CSC485A/586A/SENG480

Assignment 3 Part 2

G8 - Junnan Lu, Bill Xiong, Colum McClay

Abstract

In this project we present the Reliable Datagram Protocol, a modified version of a transport-layer protocol built on top of UDP. UDP was formally classified and defined by David Reed in 1980. UDP provides transaction-oriented, stateless transfer of files over the transport layer of the internet protocol suite. One defining feature of UDP is the lack of retransmission delays, which makes the protocol extremely attractive to applications that rely on real-time data, such as voice-over-IP applications, online games, and most applications built on top of the Real Time Streaming Protocol. Since the growth of bittorrent, UDP has seen a massive surge in usage and still accounts for a large part of the traffic across the internet today.

With RDP we address several weaknesses and improve other features of UDP, including out of order packets, packet loss and guarantees packet delivery.

Program Behavior

Our RDP protocol is layered on top of the existing UDP protocol, and is implemented entirely in C. First, a sender and a receiver are initialized. The sender is in charge of establishing the connection, and tries up to 5 times in case of SYN packet loss. Once the connection has been acknowledged by both parties, data transfer may begin:

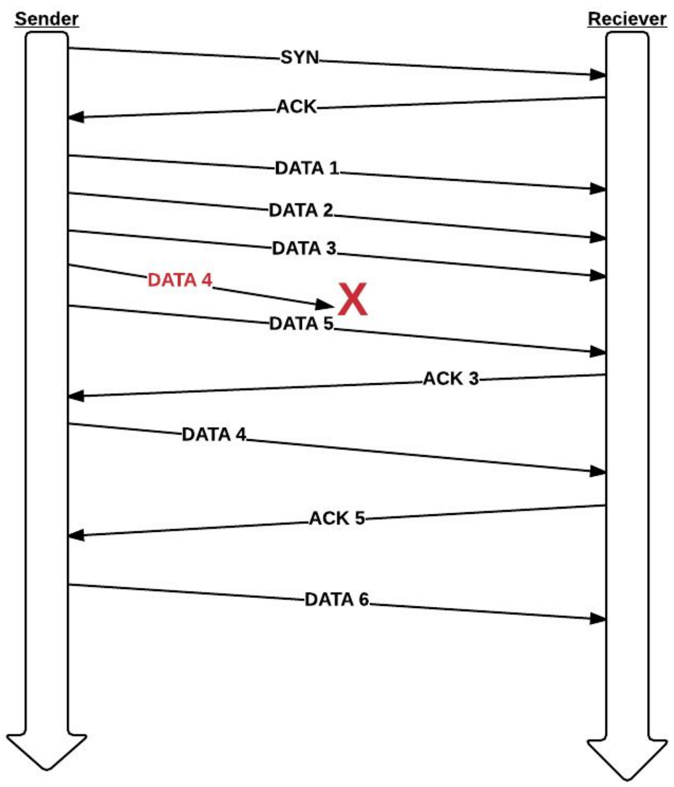


Figure 1. The RDP Protocol Flowchart. In the diagram, the first attempt to send DATA 4 fails.

The RDP Protocol handles packet reliability similarly to TCP with sequence numbers and acknowledgement numbers. RDP also exhibits a feature of congestion control: sliding window. This prevents the receiver from receiving too many packets without being able to store them. Although modern computers are able to store many packets, RDP has been modeled to only handle up to 5 packets to ensure that the mechanism works properly.
The following figure details what the RDP header contains:

RDP Header Field	**Possible Value**	**Significance**
Protocol Type: RDP	RDP	RDP Protocol
Type: _type_	DAT/ACK/SYN/FIN/RST	Type of packet
Sequence: _seqno_	0	byte sequence number
Acknowledgement: _ackno_	900	byte acknowledgement num.

Payload: _length_	900	RDP Payload byte-length
Window: _size	10240	RDP window size in bytes
(the empty line)		Signifies end of header

Figure 2: RDP Header Protocol

RDP SoftWare Component Diagram

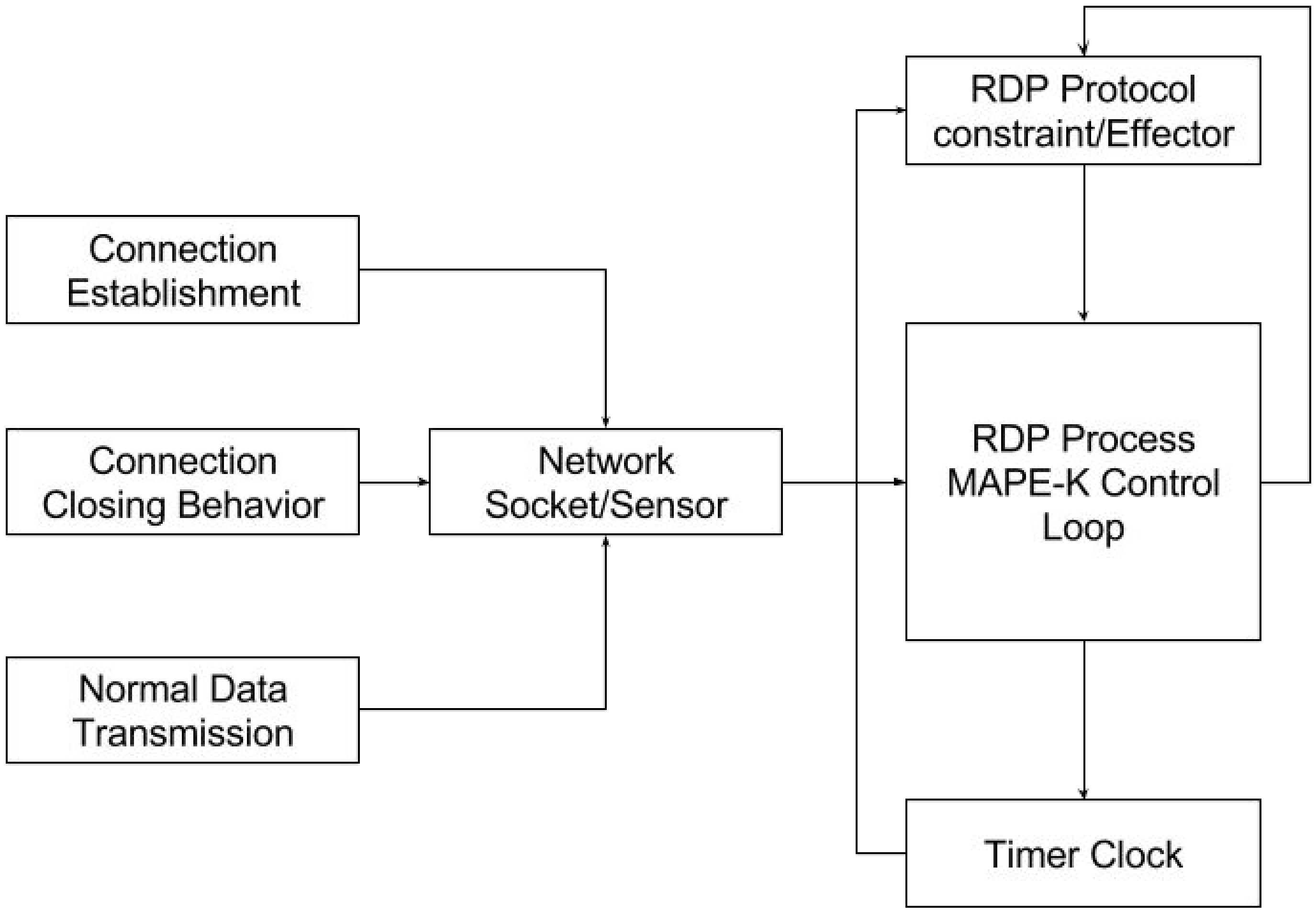

RDP Protocol constraint/Effector
Connection Establishment
Connection Closing Behavior
Network Socket/Sensor
RDP Process MAPE-K Control Loop
Normal Data Transmission
Timer Clock

Figure 3. RDP protocol's System architecture

The figure 2 shows the system architecture of the RDP system. The sensor consists of three components. They are connection establishment, connection closing and normal data transmission. Socket is a communication mechanism that allow remote hosts to develop data transfer. In here socket is working as the sensor for the RDP system to detect incoming data packets or request.

Once the data packet or request is arrived, the RDP process proceeds to the control loop where the RDP protocol constraint will be applied. The RDP protocol constraint is the system effector in RDP system. It will alter or adapt the behavior of the RDP process depend on the situation. The timer clock here can provide timing information to the effector in order to make better decision.

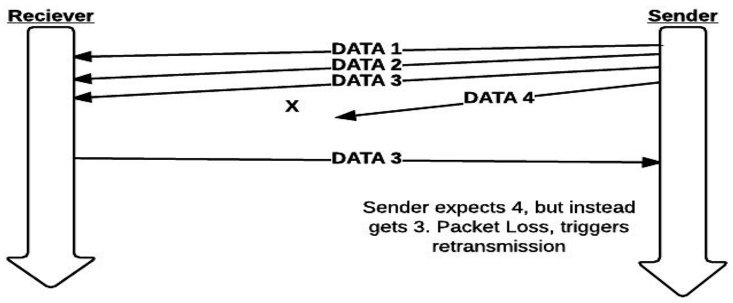
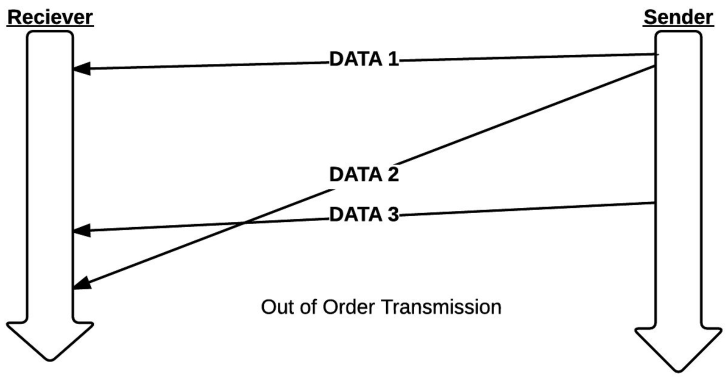

Reciever
Sender
DATA 1
DATA 2
DATA 3
DATA 4
X
DATA 3
Sender expects 4, but instead gets 3. Packet Loss, triggers retransmission

Figure 4.



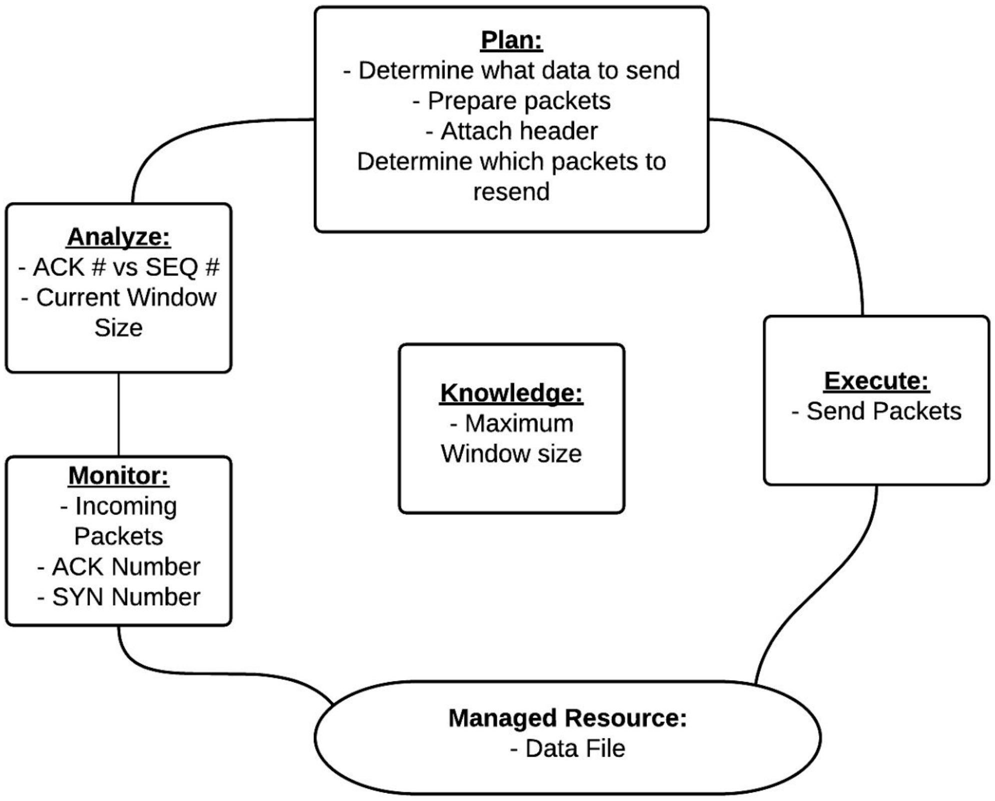


Figure 5. The Protocol's MAPE-K Diagram

Monitor: The monitoring aspect happens upon arrival of the RDP packet. The Acknowledgement number, sequence number and window size are extracted from this.

Analyze: Comparing the current sequence number to the incoming acknowledgement number allows the system to analyze whether the other party is behind in the communication.

Plan: How far behind is the receiver? What data is it missing? We must now form packets with the appropriate data to transmit. This can either be new data, or data that has already been transmitted, but lost in communication.

Execute: This is actually sending the prepared packets through the socket.

Data Flow Diagram

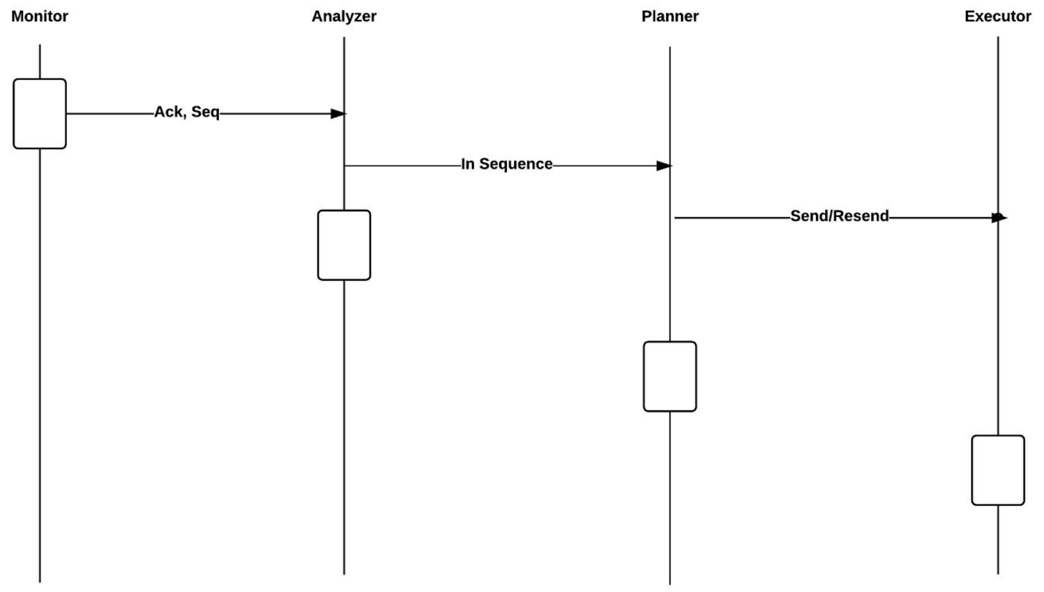


Figure 6. Data flow diagram illustrating the movement of data through the RDP Protocol

Repository

The code used to run this project and higher resolution diagrams can be found on our repository at https://github.com/bxio/UVicSAS_A3

Group Members

CSC586a-Junnan-Lu-V00217112
SEng480-Jiashu-Xiong-V00737042
CSC485a-Colum-McClay-V00745851

Works Cited

http://ipv6.com/articles/general/User-Datagram-Protocol.htm
https://tools.ietf.org/html/rfc768