International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 6, August - 2012

# Sliding-block Window Protocol for Interplanetary Links

**Saroj Kumar Chandra**
Department of Computer Sc. & Engineering,
Chouksey Engineering College, Bilaspur(CG)

**Prince Kumar Sahu**
Department of Computer Sc. & Engineering,
Chouksey Engineering College, Bilaspur(CG)

***Abstract***: At data link level a protocol is needed to keep record of the *protocol data unit* (PDU) sequences sent and their respective acknowledgements received by the *Interplanetary Internet* (IPN) gateways linking two disparate regions. *Sliding window protocol* (SWP) can be used for the purpose. With that sliding window works as a variable-duration window that allows a sender to transmit a specified number of data units before an acknowledgment is received or before a specified event occurs. For IPNs bandwidth is very low. The links have large round trip time and suffer frequent blackouts. Even if the data rates are low the bandwidth-delay product has a large value. SWP suffers many disadvantages. Our proposed improvisation called *sliding-block window protocol* (SBWP) proves to be a better option.

***Keywords***- Interplanetary Internet, sliding window protocol, sliding-block window protocol, protocol data unit, block number.

## I. INTRODUCTION

Interplanetary networks are particularly having high latency intermittent links. Hence there is need for special kind of protocols, software's and hardware infrastructure for them. Space communication infrastructure has blown out of the need to provide special services to each new mission as they were implemented. Such vertically organized missions used infrastructure pieces that were designed to that single mission's requirements, resulting in communications assets only useful to that single mission. Hence, space communication assets must be developed into a more horizontal infrastructure [3] [4] [5], so the capabilities can be used in any kind of mission. They can take advantage of the capabilities offered by the *Internet* and its technologies. One of the primary advantages of the *Internet* is that it is truly horizontal in structure. *Interplanetary Internet* (IPN) [1] [2] was proposed to define the architecture and protocols necessary to permit interoperation of the *Internets* resident on Earth with other remotely located *internets* resident on other planets or spacecraft in transit.IPN would essentially have at least two *Internets* (regions) separated by interplanetary distances. Let one of them is at earth and the other is on mars. The *Internet* on mars has a group of nodes which are always connected to each other directly or indirectly without any disruption. Let us visualize it as a group of rovers spread over 100 km radius. Some of them can communicate through wireless and some (that are far away) through a geosynchronous satellite above them in the skies of mars. Such an *Internet* would run in same way as the *Internet* works on earth because the delays are in milliseconds and there is no disruption. But the link between the *Internet* on earth and the *Internet* on

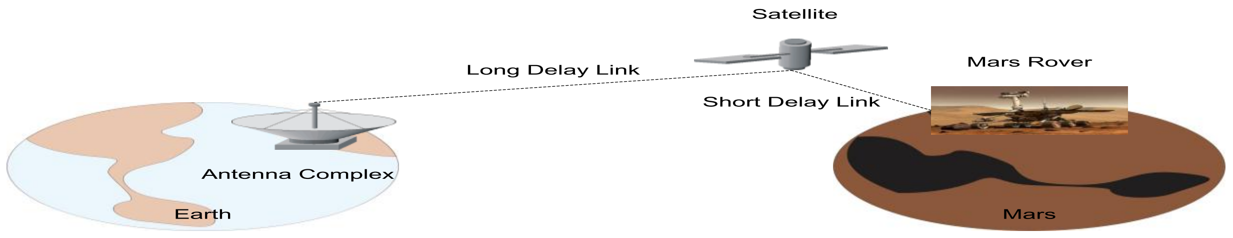

Fig1. An interplanetary communication network.

mars have a large delay and is intermittent (prone to disruption). Let there be a single link only (no congestion). There is need for a point-to-point transmission protocol that would ensure reliable transmission over the link. This link can't be taken as simple terrestrial link. But, from earth user want to see the nodes on mars as simple nodes on *Internet* that respond somewhat with a delay.

At data link level a protocol is needed to keep record of the frame sequences sent and their respective acknowledgements received by the IPN gateways linking two disparate regions. Sliding window protocol can be used for the purpose. With that sliding window works as a variable-duration window that allows a sender to transmit a specified number of data units before an acknowledgment is received or before a specified event occurs. For IPNs bandwidth is very low. The links have large round trip time and suffer frequent blackouts. The error rates are in the order of $10^{-1}$. Even if the data rates are low, the bandwidth-delay product has a large value. Simple *sliding window protocol* (SWP) using selective repeat strategy for retransmission is not suitable for IPNs and suffers following inconveniences:

- Protocol Data Units (PDU) need to be acknowledged individually.
- Due to large bandwidth-delay product the window size is very large.
- Due to repeated retransmissions transmission time suffers.

## II. SLIDING-BLOCK WINDOW PROTOCOL

*Sliding-block window protocol* (SBWP) is an improvisation over *sliding window protocol*. Here too the PDUs are basic units having unique *sequence number* as identifiers (normally, sequence number identifies the byte number in a byte stream). But PDUs are grouped in blocks for specific purpose. Therefore a set of PDUs have a common *block number*. The PDUs of a block are acknowledged in group. That means acknowledgements are deferred until a whole block is not received with or without errors. The bandwidth-delay product has a large value for IPNs. A large number of PDUs can fill the link. Let's say *x* number of PDUs fill the link. The PDUs are grouped in blocks. Size of a block is fixed as *n* number of PDUs. Then the number of blocks filling the link is *x*/*n*.

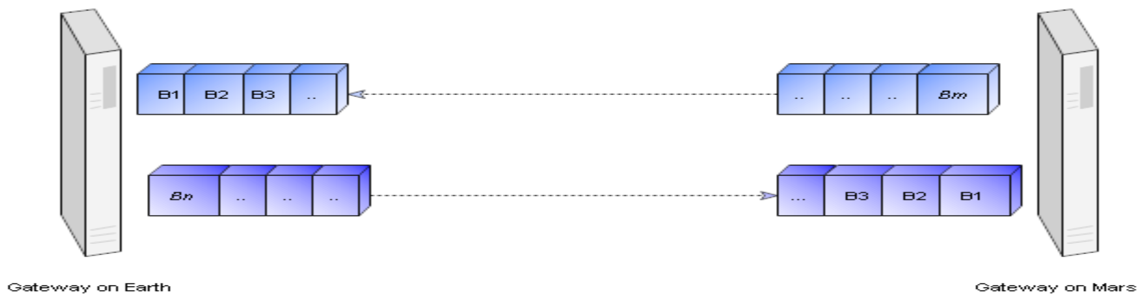

Fig2. Block transfer between gateway on Earth and gateway on Mars.

Fig3 shows a flowchart that provides the details of the working of *sliding-block window protocol*. The window is loaded with PDUs to be sent over the link. The *forward pointer* p1 is initialized to block 0. The *acknowledgement pointer* is also initialized to block 0. A block of PDUs is sent from the sending window. The *forward pointer* is moved to next block on the window that is block 1. Now look if some block acknowledgment has arrived from the receiver side. If not, send a new block from the window and move the *forward pointer* further. If some acknowledge PDU has arrived for some block sent before, open it and see which of the PDUs have not been delivered safely. If all PDUs were delivered successfully move the acknowledgement pointer to the next block on the window.

In case of errors in the PDUs, the next step will be to check the mode of operation of the window. It is important to know that the window will work in only one of the modes at a time. A window cannot switch modes while in operation. The flowchart shows both modes together so that a comparison can be made. The differences are clearly visible in two modes of operation.

The two modes of operation of SBWP are simply named mode1 and mode2. The details of these modes are given below:

*Mode1*: Actually this mode is adapted from normal sliding window protocol used by TCP. Here the erroneous PDUs are retransmitted once again in a group. This group forms a block, but the PDUs successfully delivered are not in it. The acknowledgment pointer is not moved forward. The window resumes sending blocks lined next. When acknowledgement PDU is received again for the same block, it is seen for the contents. If there were no errors in the delivery of packets this time the acknowledgement pointer is moved forward before resuming sending of new blocks. If again there were errors the acknowledgement pointer is kept at the same point on window. The erroneous PDUs are sent again. In short the process is repeated until every PDU from the block is delivered successfully. After successful delivery the acknowledgment pointer is moved to next block. The acknowledged block is freed. Now new data can be loaded to this block on the window.

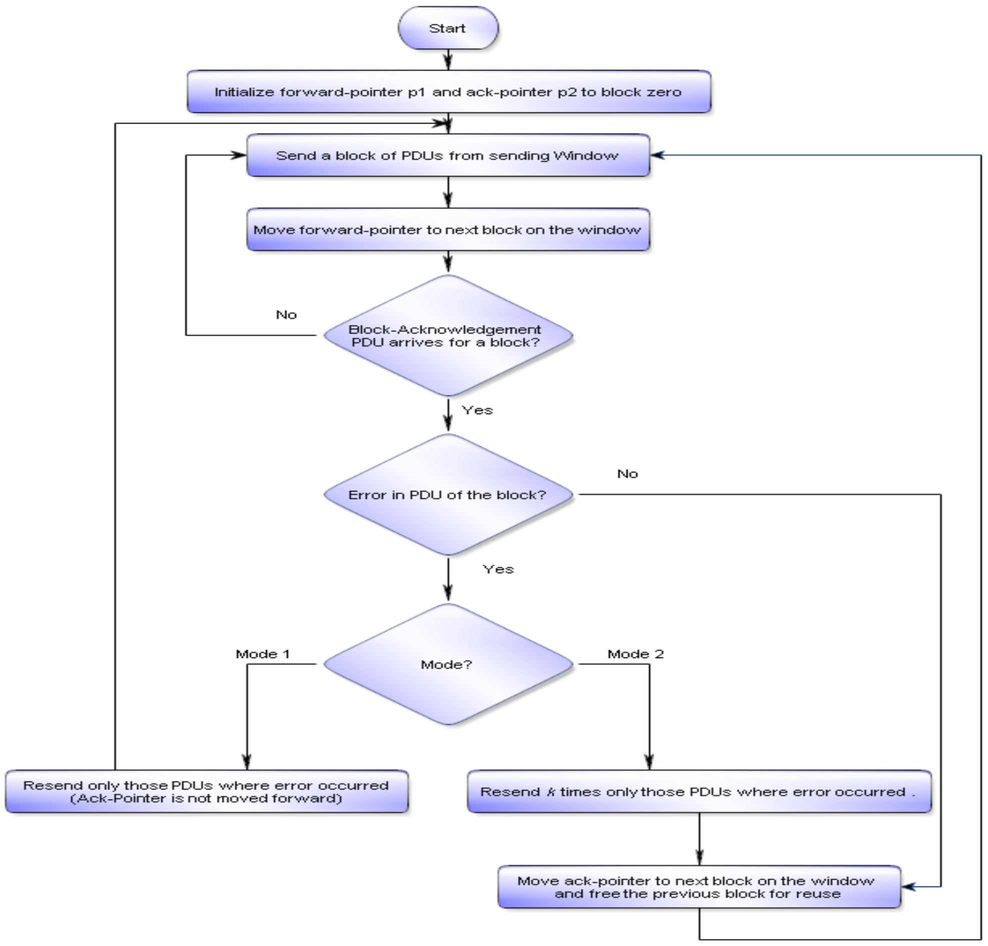
Start
Initialize forward-pointer p1 and ack-pointer p2 to block zero
Send a block of PDUs from sending Window
Move forward-pointer to next block on the window
Block-Acknowledgement PDU arrives for a block?
No
Yes
Error in PDU of the block?
No
Yes
Mode?
Mode 1
Mode 2
Resend only those PDUs where error occurred (Ack-Pointer is not moved forward)
Resend k times only those PDUs where error occurred .
Move ack-pointer to next block on the window and free the previous block for reuse

Fig3. The Sliding-block window protocol.

*Mode2*: This is the new one. Here too the erroneous PDUs are sent again together in a group. Similar to mode1 the group forms a block. But the PDUs are sent with some redundancy i.e. every PDU is sent $k$ times. This reduces probability of error to a very small value. It is assumed that probability of error is negligible so move the acknowledgement pointer to next block. The acknowledged block is freed. Now new data can be loaded to this block on the window.

## III. NUMBER OF TRANSMISSIONS AND PROBABILITY OF ERRORS

Bit error rates are very high for interplanetary links usually on the order of $10^{-1}$ [2]. Hence, there a need for strong forward correction encoding schemes. Such encoding schemes add redundancy to the data, eventually eating in more bandwidth. Ending this discussion on bit error rate ($e$) here, let us introduce PDU error rate $p$. PDU error rate can be informally defined as fraction of the PDUs that may not reach the destination safely. If error rate $p$ is 0 then there is no need for retransmission. But for IPNs $p$ can have values up to 10 percent (0.1 by fraction). So, PDUs are retransmitted many times. The retransmissions reduce the probability of errors. Let $K$ be the total number of times a block is transmitted, then the probability of errors would be $(p)^{K}$. Fig4 shows how drastically the probability of errors decreases with increase in number of retransmissions.

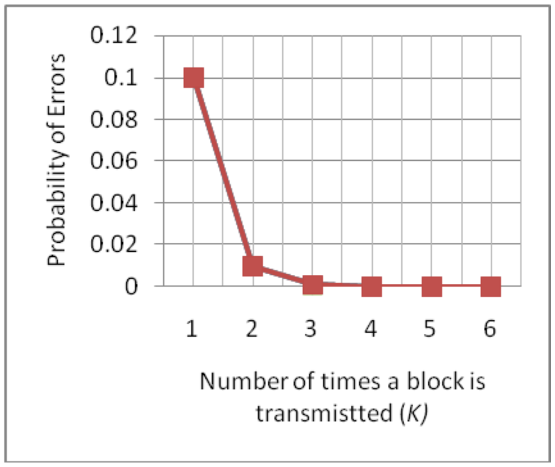

Fig4. Relation between number of retransmissions (*k*) and the probability of errors.

The number of retransmissions has great impact upon the bandwidth utilization, delivery time for a block, and storage requirements (window size). This fact will guide the evaluation of sliding-block window protocol.

## IV. ANALYSIS

Sliding-block window protocol (SBWP) works for data linking. It works in two modes explained in previous chapter i.e. mode1 and mode2. Actually this mode1 is adapted from normal sliding window protocol used by TCP. Mode2 is the new one. So comparing mode1 and mode2 will also give the comparison between our sliding-block window protocol and the conventional sliding window protocol.

Features of sliding-block window protocol those are different from simple sliding window protocol:

- Blocking is done to form groups of PDUs.
- PDUs are transmitted in blocks as one unit.
- PDUs of a block are acknowledged together. Such group acknowledgements reduce over head. For IPNs it's almost impossible to acknowledge every PDU one by one.
- The erroneous PDUs from a block are sent again in a group. That means blocking is used consistently.
- The forward pointer and acknowledgement pointer moves forward a block at a time.
- Mode2 is the preferred mode of operation.
- The acknowledgements are send reliably (the probability that an acknowledgements will be lost is extremely low and can be ignored).

Characteristics that are to be compared between mode1 and mode2 are:

- Bandwidth Utilization
- Delivery Time for a block
- Storage requirements (window size)

*Bandwidth Utilization*: First of all let us compare the bandwidth utilization by mode1 and mode2. Let the error rate or probability of error for the transmission of PDUs is *p*. For simplicity error rate here is PDU error rate *p*. So, *p* refers to the fraction of PDUs that may not reach the destination safely. If error rate *p* is 0 then the bandwidth utilization will be 100 percent because there is no need for retransmission. But for IPNs *p* can be up to 10 percent (0.1 by fraction). So, PDUs are retransmitting that decreases the bandwidth utilization.

For mode1 the retransmissions are done as many until all the PDUs from a block are not delivered accurately. When a block is sent the window waits for the acknowledgement. When acknowledgement arrives the erroneous PDUs are sent again. The window waits for the acknowledgment again. On arrival of the acknowledgement the erroneous PDUs are sent again. This time the PDUs with error for the block were lesser. Let the process repeats $k$ times. The probability of error reduces to $(p)^{k+1}$. Let there be $n$ PDUs in a block. The total number of PDUs sent in mode1 will be $n + n*p + n*(p)^2 +\ldots\ldots n*(p)^k$. The bandwidth utilization in mode1 is $n/ (n + n*p + n*(p)^2 +\ldots\ldots + n*(p)^k)$ or $1/ (1+ p + p^2 +\ldots\ldots + p^k)$. Hence, greater the value of $k$, lesser is the bandwidth utilization. For mode2 the retransmissions are done once but redundantly. When a block is sent the window waits for the acknowledgement. When acknowledgement arrives the erroneous PDUs are sent again $k$ times at the same time. The probability of error reduces to $(p)^{k+1}$. Let there be $n$ PDUs in a block. The total number of PDUs sent is equal to $n + k*n*p$. The bandwidth utilization is $n/ (n + k*n*p)$ or $1/ (1 + k*p)$. Here too, more the value of $k$ lesser is the bandwidth utilization.

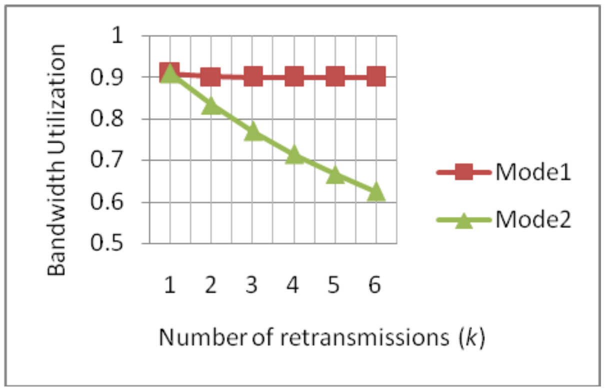

Fig5. Bandwidth utilization in various modes of operation.

Fig5 shows the comparison graph of two modes of operation. The error probability $p$ was assumed to be equal to 0.1. Retransmissions are unavoidable to curb error probability, so mode1 will have better bandwidth utilization than mode2. For mode2 the bandwidth utilization goes down steeply with increase in number the number of retransmissions. If 3 retransmissions are enough for successful delivery of a block then the bandwidth utilization in mode2 will be above 75 percent. Mode2 deliver the blocks lot faster than mode1. For faster delivery the bandwidth utilization of 75 percent is not bad.

*Delivery Time of a Block*: The time taken by a block to reach the destination is equal to the propagation delay. For simplicity ignore transmission time and receiving time of a block. Thought for IPNs transmission time and receiving time of a block has large values but can be ignored in comparison with propagation delay. The round trip time (RTT) is two times the propagation delay. The delivery time of a block is the time taken to send it to the receiver completely without any error.

This value affects the effective probability of error after retransmissions. The probability of error becomes $(p)^{k+1}$. If no retransmission is done then the block will be delivered in 1/2* RTT, but $n*p$ number of PDUs will not be delivered accurately. If retransmission of erroneous PDUs is done once then delivery time is equal to 1/2*RTT + RTT

This is because the sending window will wait for the acknowledgement after sending a block. The block will reach the receiver in 1/2*RTT. The acknowledgement will come back in 1/2*RTT. The retransmitted PDUs will reach the receiver in 1/2*RTT. The total time comes out to be 1/2*RTT + RTT. There are $k$ retransmissions in mode1. Hence the delivery time will be 1/2*RTT + $k$*RTT (for $k$= [0, 1, 2, 3……]).

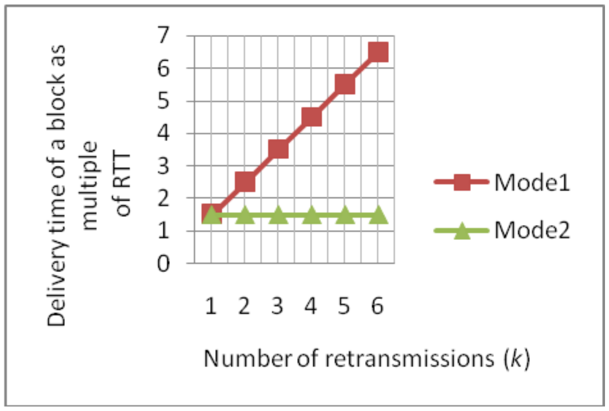

Fig6. Delivery time of a block in various modes of operation.

The delivery time for mode1 depends upon the value of *k* (number of retransmissions). The delivery time for mode2 does not depend upon the value of *k* (number of retransmissions). Here the retransmissions are done only after receiving first acknowledgement. The window moves its acknowledgement to next block and releases the buffer for reuse. Hence the delivery time for mode2 is 1/2*RTT + RTT (for any *k*= [1, 2, 3……]).

Fig6 shows the comparison graph of two modes of operation. Retransmissions are unavoidable to curb error probability so mode1 will take much more time to deliver a lock successfully without any error. Mode2 will take a fixed amount of time that is 1.5 times the RTT, so it is preferable.

*Storage Requirements (Window Size)*: One has to store the PDUs in some memory before transmission. For interplanetary links the data rates are low (because of limited bandwidth and high attenuation) but data rate-delay product has a large value. Here for simplicity the units of data rate will be in *bits per second*. So, while using sliding-block window protocol the window size will be very large. That will need huge volatile memory if implemented in conventional manner. So, one may have to use permanent storage for partially storing the frames on the window. Here there are two modes of operation called mode1 and mode2. The minimum window size needed in both the cases will be different.

In mode1 the minimum storage required depends upon the maximum allowable retransmissions (*k*) of erroneous PDUs. If only one retransmission is allowed the minimum storage requirement will be *data-rate**RTT. The size will vary with the choice of *k*. For *k* number of retransmissions the minimum storage requirement will be *k***data-rate**RTT.

In mode2 the minimum storage required for the window will be *Data-rate**RTT. This value is fixed in mode2. It doesn't vary with the number of retransmissions (*k*). This is because a block is kept on the window for round trip time (RTT) only. After that the buffer used by the block is freed to be used again. This will definitely reduce the volatile storage requirements.

Fig7 shows the comparison of mode1 and mode2 storage requirements. Retransmissions are unavoidable to curb error probability so mode1 will use many times the memory needed by mode2. Clearly the mode2 is preferable to mode1 when scarcity of memory is there.

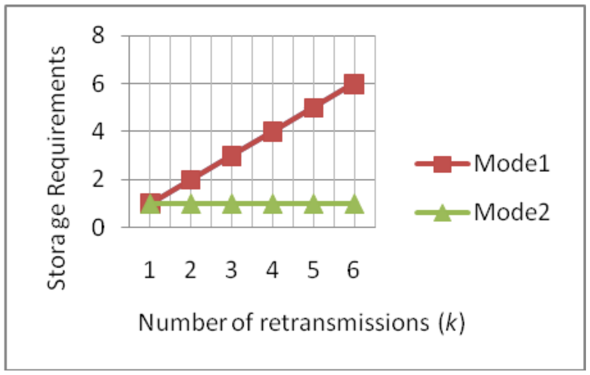

Fig7. Storage requirements for various modes of operation.

\

## V. CONCLUSION

The theoretical analysis gives formulas for bandwidth utilization, delivery time for a block, and storage requirements (window size). The mode1 based on conventional systems provides better bandwidth utilization but fails to provide faster delivery and the storage requirements are very high. The mode2 fresh approach provides faster delivery and has low storage requirements. It is somehow inferior in bandwidth utilization. The delivery time for mode1 is many times as compared to mode2. Also the memory used is also very high in mode1. Careful choice of the value of number of retransmissions (*k*) can make it best in certain conditions. So mode2 is better of the two.

## VI. REFERENCES

[1] IPN Special Interest Group, http://www.ipnsig.org/reports/memo-ipnrg-arch-00.pdf.

[2] R. C. Durst, P. D. Feighery, K. L. Scott, "Why not use the Standard Internet Suite for the Interplanetary Internet?" , http://www.ipnsig.org/techinfo.htm.

[3] J. H. Saltzer, D. P Reed, and D. D. Clark, "End-to-End Arguments in System Design" ACM Trans. Comp. Sys. 2.4, Nov. 1984.

[4] "Section 4, Space/Ground Trades", Consolidated Space Operations Contract Architecture Baseline, CSOC document, December 1998.

[5] J. Hayden, "Telecommunications for the CSOC, an Internet Interface to NASA Satellites", 1999 Second Annual International Symposium on Advanced Radio Technologies, September 8- 10, 1999