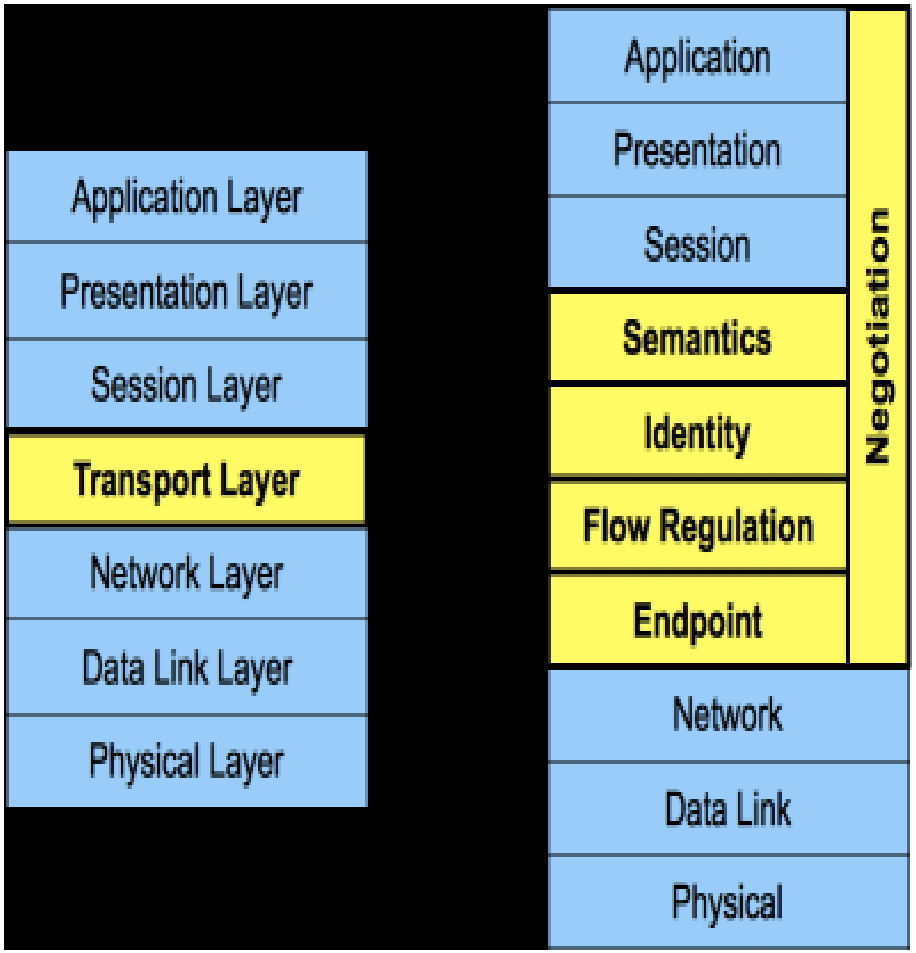
- End-to-end model has been under duress due to proliferation of middleboxes like NATs, Firewall.

- Legacy transport work well but deployment black-holes for newer technology.

- Hence, Limiting evolution of internet.

Transport Next Generation(TNG)

A new transport service architecture that decomposes "true" end-to-end transport functions such as reliable packet delivery and security from middlebox-relevant functions such as endpoint naming and congestion control.

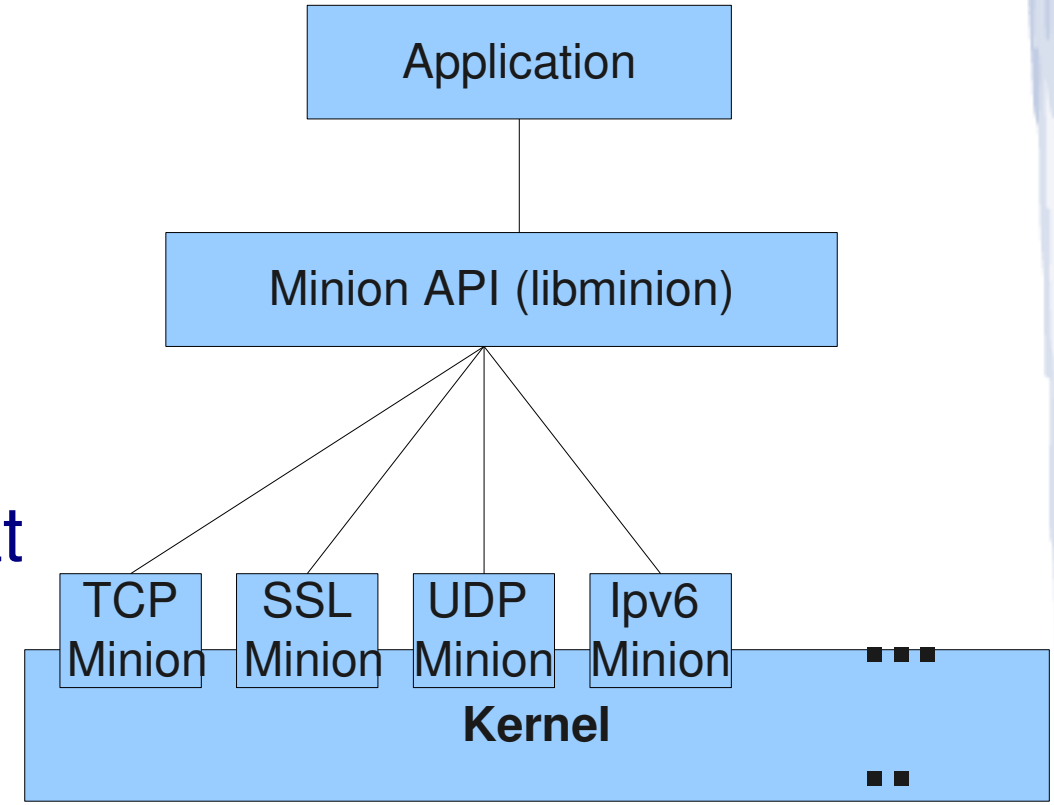


The Minion Suite

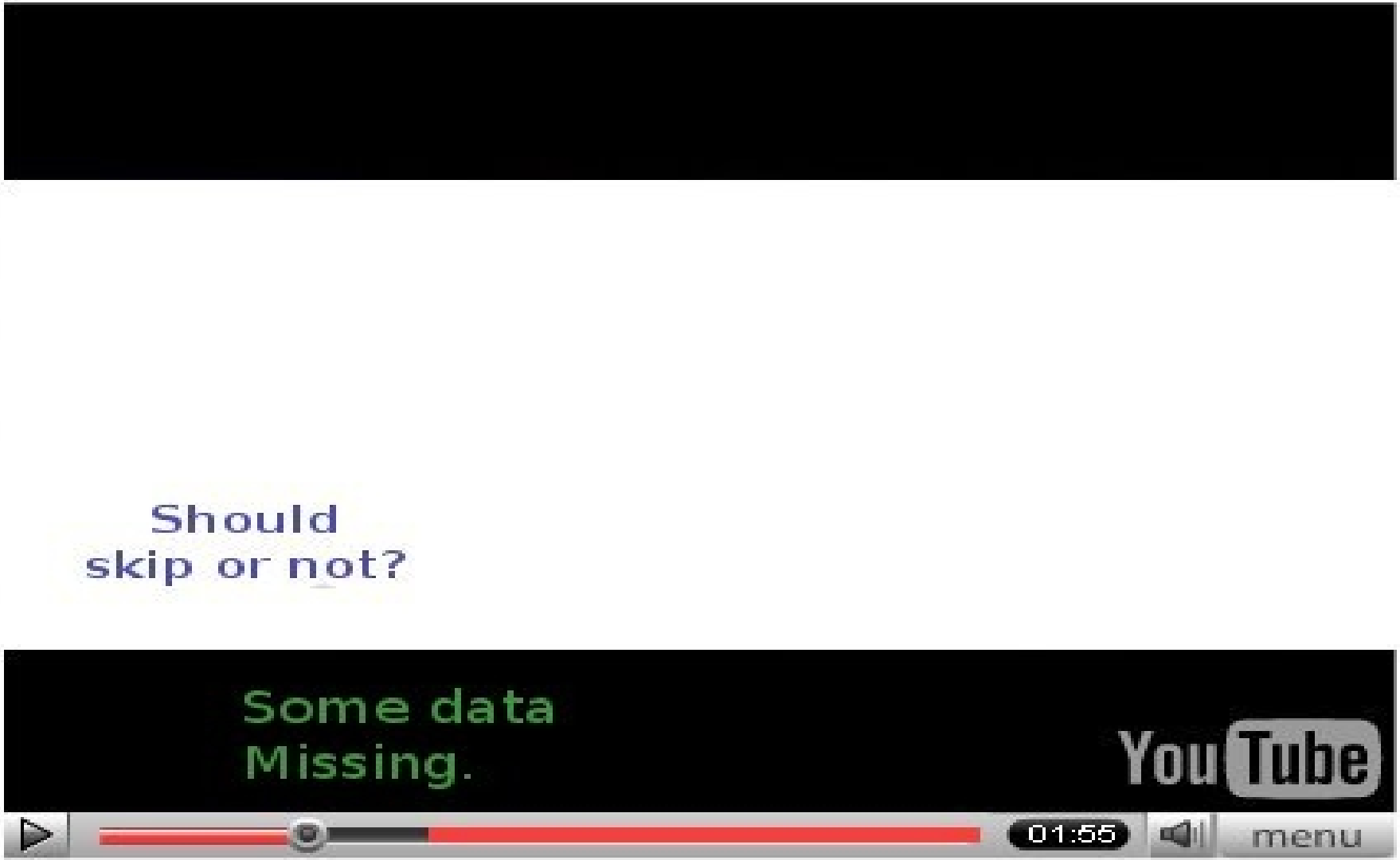
- Minion suite: a suite of protocols that uses legacy transports—UDP, TCP, and SSL—to provide a generic unordered datagram service between communicating end-points.
- Acts as a substrate on top of which more sophisticated communication primitives (such as partial-ordering) can be built and deployed.
- The minions are modified forms of the legacy transports where the protocols appear unmodified on the wire, thus making deployability through middleboxes possible. (Transparent)

Explaining the Problem

- TCP, which is a stream based protocol, delivers data in form of contigous chunks to higher layers.
- Even though the out-of-order data is available at lower layers, it is not delivered.
-



If user(appl layer) is given ability to decide!



New Layer Design

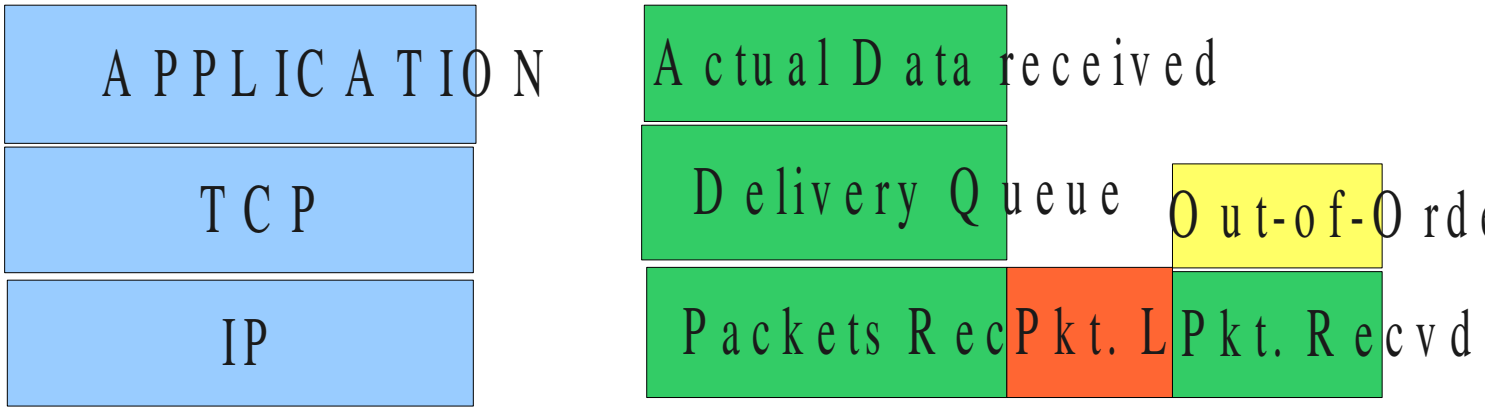


Fig. Current Architecture.

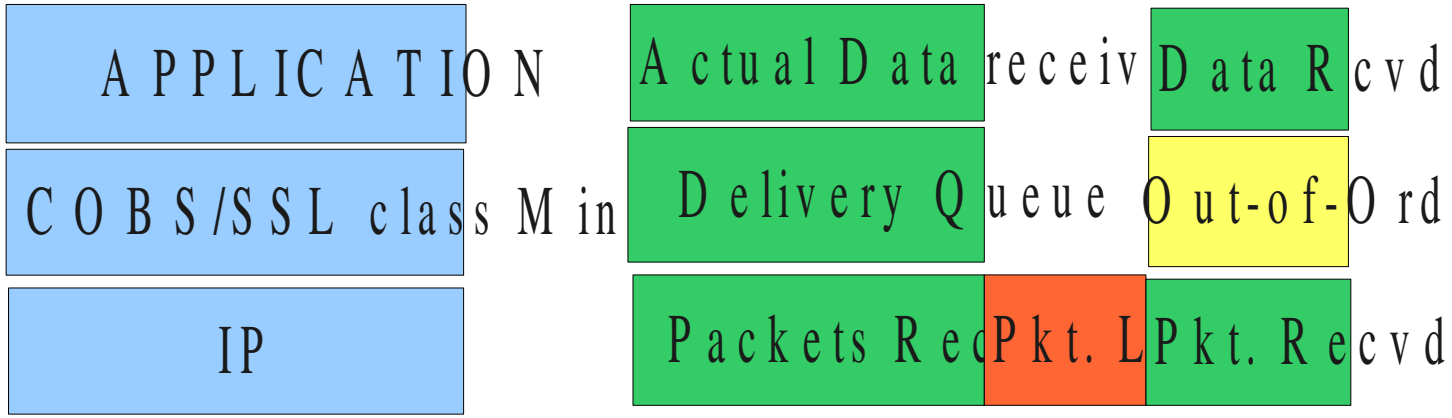


Fig. COBS/SSL Class Minion Architecture

COBS Minion

- One such attempt is Consistent Overhead Byte Stuffing (COBS) Minion.
- The COBS Minion ensures that the data is delivered to higher layers as soon as it is available to the lower layers.
- The task of data delivery is difficult because of numerous factors!
- Like Disordered packet arrival, packet loss, packet duplication, record extraction dependecies, packet concatenation and fragmentation.

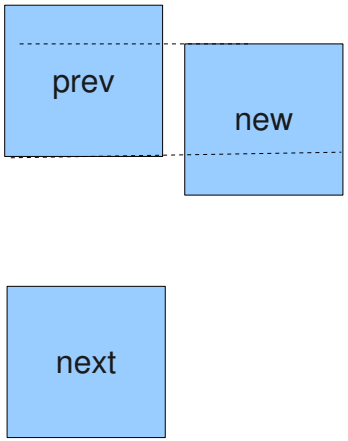
Packet Reordering

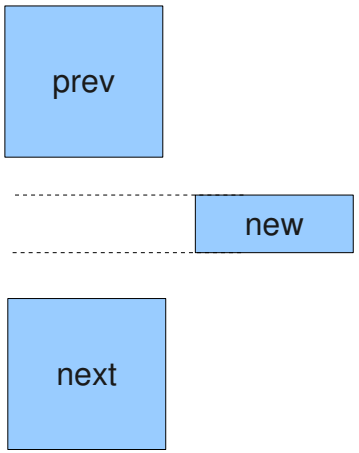
- We need to extract data out of even the reordered packet upon it's arrival. (No waiting)
- Specialized data structure designed for this purpose which maintains holes for the data unarrived.
- If higher layers have extracted complete info from that packet, the memory is released, but meta-data is maintained indicating succesful arrival for future use, like duplicate arrival.(Efficient Memory Management)

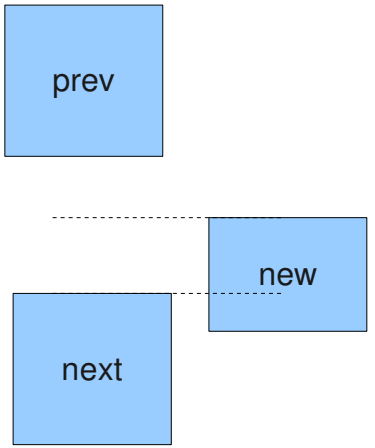
Unshaped Packet Arrival

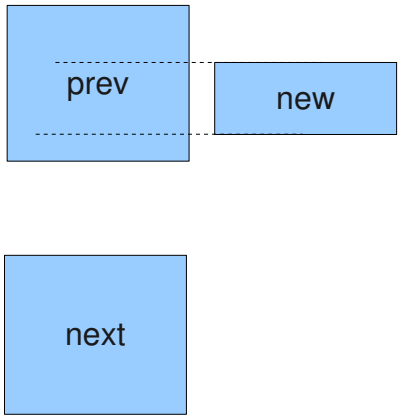
- Network can concatenate, duplicate or fragment the packets hence we need to get rid of such packets.
- But we use the data as soon as it arrives and release it after use, dealing with concatenated packets become difficult.
- Maintaing meta-data helps, imagine any virtual content based on meta-data.
- We need to extract only new data from the arrived packet!

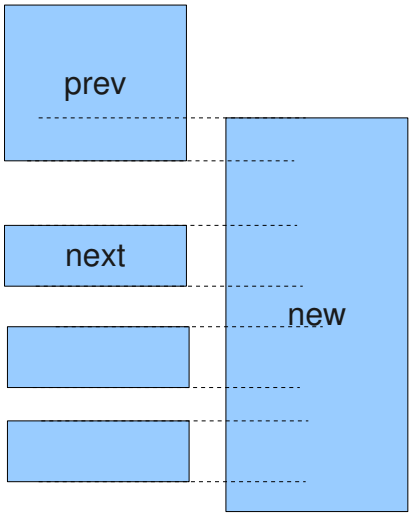
Some scenarios of packet manipulation in the network





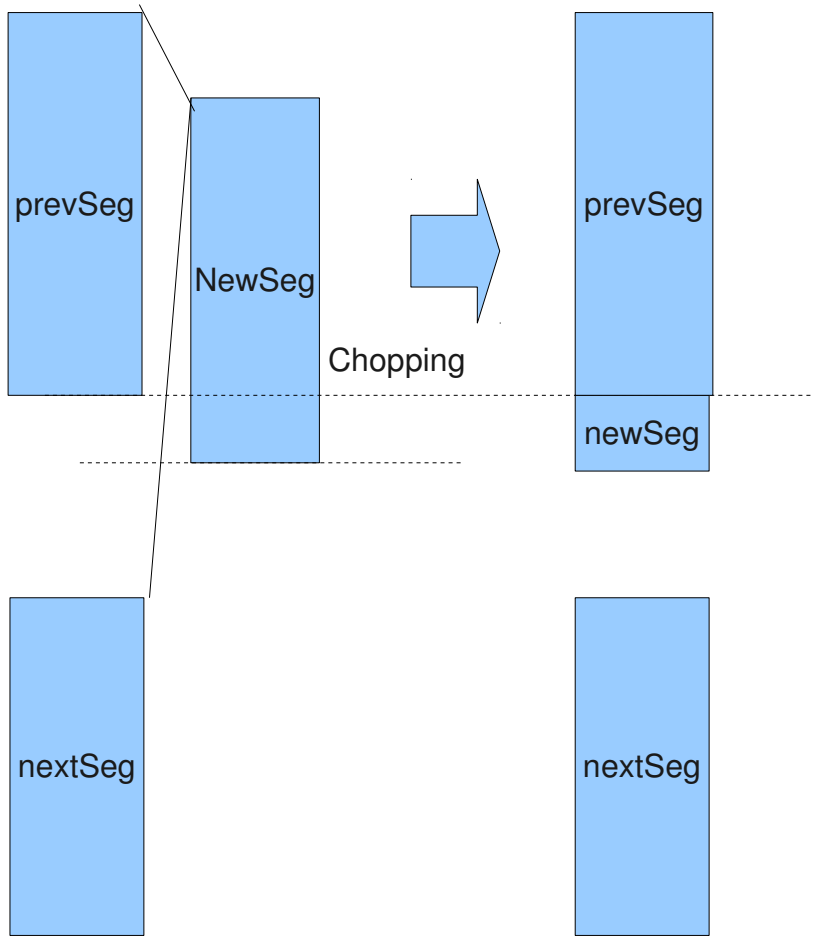






Extraction of new data from duplicates

- A new segment is formed.
- Whole process transparent to upper layer.
- Now we may proceed to extract ”record” from newly arrived segement.



Modifications to kernel for obtaining meta-data

Normal TCP

n=read(sd,buff,max);

Data of next higher sequence number returned in buffer.

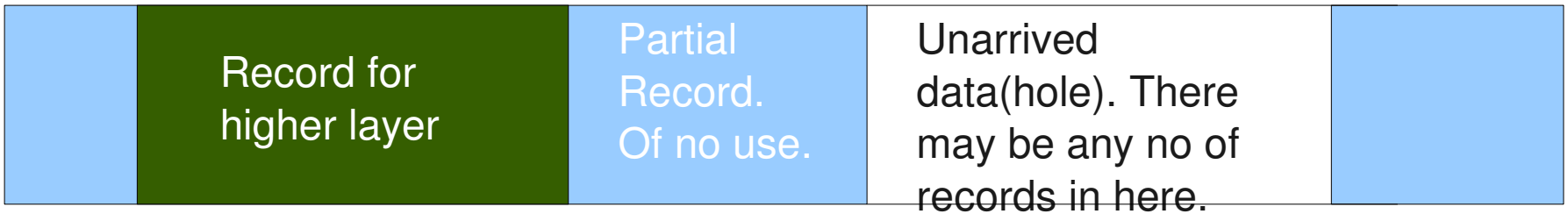
Data size=n

- Modified TCP

  n=read(sd,buff,max);
- Latest available TCP segment with 5 byte header in beginnig.
- 1st byte is flag.
- Next 4b seq. number.
- Data Size = n-5

What do we want to extract?

We need to extract ”records” which are lying inside the payload of arrived ”segment”. These records are to be delivered to higher layers. But We don't know where are they lying in a segment!!!!!!



COBS

We need some kind of ”marker” to indicate record starting.

This is provided by Consistent Overhead Byte Stuffing(COBS).

Removes all occurences of 0x00 from ”record” with a trade-off of some data size.(<0.4%)
0x00 can be used as marker byte.

Algorithm

Rules: 0x01 indicate 0 bytes followed by a 0x00.
0x02 indicate 1 byte followed by a 0x00.
0xfe indicate 253 bytes followed bya 0x00.
0xff indicate 254 bytes not followed by a 0x00.

Find 0x00 in data given.

Break into parts at point of occurrence of 0x00 or continous 255 bytes which ever is less.

Encode parts using rules.

Example

Data in hex.

12 23 00 --- 300 bytes --- 00 ff e4 78 00 89 00 00

03 12 23 ff --254 bytes -- 2e --46 bytes-- 04 ff e4 78 02 89 01

Extraction Algorithm

(1)Locate all markers in the new ”segment”.
(2)Extract the ”records” that are fully inside.
(3)For partial ”records” check if either previous or/and next segements have arrived.
(4)If so then, check if partial ”record” can be extracted from data of previous/next segement.
(5)The beginning of partial record can be in nth segment before current one. Check recursively
(6)Keep on updating log and meta-data as records are extracted.
(7)Release data memory if all records are extracted. Do recursively for all seg. in influence.

Advantages of COBS Minion

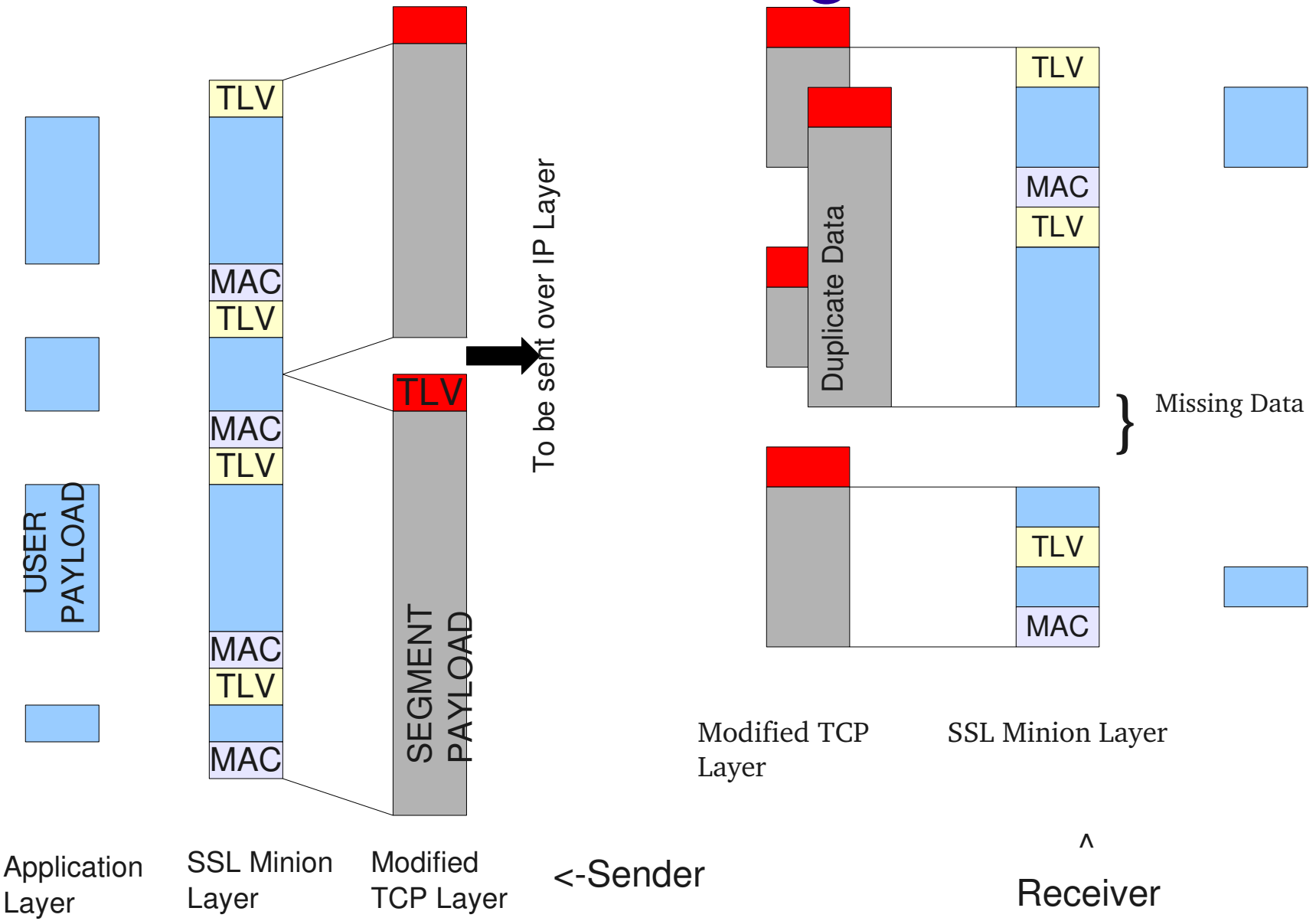
- It efficiently solves the problem of out of order data delivery to the higher layers.
- The encoding limits maximum overhead to 0.4%.
- Does not require any key exchange or handshake before trasfer.
- The process is reversible without any loss.

- Other class of minion can be based on MAC checking and doesnot involve use of markers!

The SSL Minion

- Uses TLS Message Authentication Codes for ascertaining the presence of a ”record”.
- Sender appends a header(TLV) in the beginning and MAC at the end.
- MAC = f(KEY, data);
- KEY is already exchanged using TLS Handshake Protocol.
- Standard TLS record layer header used with a record.

Process in figures.



Extraction in SSL Minion

(1)Guess and Verify Policy: Starting of record is guessed on basis of certain parameters like length field in TLV has to be less than MAX etc.

(2)Based on guessed header we obtain length of record. MAC must be after this.

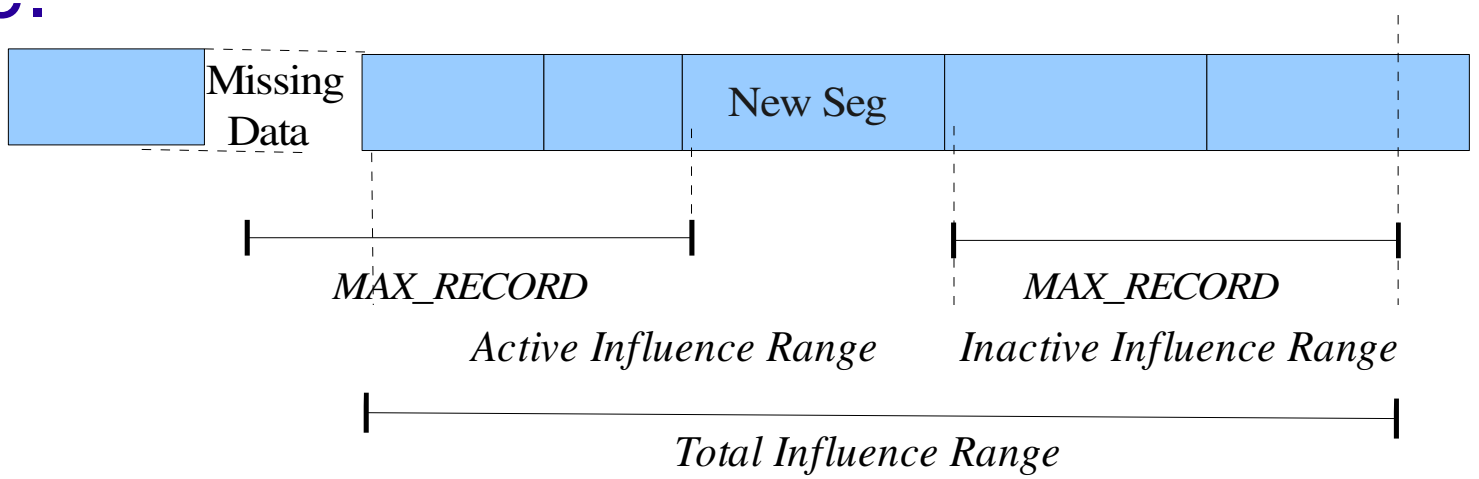
(3)MAC is calculated on this entire message. If it matches the MAC sent then we have identified the record.

(4)We again may have partial records and even worst partial headers extending to other seg.!

(5)We need to examine the influence range of the newly arrived segment.

Contd.

(6)All guesses which have not been verified in active influence range are checked.
(7)Log and meta-data to be updated if either a guess fails or record is found.
(8) Local optimizations.(If success, remove all guessed headers in it's range)
(9) Release memory when complete extraction done.



The Network Simulator

A simulator layer was developed to check the working of both the minions.

It was necessary as existing local simulators (eg. Netem or dummynet) do not simulate modified TCP kernel.

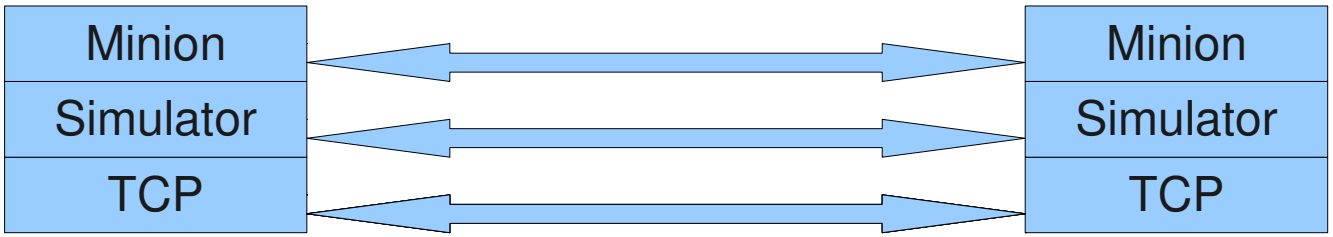
It can perform following function:

1) Duplication
2) Re-ordering
3)Concatenation

Both minions were successfully tested using the simulator.

contd.

Simulator layer is based on netem and works on either end point. (not in the network)
The degree of reordering, duplication etc. Can be easily controlled by the user.
Has to be installed on just both sides on top of TCP layer(normal).



Conclusion

- Both minions were succesfully tested for performance under varied load.
- The purpose of quick delivery of data(record) is achieved as the processing is done segment by segment.
- The minions provide quick delivery to higher layers but it is upto application on how to make use of the out-of-order data.

Future Work

- Work is to be done to develop Minion API which will integrate minion suite to higher layer.
- Applications which can make use of out of order data are not available and need to be develped.
- New naming schema is to be developed for internet browsing to accommodate minion suite.

References

(1)Bryan Ford, Janardan Iyegar ”Breaking up the Trasport Logjam”, HotNets-VII 2008.
(2)Stuart Cheshire, Mary Baker ”Consistent Overhead Byte Stuffing” IEEE/ACM Transactions on Networking (TON) , Vol 7, pg 159-172, 1999.
(3)J. Iyengar and B. Ford, ”A Next Generation Transport Services Architecture”. Internet-Draft draft-iyengar-ford-tng-00,IETF July 6, 2009.
(4)Bryan Ford and Janardhan Iyengar ”An Efficient Cross-Layer Negotiation Protocol”, . HotNets-VIII, October 2009.

Thanks To:

Prof. Bryan Ford for his constant guidance and support throughout the internship.

Students and staff at Yale university and F&M College for their support.

IIT Roorkee, for providing me the opportunity for the internship.

QUESTIONS?