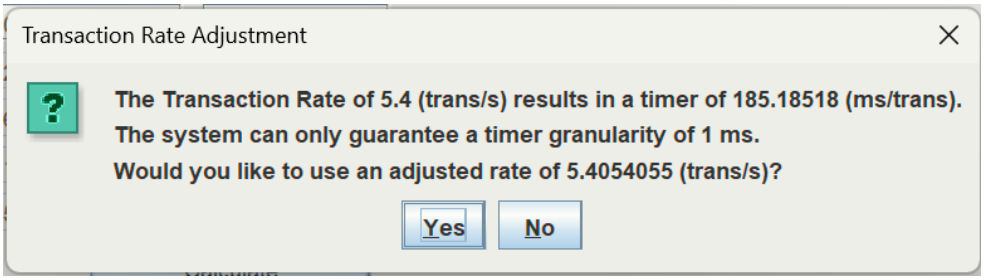
DMF Throughput Calculators

Throughput Calculators are provided to give a bigger picture view of bytes and packets involved in the DMF. With the calculators you can adjust the total throughput to your needs and calculate the transaction rate to reach it. You can also adjust the segment size to see how that affects the packet counts. In addition, depending upon the DMF type, different adjustments are possible.

The first two sections cover common items.

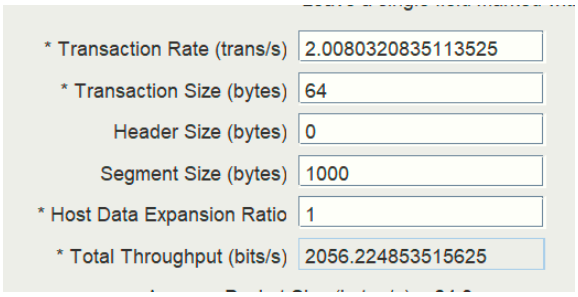
Transaction Rate Precision

When the transaction rate of a DMF is changed, you may see a dialog like this:



Due to the precision of our timers involved with DMF execution, sometimes the transaction rate will have to be adjusted slightly. To get 5.4 trans per second, Landslide would need a timer to fire 5.4 times per second. That would require a timer of 185.18518 ms/transaction and our timer can only do 185ms or 186ms. We would round it down to 185ms which makes the resulting transaction rate 5.4054055.

In the display it might show double precision but ultimately treated as a float value. The extra fraction digits can be ignored.



Or once a throughput is set, if there are too many fraction digits it might get a final adjustment to float precision when it is saved.

Control Packets and Header Size for Estimating Realistic Network

We have separated the packet and data on DMF calculators for Basic, Advanced and Lite DMFs. And we have provided “Number of Tx/Rx Control Packet per Transaction” fields to allow users to adjust the flow based on realistic network.

The calculator assumes that the size of all Tx/Rx Control Packets is Header Size.

And therefore, the existing value of Total Throughput is increased by: Header Size * (Number of Tx Control Packet + Number of Rx Control Packet).

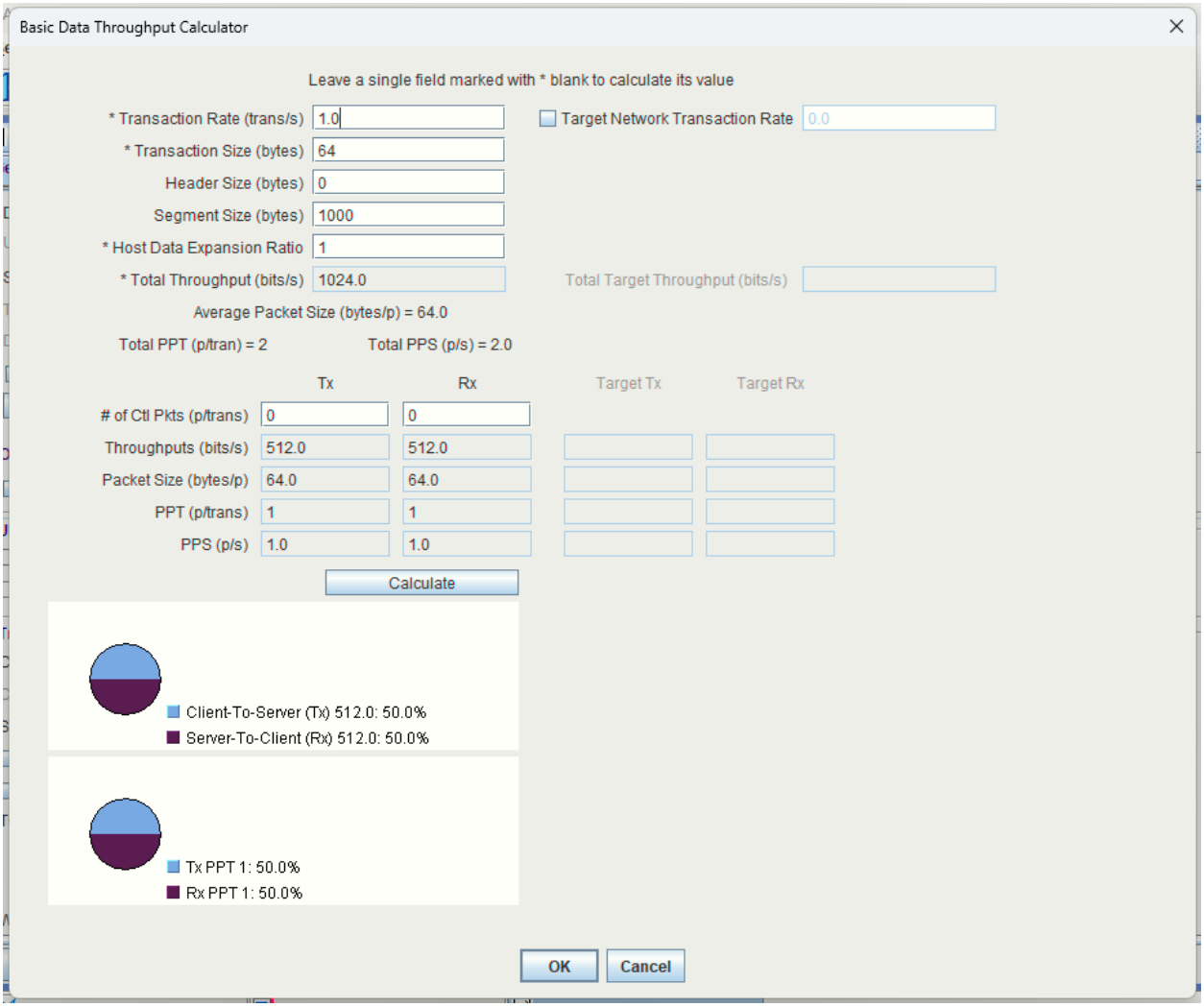
Also, the existing value of Total Packets per Transaction is increased by: Number of Tx Control Packet + Number of Rx Control Packet.

We have provided individual values of Tx and Rx on calculators, and you'll see the correlation between what you see on Traffic Mixer and them. We also add a new chart to indicate sent and received Packets per Transaction.

The Control Packets and Header Size have no impact on how Landslide executes the test, but they do reflect in the Traffic Mixer, to allow for more accurate predictions/estimates.

Basic Data Throughput Calculator

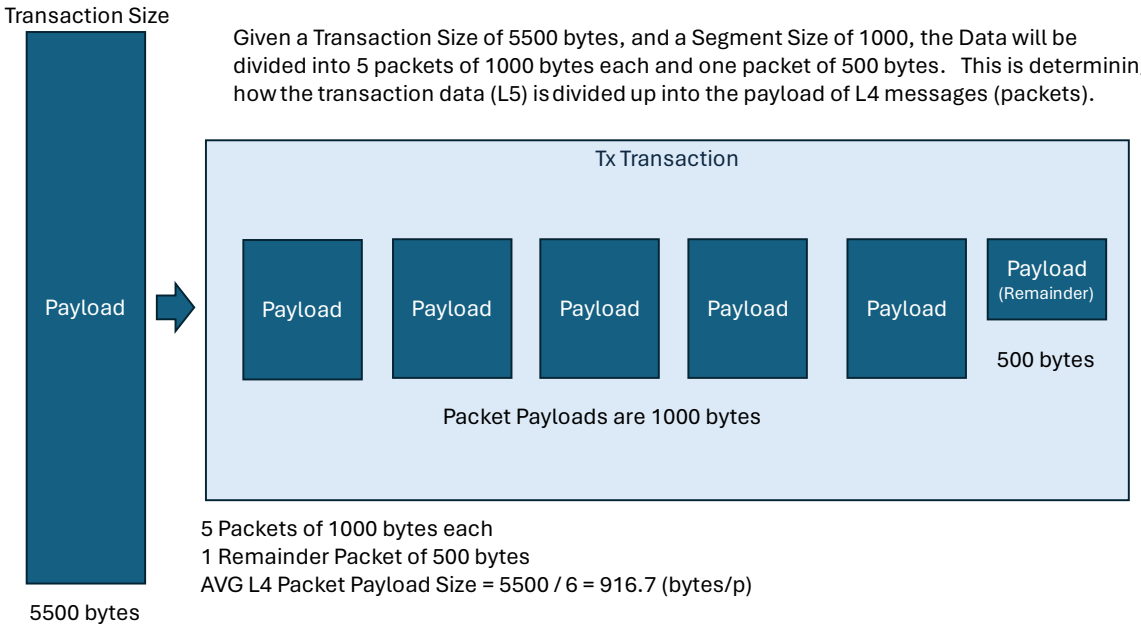
Used to help configure a Basic Data DMF, providing automatic calculation of the number of packets and throughputs and providing a way to include header size and extra control packets into the flow. While setting header size and control packets will update the values, the values set in this dialog do not have any effect on actual execution. They are provided here so that users can better estimate their numbers based on their own understanding of how L3 and L4 will operate.



Transaction Size sets the total bytes that will be sent from client to server (Tx) for each transaction of the DMF.

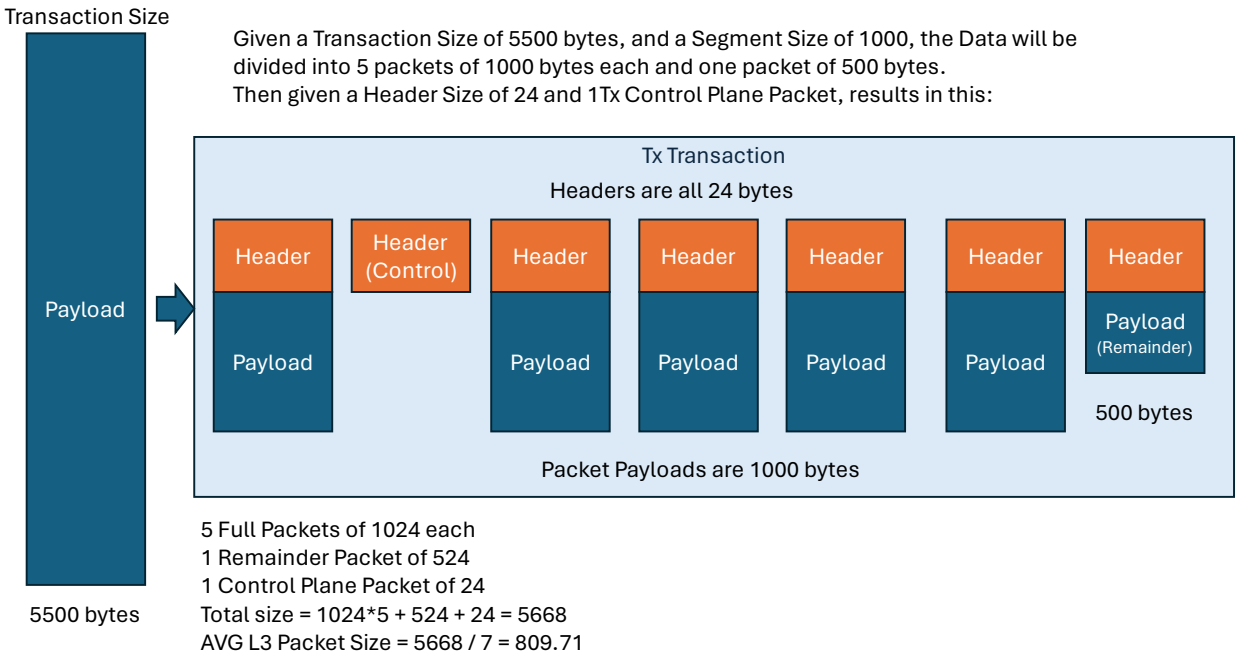
Target Network Transaction Size enables and sets an alternate transaction rate when using Mobility. When enabled the target network values will be calculated too. The only thing that can be changed on the target network is the transaction rate, ever other configurable property remains the same.

Segment Size sets the size of each segment/chunk/packet/message that Landslide will transport the data with. This determines how many packets it takes to transport the data (L4). Segment Size must be > 0 and Segment Size should also be adjusted based on the MTU size or else there will be fragmentation at the IP layer (L3).



If the Segment Size + L4 Header > MTU, these Packets will be further subdivided by IP Layer and this calculator does not take that into direct consideration, however Control Packets can be used to help account for a little.

Header Size sets how many bytes the calculator will add to each packet determined to be sent or received.

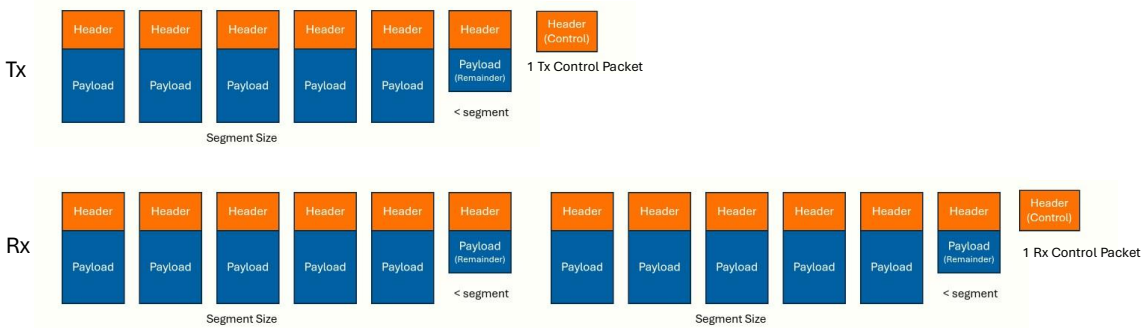


Host Data Expansion Ratio (HDE) sets how many times the defined traffic sent from client to server (Tx) will be sent from server to client or received (Rx) for each transaction of the DMF. Simply stated the Rx data will be Transaction Size * HDE, but more detailed, the { Rx Packets } = { Tx Packets } * HDE. The Control Packet configuration is separate and in addition to this.

# of Tx Control Packets and **# of Rx Control Packets** set how many extra packets to include in the calculations for Tx and Rx respectfully. These will be counted as containing Header Size bytes, so Header Size must be > 0 to use Control Packets.

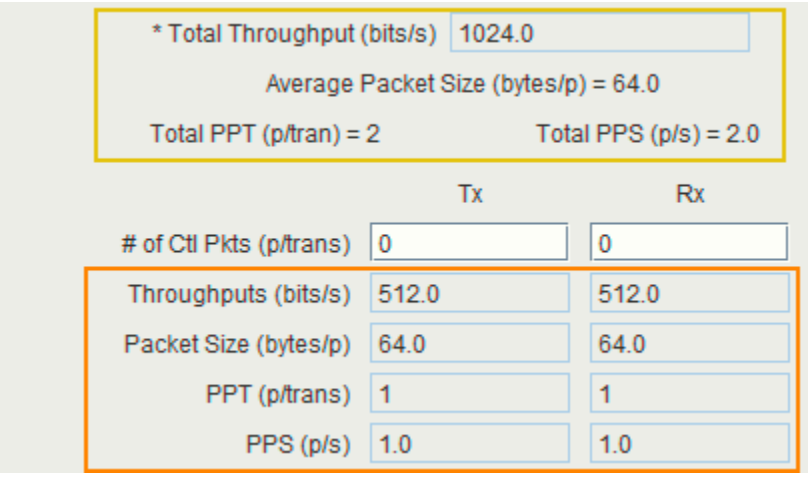
Host Data Expansion Ratio (HDE) determines the ratio of Tx to Rx, where Rx L4 Segments = Tx L4 Segments * HDE:
HDE=0 means no Rx
HDE=1 means Rx = Tx
HDE=2 means Rx = 2 x Tx

Given a configuration that results in a set of packets for Tx, those same packets are repeated multiple times for Rx and the AVG Packet Size for Tx and Rx should be the same unless there are individual Control Packets affecting the AVG . In the example below, HDE is set to 2 and Tx Control Packets/trans = 1 and Rx Control Packets/trans = 1.



When configuring Tx and Rx Control Packets in the calculator, keep in mind they are per-transaction for Rx too, if you want an even ratio, you will need to set Rx Control Packets = Tx Control Packets * HDE on your own to have equal number of Control Packets per set of Payload Packets Example above Rx Control Packets is set to 1, so there will be 1 Control Packet for two sets of the Tx Packets, changing the AVG Packet Size for Rx compared to Tx.

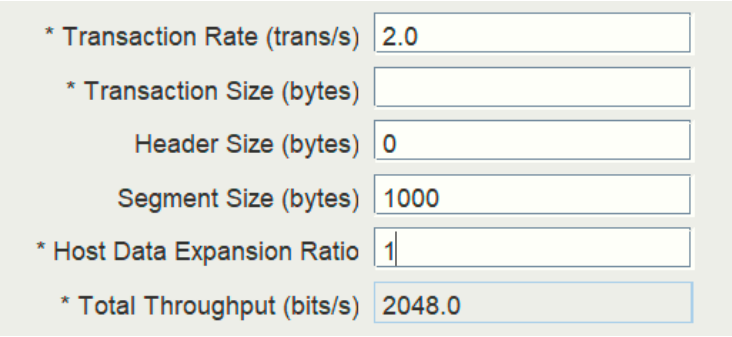
Purely calculated fields:



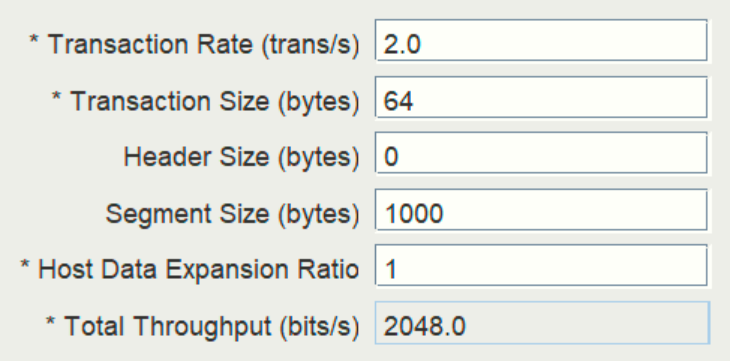
Top section is total bi-directional details, e.g. Average Packet Size (bytes/p) shows the Average Packet Size in both directions.
Bottom orange section is separated for each direction, e.g. Packet Size (bytes/p) shows the Average Packet Size in the given direction.

Any field with * can be left blank and the calculator will notice it and calculate its value. But you cannot have more than one blank field.

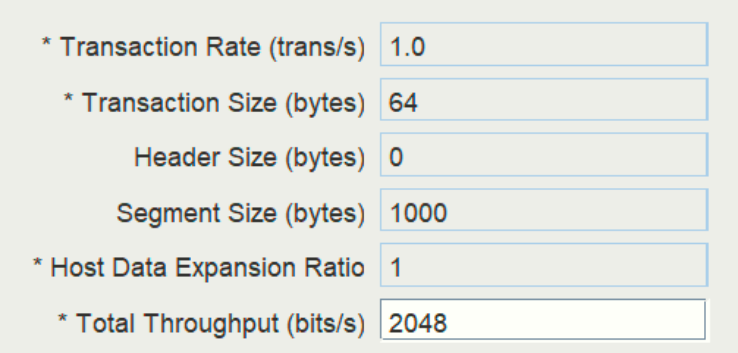
For example, we fill in other fields and leave Transaction Size blank:



Then click Calculate and the calculator will bring you, its value.

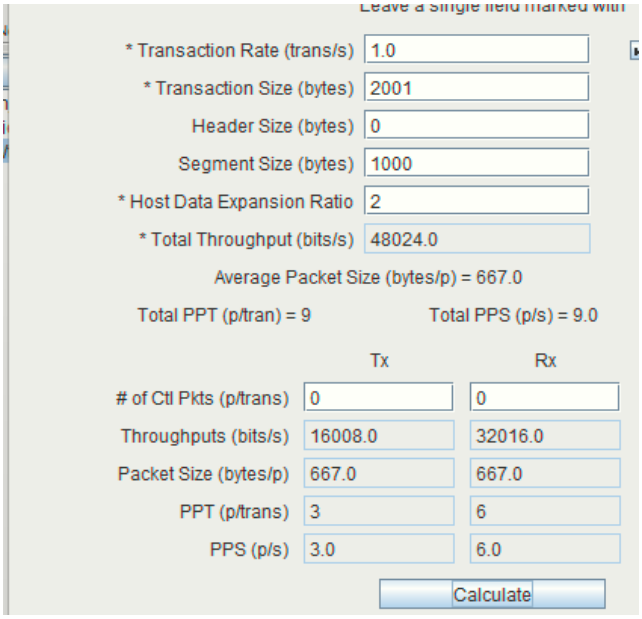


And there is a Throughput editing mode, when you double click in Total Throughput it will become enabled and allow you to enter the throughput (bidirectional) that you want to reach, then click Calculate button and the Transaction Rate will be adjusted to meet that Throughput.

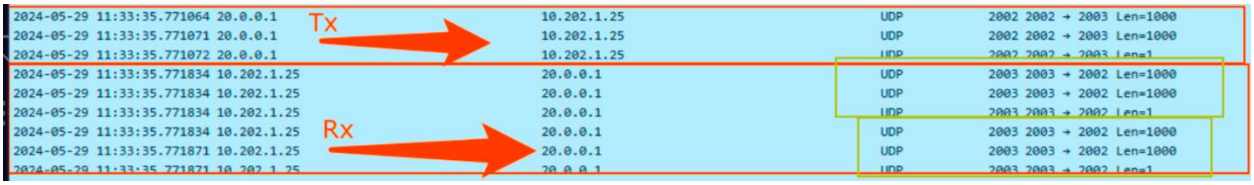


You must click Calculate or click Enter key to exit this mode.

Example with HDE and adjusting for Headers



Given the settings above to send twice as much Rx as Tx, when executed will produce results like the PCAP below.

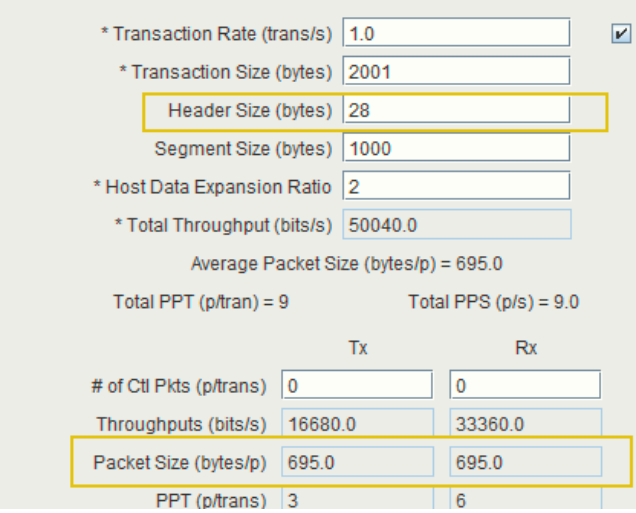


PCAP shows that the single Tx Transaction has 2 1000 Byte segments and one 1 Byte Segment, and the Rx repeats the same sequence of packets twice.

On the L3 Client measurements tab, the AVG Packet Size is the same for both Tx and Rx, but the Bytes, Packets and Throughput are double for Rx.

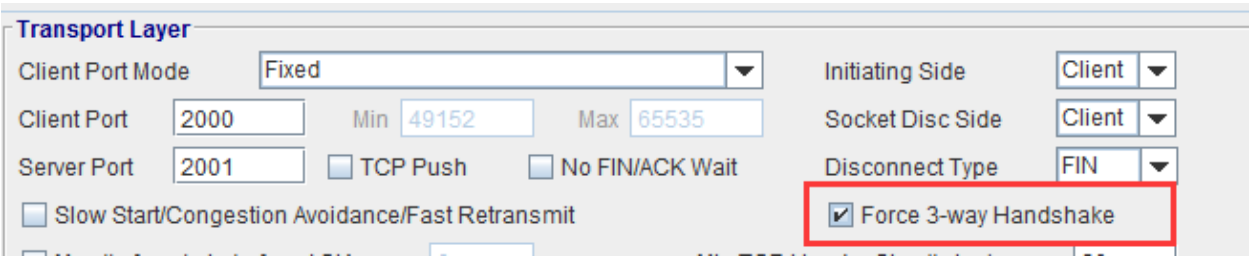
AA	AB	AD	AE	AG	AH	AI	AJ	AK
Total Bits Sent/Sec	Total Bits Sent/Sec (P-I)	Total Bits Received/Sec	Total Bits Received/Sec (P-I)	Total Bits/Sec	Total Bits/Sec (P-I)	Average Packet Size (B/p)	Average Sent Packet Size (B/p)	Average Received Packet Size (B/p)
0	0	0	0	0	0	695	695	695
3508.993373	3508.993373	7017.986747	7017.986747	10526.98012	10526.98012	695	695	695
3508.993373	0	7017.986747	0	10526.98012	0	695	695	695

Also note that L3 Client measurements show 695 bytes/p, that is due to the header 8 for UDP and 20 for IPv4. And this can be configured in the DMF Calculator to reflect that ahead of time:



Example with Control Packets

If Force 3-way Handshake is checked that will include extra Control Packets in the execution.



In the calculator, set # of Tx Control Packets to 4 and # of Rx Control Packets to 2. And set Header Size to 58, which includes MAC, IP, TCP and CRC. For a DMF with just 64 bytes per transaction, Segment Size of 1000 and HDE=1, that will be just 1 packet with 64 bytes of payload in each direction.

Tx/Rx Throughput: 8 bits/byte * (payload bytes + header bytes + (#ctlPackets * header bytes) * trans rate trans/s

$$8 * (64 + 58 + 4 * 58) * 1.0 = 2832.0\ bits/s$$

$$8 * (64 + 58 + 2 * 58) * 1.0 = 1904.0\ bits/s$$

And Tx PPT is 5 (1 + 4 control). Rx PPT is 3 (1 + 2 control).

Then all information goes into Traffic Mixer

Inst #	DMF Library/N	DMF Protoc	User-[Mix-	Rate (trans/	Tx PPT (p/trans)	Tx PPS (p/s)	Tx (B/trans	Tx Throughp	Tx Avg Packet	Tx %	Rx PPT (p/trans)	Rx PPS (p/s)	Rx (B/trans	Rx Throughp	Rx Avg Packet	Rx %	Ratio Tx/Rx
TC	*Test C...*			1.0		5.0		2.765	70.8			3.0		1.859	79.333		*60/40*
MN	*Per-M...*			1.0		5.0		2.765	70.8			3.0		1.859	79.333		*60/40*
0-0	*sms/B...*	*tcp*	tcp	1.0	5.0	5.0	*354*	2.765	70.8	100.0	3.0	3.0	*238*	1.859	79.333	100.0	*60/40*

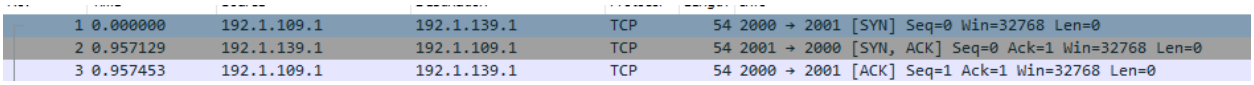
The Throughput on Traffic Mixer is in kbps, so Tx Throughput is:

$$2832.0 \div 1024 = 2.765\ kbps$$

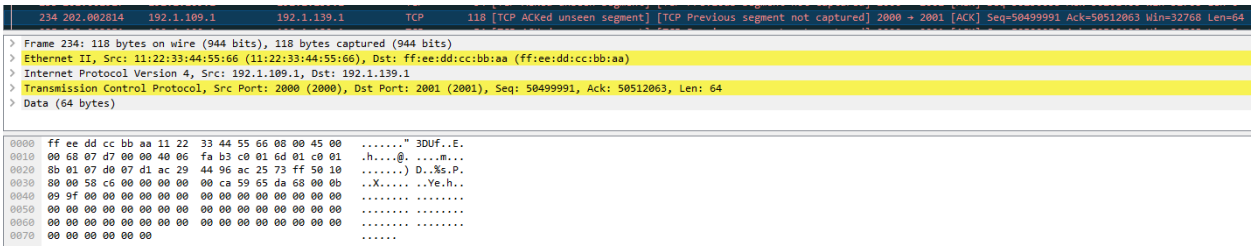
In Test Reports, you'll see Packets per Second and Bits per Second. Then you'll find the correlation between what you see on the Calculator, Traffic Mixer and Test Reports:

_3 Client::Total Packets Sent/Sec (P-I)	5	4.999	4.999	5.001
_3 Client::Total Packets Received/Sec (P-I)	3	2.999	2.999	3.001
_3 Client::Total Bits Sent/Sec (P-I)	2,831.811	2,831.434	2,831.245	2,832.755
_3 Client::Total Bits Received/Sec (P-I)	1,903.873	1,903.619	1,903.492	1,904.508

You can find everything in Port Capture. Here are control packets:



And the packet whose size is 64 bytes.

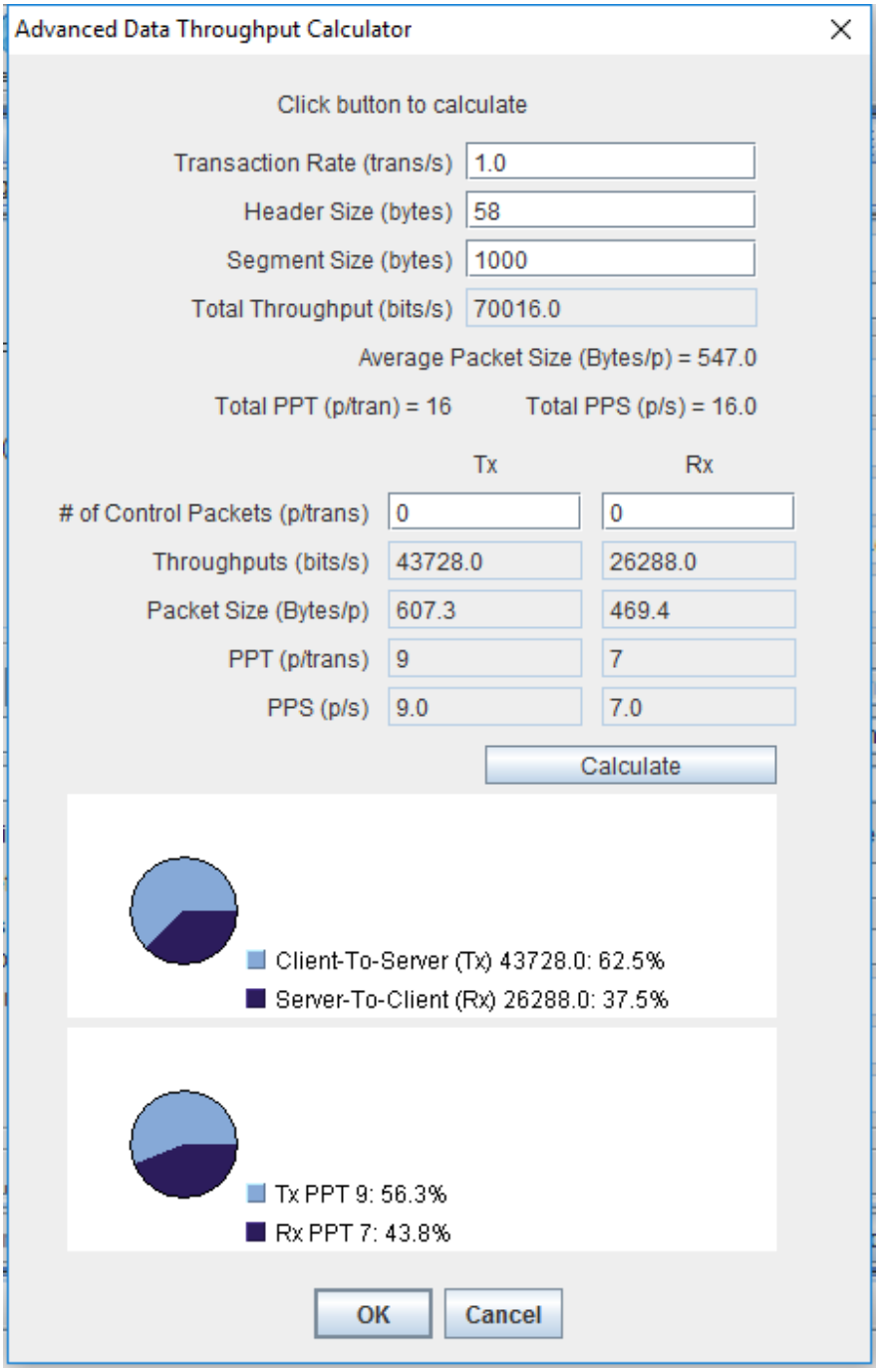


Advanced Data Throughput Calculator

Advanced Data DMFs have a predefined list of messages exchanged and so there are fewer fields to adjust. The focus is on Segment Size, Header Size and Control Packets, determining the final sequence of Packets involved. The Advanced Data calculator can build the packets based on the separate messages; each message would be independently divided into packets with the final packet in each message possibly being a remainder packet with less than Segment Size of Payload Data (See Basic Data Calculator for how Segment Size determines Packets).

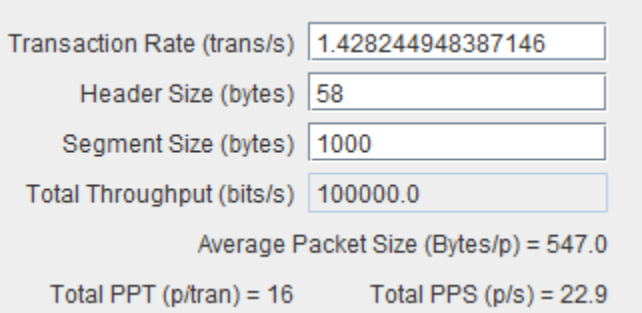
Segment Size sets the size of each segment/chunk/packet/message that Landslide will transport the data with. This determines how many packets it takes to transport the data (L4). Segment Size must be > 0 and Segment Size should also be adjusted based on the MTU size or else there will be fragmentation at the IP layer (L3).
Header Size sets how many bytes the calculator will add to each packet determined to be sent or received.
# of Tx Control Packets and **# of Rx Control Packets** set how many extra packets to include in the calculations for Tx and Rx respectfully. These will be counted as containing Header Size bytes, so Header Size must be > 0 to use Control Packets.

User Case 1: Advanced HTTP DMF with 5 messages using 4 TDFs

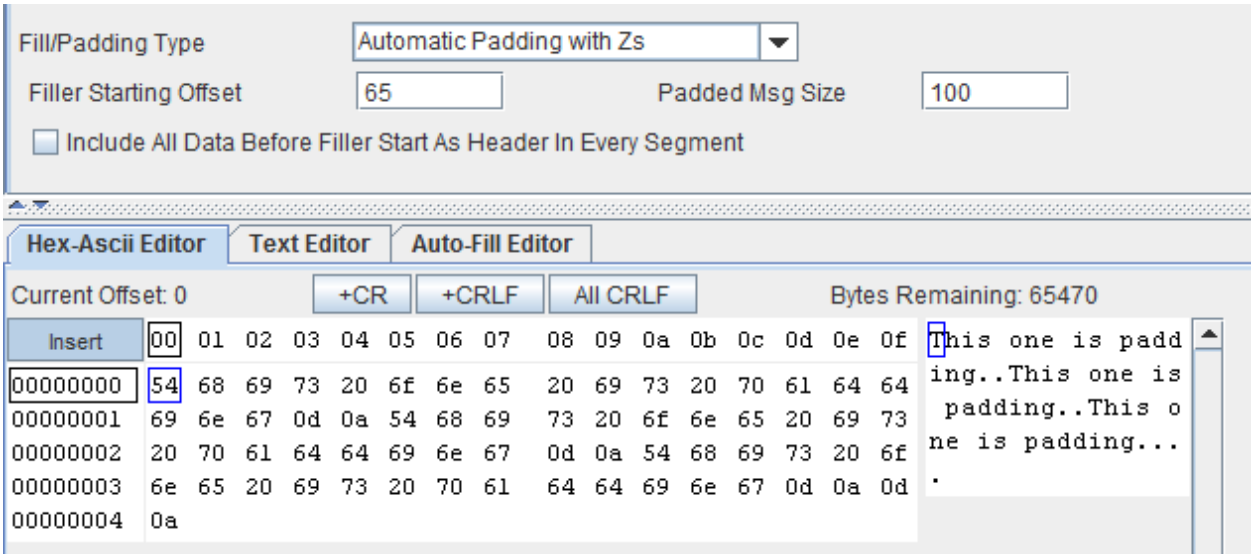


This is a HTTP DMF so I don't need # of Control Packets. Just set them to 0. Set Header Size to 58, which includes MAC, IP, TCP and CRC.

If I click and re-set the Total Throughput to 100000, it will calculate a new Transaction Rate for me.



In this case, there are 3 TDFS and one using Padding:



For a better display, I set Transaction Rate back to 1. And you'll see all above in Traffic Mixer:

Inst #	DMI Library/I	DMF Protoc	User-I Mix-	Rate (trans/	Tx PPT (p/trans)	Tx PPS (p/s)	Tx (B/trans	Tx Throughp	Tx Avg Packet	Tx %	Rx PPT (p/trans)	Rx PPS (p/s)	Rx (B/trans	Rx Throughp	Rx Avg Packet	Rx %	Ratio Tx/Rx
TC	*Test C...*			1.0		9.0		42.703	607.3			7.0		25.671	469.4		*62/38*
MN	*Per-M...*			1.0		9.0		42.703	607.3			7.0		25.671	469.4		*62/38*
0-0	*sms/H...*	*http*	http	1.0	9.0	9.0	*5466*	42.703	607.3	100.0	7.0	7.0	*3286*	25.671	469.4	100.0	*62/38*

Here are test reports:

L3 Client::Total Packets Sent/Sec (P-I)	10	10.001	10	10
L3 Client::Total Packets Received/Sec (P-I)	7	7	7	7
L3 Client::Total Bits Sent/Sec (P-I)	44,192	44,194.946	44,192	44,192
L3 Client::Total Bits Received/Sec (P-I)	26,288	26,289.753	26,288	26,288

This is the first TDF which contains padding and we can find it in Port Capture.

Pkt #	Direction	Time(ms)	Delay(ms)	Data
1	→	0	0	This one is padding..This one is padding..This one is padding....
2	←	0	0	HTTP/1.1 200 OK..Date: Mon, 25 Oct 2004 23:59:59 GMT..Content-Type: text/ht...
3	→	0	0	GET sampleFile.html HTTP/1.1..Host: www.texaslotto.com..User-Agent: xPTS/2...
4	←	0	0	HTTP/1.1 200 OK..Date: Mon, 25 Oct 2004 23:59:59 GMT..Content-Type: text/ht...
5	→	0	0	
6	←	0	0	HTTP/1.1 200 OK..Date: Mon, 25 Oct 2004 23:59:59 GMT..Content-Type: text/ht...
7	→	0	0	GET sampleFile.html HTTP/1.1..Host: www.texaslotto.com..User-Agent: xPTS/2...
8	←	0	0	HTTP/1.1 200 OK..Date: Mon, 25 Oct 2004 23:59:59 GMT..Content-Type: text/ht...
9	→	0	0	GET sampleFile.html HTTP/1.1..Host: www.texaslotto.com..User-Agent: xPTS/2...
10	←	0	0	HTTP/1.1 200 OK..Date: Mon, 25 Oct 2004 23:59:59 GMT..Content-Type: text/ht...

Hex/Ascii | Text

Insert | Bytes Left: 65470

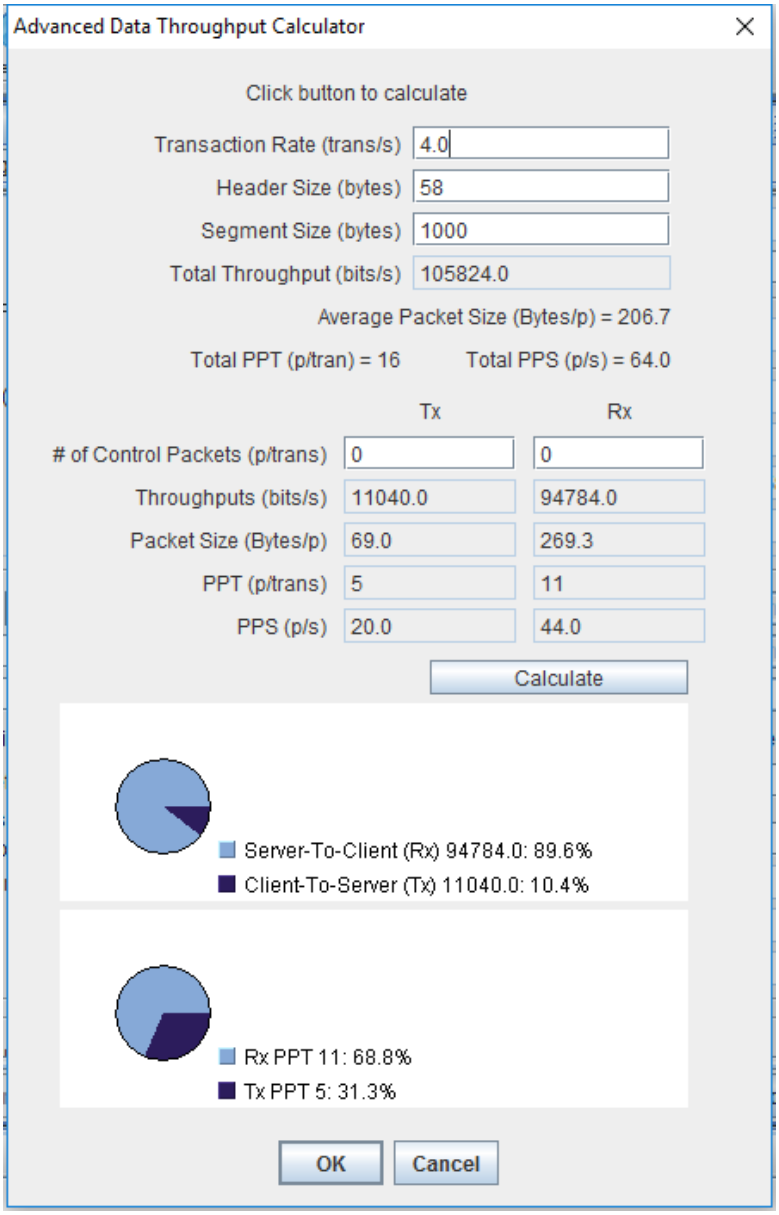
```
          00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f
00000000  54 68 69 73 20 6f 6e 65  20 69 73 20 70 61 64 64  This one is padd
00000001  69 6e 67 0d 0a 54 68 69  73 20 6f 6e 65 20 69 73  ing..This one is
00000002  20 70 61 64 64 69 6e 67  0d 0a 54 68 69 73 20 6f   padding..This o
00000003  6e 65 20 69 73 20 70 61  64 64 69 6e 67 0d 0a 0d  ne is padding...
00000004  0a                                                .
```

```
7 2.091026   192.1.109.1   192.1.139.1   TCP   154 2000 → 80 [ACK] Seq=9889 Ack=5876 Win=32768 Len=100
```

```
> Frame 7: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits)
> Ethernet II, Src: 11:22:33:44:55:66 (11:22:33:44:55:66), Dst: ff:ee:dd:cc:bb:aa (ff:ee:dd:cc:bb:aa)
> Internet Protocol Version 4, Src: 192.1.109.1, Dst: 192.1.139.1
> Transmission Control Protocol, Src Port: 2000 (2000), Dst Port: 80 (80), Seq: 9889, Ack: 5876, Len: 100
```

```
0000  ff ee dd cc bb aa 11 22  33 44 55 66 08 00 45 00   ......." 3DUf..E.
0010  00 8c 04 07 00 00 40 06  fe 5f c0 01 6d 01 c0 01   ......@. ._..m...
0020  8b 01 07 d0 00 50 81 4a  e7 0d 81 43 39 55 50 10   .....P.J ...C9UP.
0030  80 00 e7 6b 00 00 54 68  69 73 20 6f 6e 65 20 69   ...k..Th is one i
0040  73 20 70 61 64 64 69 6e  67 0d 0a 54 68 69 73 20   s paddin g..This
0050  6f 6e 65 20 69 73 20 70  61 64 64 69 6e 67 0d 0a   one is p adding..
0060  54 68 69 73 20 6f 6e 65  20 69 73 20 70 61 64 64   This one  is padd
0070  69 6e 67 0d 0a 0d 0a 5a  5a 5a 5a 5a 5a 5a 5a 5a   ing....Z ZZZZZZZZ
0080  5a 5a 5a 5a 5a 5a 5a 5a  5a 5a 5a 5a 5a 5a 5a 5a   ZZZZZZZZ ZZZZZZZZ
0090  5a 5a 5a 5a 5a 5a 5a 5a  5a 5a                    ZZZZZZZZ ZZ
```

User case 2: FTP Mainflow + 1 Subflow with a TDF

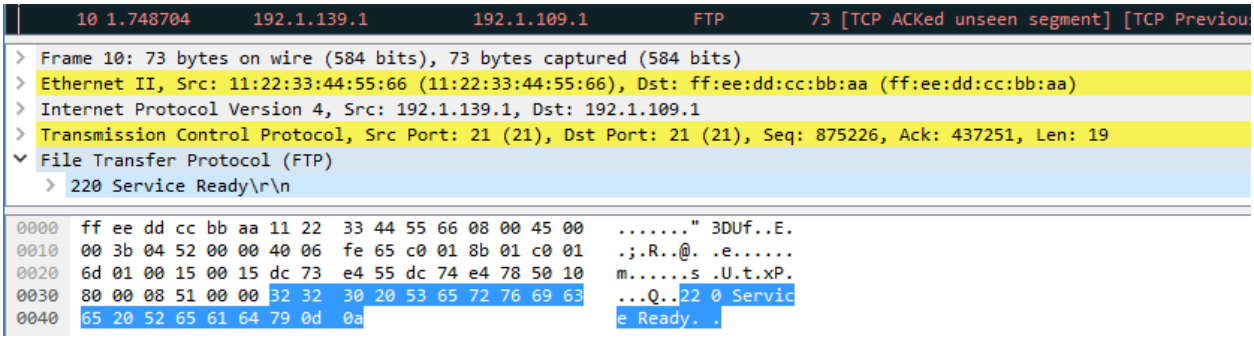


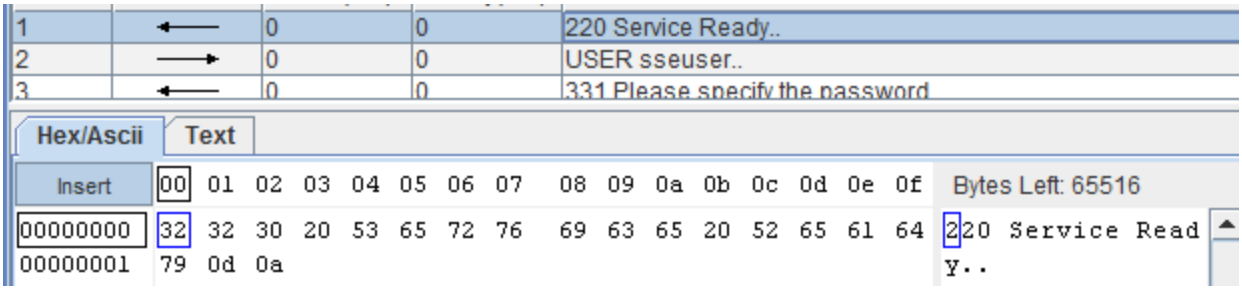
This case contains only one TDF which is in a response message of Subflow.

Test reports:

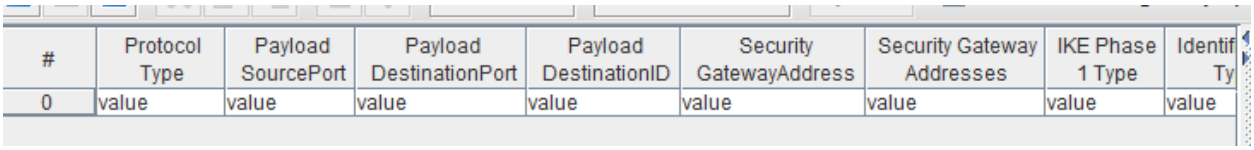
L3 Client::Total Packets Sent/Sec (P-I)	21.468	32.067	20	19.014
L3 Client::Total Packets Received/Sec (P-I)	29.202	43.733	27.2	25.876
L3 Client::Total Bits Sent/Sec (P-I)	11,090.339	16,564.267	10,331.2	9,824.16
L3 Client::Total Bits Received/Sec (P-I)	53,085.139	80,207.467	49,427.2	47,085.061

You'll find message data in FTP packets.

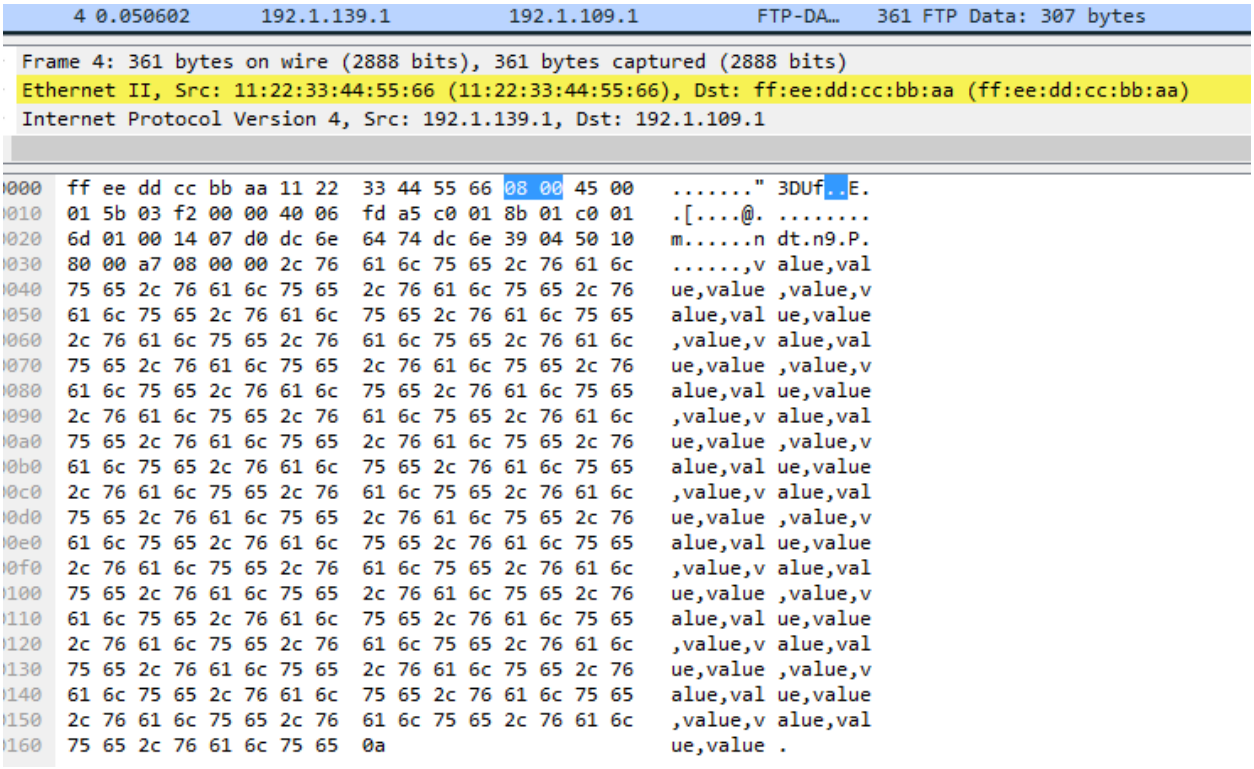




This is the TDF in the Subflow:

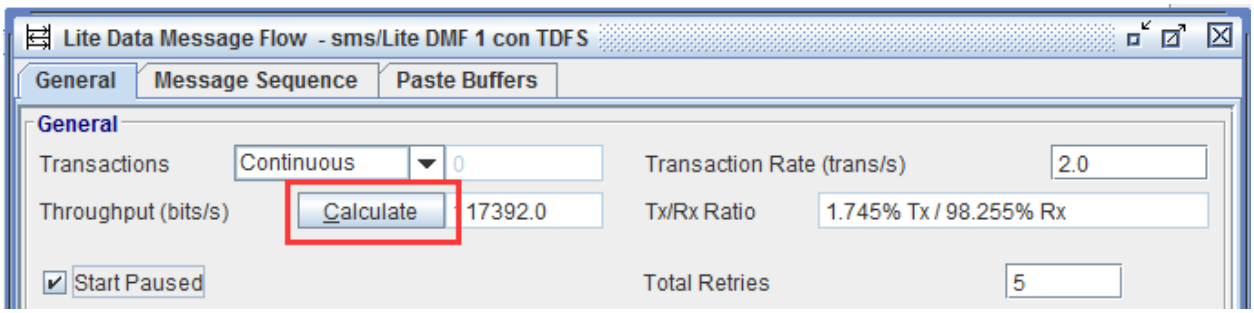


#	Protocol Type	Payload SourcePort	Payload DestinationPort	Payload DestinationID	Security GatewayAddress	Security Gateway Addresses	IKE Phase 1 Type	Identif Ty
0	value	value	value	value	value	value	value	value

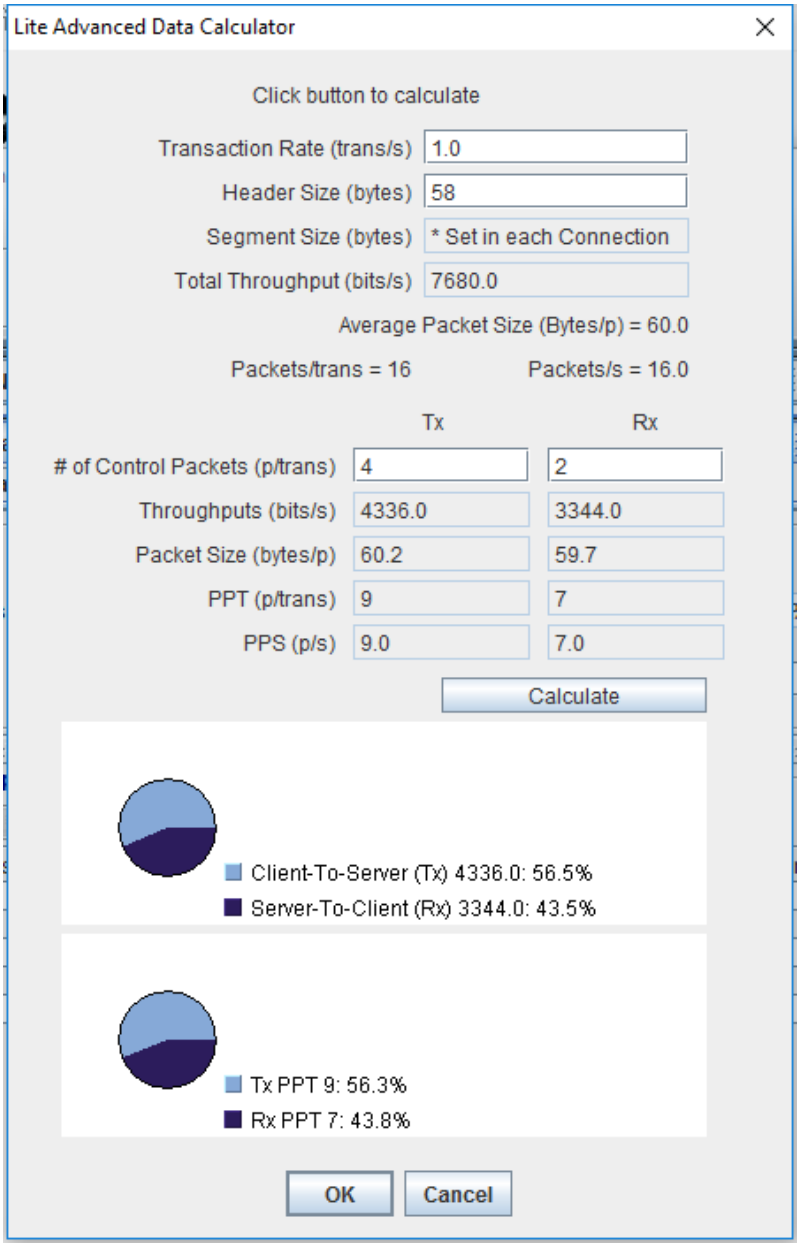


Lite Advanced Data Calculator

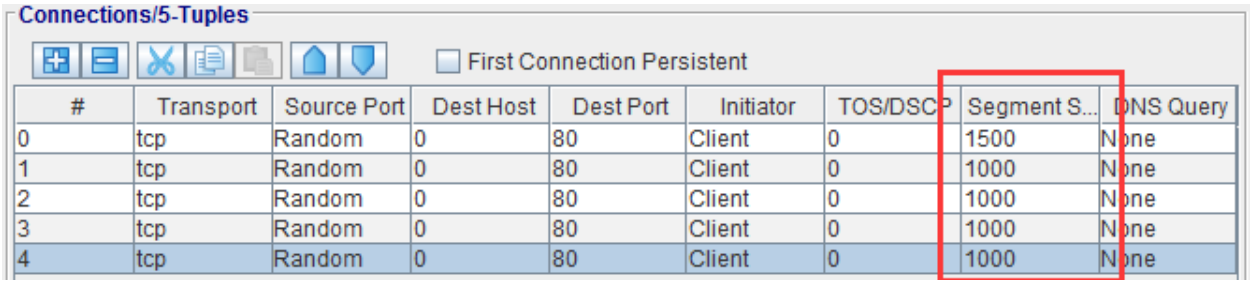
Lite DMFs ultimately are executed as Advanced Data DMFs, but the calculation and display of Throughput and Transaction Rate is different.



User case: Lite DMF with 5 Connections, 2 messages REQ/RSP per connection.



Segment Size is set in each Connection.



#	Transport	Source Port	Dest Host	Dest Port	Initiator	TOS/DSCP	Segment S...	DNS Query
0	tcp	Random	0	80	Client	0	1500	None
1	tcp	Random	0	80	Client	0	1000	None
2	tcp	Random	0	80	Client	0	1000	None
3	tcp	Random	0	80	Client	0	1000	None
4	tcp	Random	0	80	Client	0	1000	None

In Traffic Mixer:

Inst #	DMF Library/M	DMF Protoc	User-D Mix-	Rate (trans/	Tx PPT (p/trans)	Tx PPS (p/s)	Tx (B/trans)	Tx Throughp	Tx Avg Packet	Tx %	Rx PPT (p/trans)	Rx PPS (p/s)	Rx (B/trans	Rx Throughp	Rx Avg Packet	Rx %	Ratio Tx/Rx
TC	*Test C...*			1.0		9.0		4.234	60.222			7.0		3.265	59.714		*56/44*
MN	*Per-M...*			1.0		9.0		4.234	60.222			7.0		3.265	59.714		*56/44*
0-0	*sms/Li...*	*lite*	lite	1.0	9.0	9.0	*542*	4.234	60.222	100.0	7.0	7.0	*418*	3.265	59.714	100.0	*56/44*

Test reports:

L3 Client::Total Packets Sent/Sec (P-I)	0.267	0.267	0.6
L3 Client::Total Packets Received/Sec (P-I)	0.133	0.133	0.333
L3 Client::Total Bits Sent/Sec (P-I)	123.725	123.733	278.933
L3 Client::Total Bits Received/Sec (P-I)	61.863	61.867	158.933

PPS and Throughputs are very small. That's because connections were happening slowly. You'll find everything accurate in the next case.

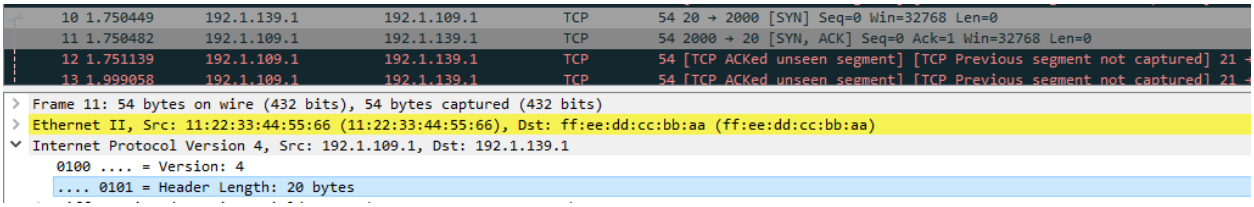
No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.1.139.1	192.1.109.1	TCP	54	80 → 62599 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
2	0.499548	192.1.109.1	192.1.139.1	TCP	54	62599 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
3	17.999972	192.1.139.1	192.1.109.1	TCP	54	80 → 64914 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
4	18.499480	192.1.109.1	192.1.139.1	TCP	54	64914 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
5	35.999944	192.1.139.1	192.1.109.1	TCP	54	80 → 64173 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
6	36.499479	192.1.109.1	192.1.139.1	TCP	54	64173 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
7	53.999957	192.1.139.1	192.1.109.1	TCP	54	80 → 63610 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
8	54.499478	192.1.109.1	192.1.139.1	TCP	54	63610 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
9	72.000053	192.1.109.1	192.1.139.1	TCP	55	56373 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=1
10	72.000321	192.1.139.1	192.1.109.1	TCP	54	80 → 63691 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
11	72.499526	192.1.109.1	192.1.139.1	TCP	54	63691 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
12	89.999982	192.1.139.1	192.1.109.1	TCP	54	80 → 61603 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
13	90.499478	192.1.109.1	192.1.139.1	TCP	54	61603 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
14	108.000012	192.1.139.1	192.1.109.1	TCP	54	80 → 51424 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
15	108.499478	192.1.109.1	192.1.139.1	TCP	54	51424 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
16	125.999960	192.1.139.1	192.1.109.1	TCP	54	80 → 53521 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
17	126.499483	192.1.109.1	192.1.139.1	TCP	54	53521 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
18	144.000364	192.1.139.1	192.1.109.1	TCP	54	80 → 54748 [SYN, ACK] Seq=0 Ack=1 Win=32768 Len=0
19	144.499531	192.1.109.1	192.1.139.1	TCP	54	54748 → 80 [ACK] Seq=1 Ack=1 Win=32768 Len=0
20	155.255010	192.1.109.1	192.1.139.1	TCP	54	[TCP ACKed unseen segment] [TCP Previous segment not

We can find one of response messages in Port Capture.

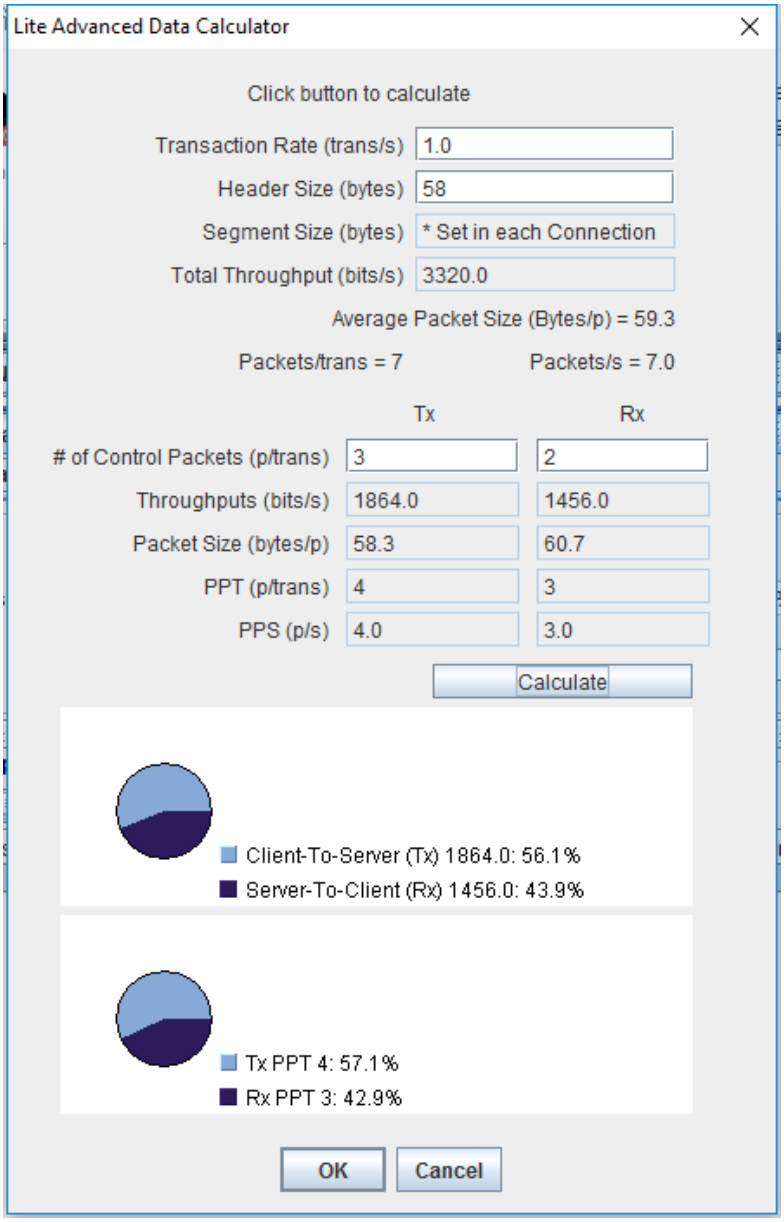
```
    54 432.004394    192.1.139.1          192.1.109.1          TCP        62 [TCP segment of a reassembled PDU]

> Frame 54: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
> Ethernet II, Src: 11:22:33:44:55:66 (11:22:33:44:55:66), Dst: ff:ee:dd:cc:bb:aa (ff:ee:dd:cc:bb:aa)
> Internet Protocol Version 4, Src: 192.1.139.1, Dst: 192.1.109.1
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 61344 (61344), Seq: 1, Ack: 1, Len: 8

0000  ff ee dd cc bb aa 11 22  33 44 55 66 08 00 45 00   ....... " 3DUf..E.
0010  00 30 04 2b 00 00 40 06  fe 97 c0 01 8b 01 c0 01   .0.+..@. ........
0020  6d 01 00 50 ef a0 1c d1  34 4e 1c d1 2b 7d 50 10   m..P.... 4N..+}P.
0030  80 00 5d 93 00 00 31 32  33 34 35 36 37 38         ..]...12 345678
```

Msg #	Connection...	Direction	Time(ms)	Delay(ms)	Data
1	0	→	0	0	.
2	0	←	0	0	12345678

We have set Header Size to 20 and you can find the proof in IP Packets.



User case: Lite DMF with 1 connection



One connection with two messages request and response.

First Connection Persistent

#	Transport	Source Port	Dest Host	Dest Port	Initiator	TOS/DSCP	Segment S...	DNS Query
0	tcp	Random	0	80	Client	0	1500	None

Msg #	Connection...	Direction	Time(ms)	Delay(ms)	Data
1	0	→	0	0	.
2	0	←	0	0	12345678

Now everything's fine in Test reports:

L3 Client::Total Packets Sent/Sec (P-I)	4	4.066	3.934
L3 Client::Total Packets Received/Sec (P-I)	3	3.066	2.934
L3 Client::Total Bits Sent/Sec (P-I)	1,864	1,895.34	1,832.656
L3 Client::Total Bits Received/Sec (P-I)	1,456	1,486.834	1,425.162

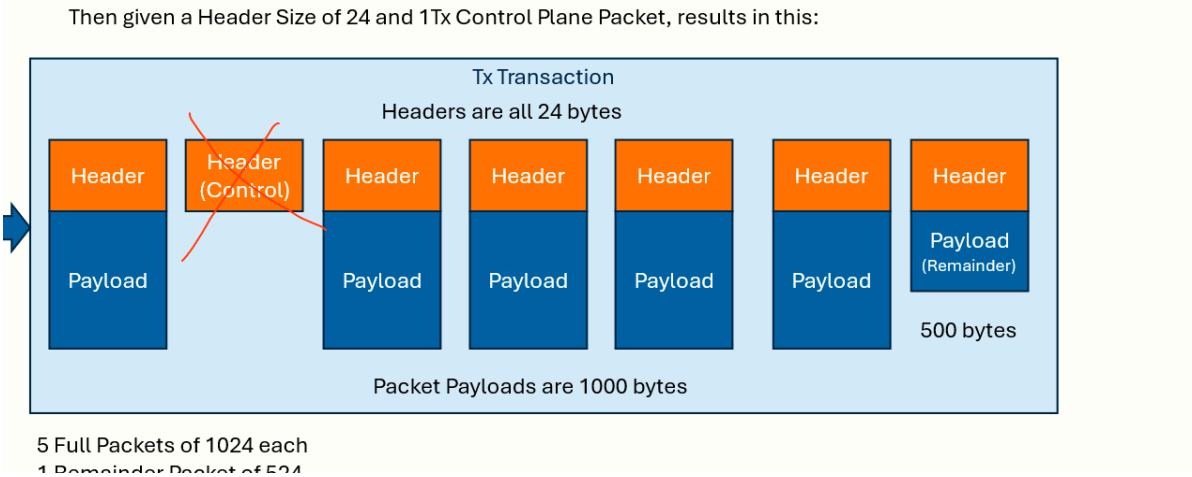
Formulas

The following is a breakdown and explanation of the methods and math involved with calculating the DMF properties.

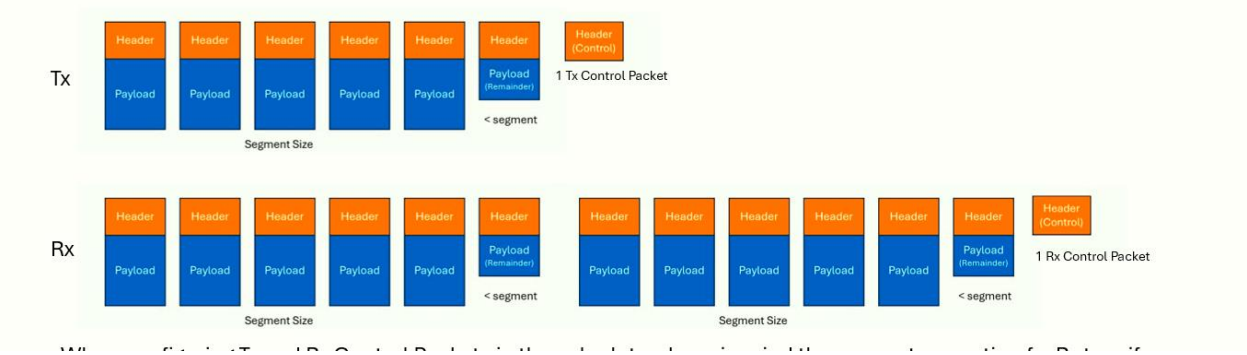
Basic Data Calculation Variables:

Total Throughput (bits/s) = **TT**

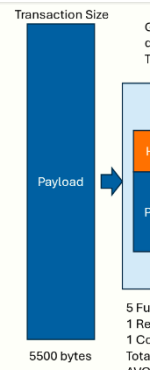
Total Data Bytes Per Tx Transaction (a single flow) = **TxBT**, this is the part we repeat on Rx based on HDE, it's the bytes in all the packets that hold data including the header:



Total (Bi-Directional) Bytes per Transaction = **TBT**, all the packets in both directions including Control. This is what is multiplied by the Transaction Rate to get TT:

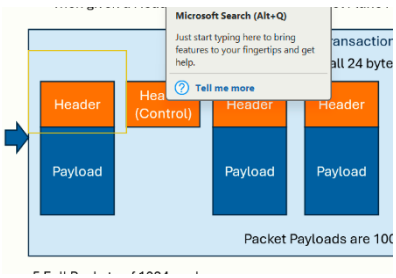


Transaction Size (bytes) = **TS**, the base amount of bytes for the DMF, Tx.



Transaction Rate (trans/s) = **TR**

Header (bytes) = **H**



Segment Size (bytes) = **S**

Host Data Expansion Ratio = **HDE**

Total Expanded Data Factor, calculating the TBT, = **TED** = **HDE**+1, (Given Rx = Tx*HDE, the Total Payload Data = Tx + Rx = (Tx*1) + (Tx*HDE) = Tx*1 + Tx+HDE ➔ Tx*(HDE+1)), we'll call HDE+1, TED

Total Packets per TxBT = P = (TS + **S** – 1) / **S**

Total Full Packets = **TFP** = **TS** / **S**

Full Packet Size (bytes) = **FPS** = **S** + **H**

Partial Packet Payload Size (bytes) = **PPPB** = **TS** % **S**

Partial Packet Size = **PPB** = **PPPB** + **H**

Empty Space in Partial Packet (bytes) = **ES** = **S** – **PPPB (**or **FPS** – **PPB**)

Total Control Packet Bytes per Transaction (bytes/trans) = **TCPB** = (**Rx Control Packets** + **Tx Control Packets**) * **H**) //This is fixed size per transaction independent of **HDE**

TxBT = ((**TS** + **S** -1) / S) * (**FPS**) – (if **PPPB** > 0 then **ES** else 0) // Total full segment packets including counting a partial packet as full MINUS the empty (payload only) space in remainder packet if one exists, leaving us with accurate Full packets + Partial Packet

To calculate total bytes per transaction:
TBT = (**TED** * **TxBT)** + **TCPB** //For a given Tx bytes the Tx+Rx bytes would be Tx * (HDE+1) + all the Control packet bytes which are independent of HDE

All units are bytes or bytes/trans, TED is unitless

To calculate total (bi-directional) throughput per second (This is common to Advanced Data and Lite DMFs too):
TT = **TBT** * **TR** * 8

bits	~~bytes~~	~~trans~~	bits
s	~~trans~~	s	~~byte~~

To calculate transaction rate (This is common to Advanced Data and Lite DMFs too):

TR (trans/s) = TT / (**TBT** * 8) // Total throughput / (Total Bytes in a transaction * 8 bits/byte) // everything on the right side can be calculated and the units cancel out correctly:

trans	~~bits~~	trans	~~bytes~~
s	s	~~bytes~~	~~bits~~

To calculate HDE (Basic Data only) the user would have to first establish a valid configuration and then clear the HDE field and set a different Transaction Rate or a different Transaction Size and then have calculator try to find an HDE that could meet that need. This might be used if the goal was to keep a static Tx size but adjust Rx to meet a throughput. HDE is an integer, and the precision will always be based on the Tx size since it is a multiple of that and only within a range of 0-100.

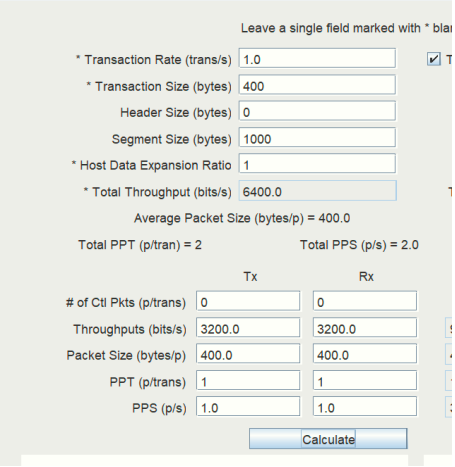
HDE = (((TT / (**TR** * 8)) – **TCPB**) / **TxBT - 1)** // TT/8 converts bits/s to bytes/s, the /TR trans/s converts to bytes/s to bytes/trans, - TCPB (bytes/trans) takes control packets out of the equation and then / the Tx (bytes/trans) cancels out the bytes/trans, leaving us with unitless HDE.

	~~bits~~	~~bytes~~	~~s~~	~~trans~~
	~~s~~	~~bits~~	~~trans~~	~~bytes~~

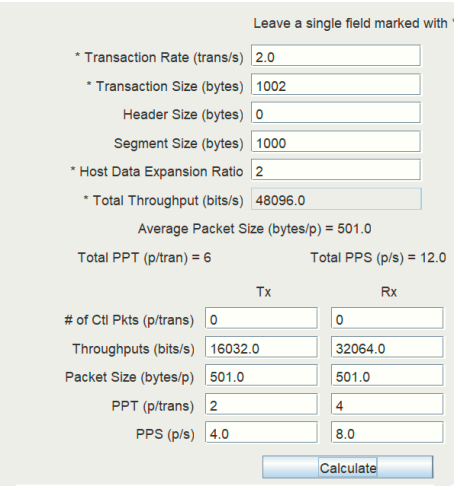
To calculate the Transaction Size is difficult due to all the variables involved, it requires a numerical search not a direct formula. A user might want to find the Transaction Size (TS) when otherwise the DMF requires a Transaction Rate (TR) greater than the 1000.0 limit. The only way to meet the throughput in this situation would be to increase the amount of data per transaction, either with HDE or TS.
TS = undocumented function

Examples of calculating the Total Throughput (TT), using three main formulas:

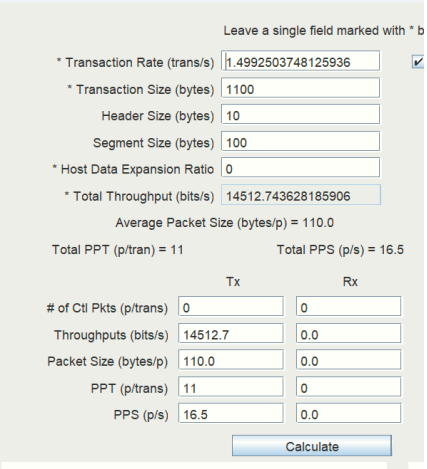
Calculate **TxBT**: ((**TS** + **S** -1) / S) * (**FPS**) – (if **PPPB** > 0 then **ES** else 0)
Calculate **TBT** = (**TED** * **TxBT**) + **TCPB**
Calculate **TT** = **TBT** * **TR** * 8

1. 

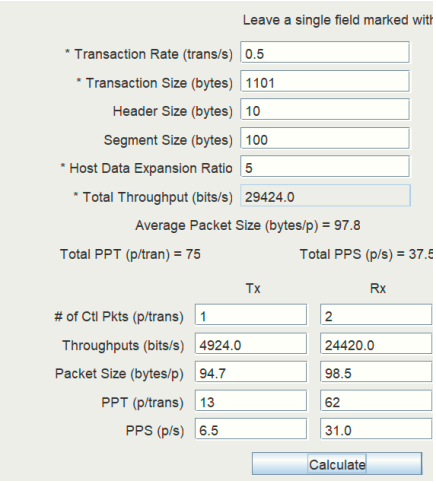
TS=400, S=1000, H=0, so FPS=1000 and PPPB=400 expect only 1 partial packet of 400.
TxBT=400+1000-1 / 1000 = 1399/1000 = 1*1000 – (1000- 400) = 1000 – 600 = 400
HDE=1
TBT=2*400+0 = 800 bytes/transaction
TR=1.0
TT = 800*1*8 = 6400 bps as seen in the image.

2. 

TS=1002, S=1000, H=0, so FPS=1000 and PPPB=2, expect 1 full and 1 partial of 2
TxBT = 1002+1000-1 / 1000 = 2003/1000 = 2*1000 - (1000-2) = 2000 – 998 = 1002
HDE=2,
TBT = 3*1002 = 3006 bytes/transaction
TR=2.0
TT = 3006*2*8 = 48096 bps as seen in the image

3. 

TS=1100, S=100, H=10, so FPS=110 and PPPB=0, expect 11 full packets 1100+100-1 / 100 = 1199 / 100 = 11*110 – 0= 1210 – 0 = 1210 (11 packets 110 each with header)
HDE=0
TBT = 1*1210 = 1210 bytes/transaction
TR=1.5 => rounded down to 1.499250374812594 due to 1 / 1.5 == 0.6666 needs to be rounded to 0.6670 1 / 0.667 → 1.499250374812594
TT = 1210*1.499250374812594*8 = 14512.7 (would be 14520bps if 1.5 could be used)

4. 

TS=1101, S=100, H=10, so FPS=110 and PPPB=1, expect 11 full packets and 1 partial 1101+100-1 / 100 = 1200 / 100 = 12*110 – (100-1) = 1320 – 99 = 1221 (11 packets 110 each with header and one with header + 1)
HDE=5, Tx Control Packets=1, Rx Control Packets=2
TBT = 6*1221 + 3*10 = 7326 + 30 = 7356 bytes/transaction
TR=0.5
TT = 7356*0.5*8 = 29424 bps