

TaylorHodges_Part1

by Taylor Hodges

Submission date: 02-May-2023 03:33PM (UTC+0100)

Submission ID: 204338087

File name: 23668_Taylor_Hodges_TaylorHodges_Part1_499749_1075386201.docx (1.44M)

Word count: 2651

Character count: 14597

Socket Programming and Concurrency

Assessment 1 – Report 3000 words

Introduction

As developers networked applications can be a minefield of hackers and bottlenecking hazards that we aim to reduce these difficulties where we can. Security can be increased with TCP data delivery methods, latency can be reduced by considering packet sizes and MTU sizes, synchronous and asynchronous communication options are available for data transfer and have varying uses, and interoperability creates compatibility regardless of language used. These topics will be further discussed in this report.

TCP data delivery

TCP

Transmission Control Protocol (TCP) is a network communications protocol used to send data across a network [1].

Data is organised into smaller packets by TCP while preserving data integrity before being sent between a server and a client [1]. It ensures the validity of the data transmitted via a network [2]; TCP creates a connection between a source and destination before transmitting data, and it makes sure that connection stays active until data transmission ends [1].

Two Generals' Problem

Preserving the integrity of packets is essential. Scenarios such as the 'two Generals' problem' explain why [3]: let's say two armies plan to attack an enemy together as attacking alone will fail, but in order for them to communicate, they must pass a message through enemy territory, this message could be compromised.

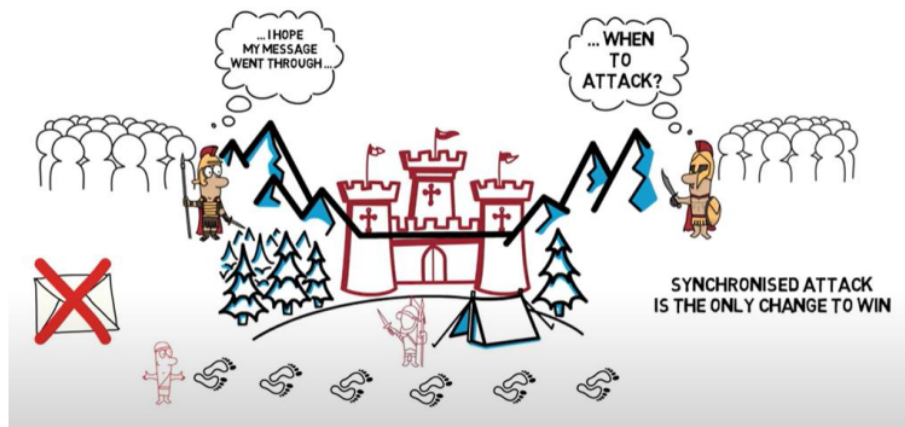


Figure 1: [4] visual example of the two general's problem scenario.

For this to be successful, one General would send a message with details of the attack plan, and the other must send a confirmation, and finally the first must send a confirmation of the confirmation. Due to the security risk of the messages being passed back and forth, it could be likely that a General does not receive a response, however they do not know if their message or the other General's message was compromised. Therefore it would be necessary for both generals to confirm the plan, causing an infinite loop [3].

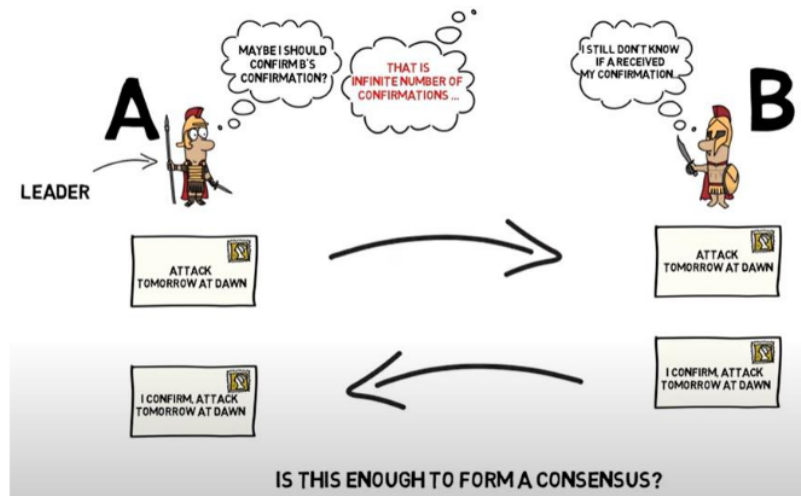


Figure 2: [4] visual example of the two general's problem confirmations.

Three-way Handshake

This is the problem that TCP must solve by establishing a three-way handshake before data is transmitted, by these three steps [5] [4]:

1. Client sends to server asking for SYN (synchronisation) - the connection.
2. Server replies with SYN-ACK (synchronisation and acknowledgement) – acknowledgement of connection request and asks client to open a connection to.
3. Client replies with (ACK) – acknowledgement and establishment of two-way connection.



Figure 3: SmartArt representation of three-way handshake.

Each side sends a sequence number (seq) in their communications and replies with an acknowledgement number (ack) that of the other's sequence number incremented [5].

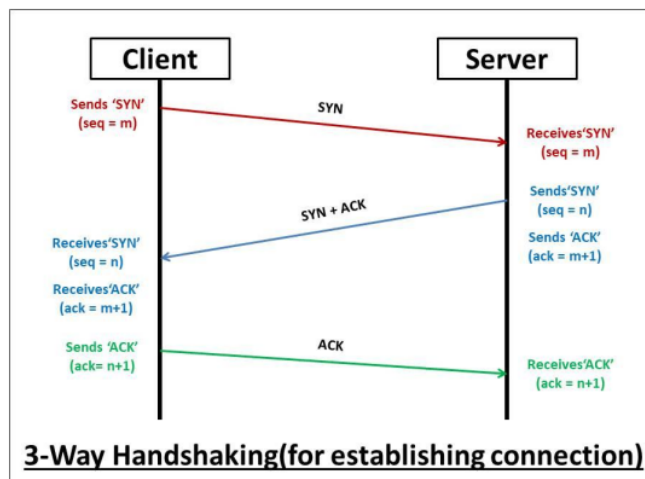


Figure 4: [6]TCP three-way handshake diagram.

Once these steps are complete, the connection is established, and the client and server can begin to transfer packets [5].

By the client and server communicating as so, a client will confirm received packets and therefore the server can deduce which packets need retransmitting, or if no confirmation is received, all packets are retransmitted [7].

As a result, they can create reliable and robust applications that can support a variety of network architectures.

The use of the three-way handshake protocol creates security between the sender and receiver and as developers should always be a priority to withhold security [8].

As developers, we want to reduce the network latency where possible, this describes how long it takes for data packets to travel between sender and receiver; network latency is increased by the number of trips between sender and receiver, so whilst we want to maintain a secure connection with connection requests and acceptations, we don't want a slow system, which is why the receiver send theirs together as one response [8].

MTU and data transfer

OSI and TCP Model

The Open Systems Interconnection (OSI) model depicts how computer systems communicate with other computer systems over a network, it has seven layers [8]:

1. Application layer
2. Presentation layer
3. Session layer
4. Transport layer
5. Network layer
6. Data link layer
7. Physical layer

When a message is sent from one node to another, it is processed from application layer through to physical layer, over a network, and received physical layer to application layer encapsulated inside.

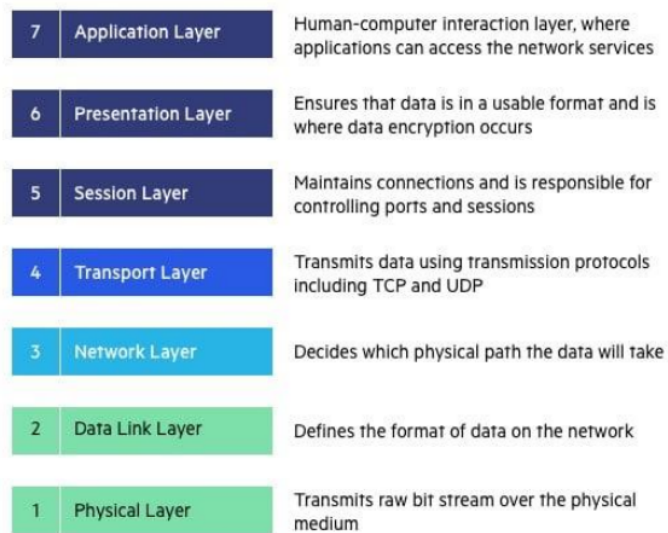


Figure 5: [8] The OSI 7 layers.

Nowadays the internet is based on the TCP/IP model due to its simplicity, but OSI is the international standard and is widely used [8].

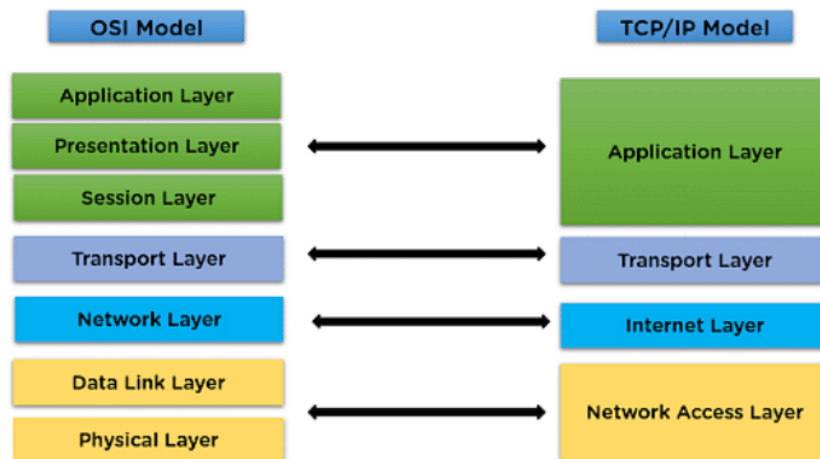


Figure 6: [9] OSI model vs TCP/IP model.

Transmission Control Protocol/Internet Protocol (TCP/IP) model differs to the OSI model as the model has less layers but is more complex due to these grouped layers. Typically, the application, presentation, and session layer are encompassed into the application layer.

The TCP/IP model has four or five layers:

1. Application layer (application, presentation, and session layers in the OSI model.)
2. Transport layer
3. Network layer
4. Data link layer*
5. Physical layer*

*On the four-layer model, the data link and physical layers are combined and known as the network access, network interface, or link layer.

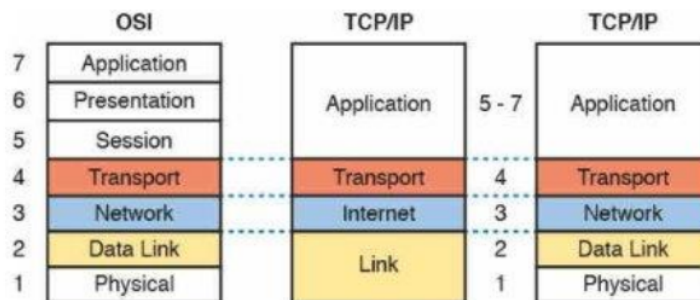


Figure 7: OSI model vs 4- and 5-layer version of the TCP/IP model.

Protocol Data Unit (PDU)

PDU is used to refer to each layer's protocol-specific control information as well as the relevant data [10]. As a PDU passes down/up a layer, the layer will add/delete their block of information about its protocol [10].

PDU have different names for their blocks of data based on what layer they are in [10]. PDU names and the layers they apply to [11]:

- PDUs
 - Application layer
 - Presentation layer
 - Session layer
- Segments – holds lots of data to be segmented into smaller pieces.
 - Transport layer
- Packets – holds IP header, source, and destination IP address.
 - Network layer
- Frames – holds source and destination in the header.
 - Data link layer
- Bits – holds 1s and 0s.
 - Physical layer

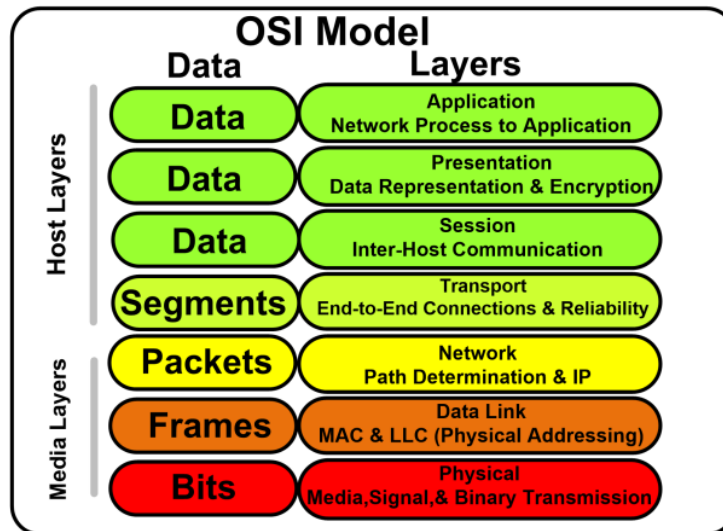


Figure 8: [12] Data type of PDUs in layers of OSI model.

Maximum Transmission Unit (MTU) and Fragmenting

The data link layer (layer 2) has a maximum number of bytes that a packet can be when sent through a network, known as the MTU [13]. The size of the MTU can depend on the type of network in use. If the size of data being transmitted is larger than the MTU, then the data will be split into packets of an appropriate size so they can be transmitted [14].

When a packet sent is bigger than the MTU of the next device in the transmission line (routers, switches, and servers) the packet must be fragmented to pass through and reassembled at its destination at the IP/network layer before passing to the transport layer [15] [16]. This means when receiving fragmented packets, that 2 or more times as many packets must be processed at the network layer.

Fragmenting creates latency as more packets are then in transit, meaning communication then becomes less efficient, so is not the most practical solution [15].

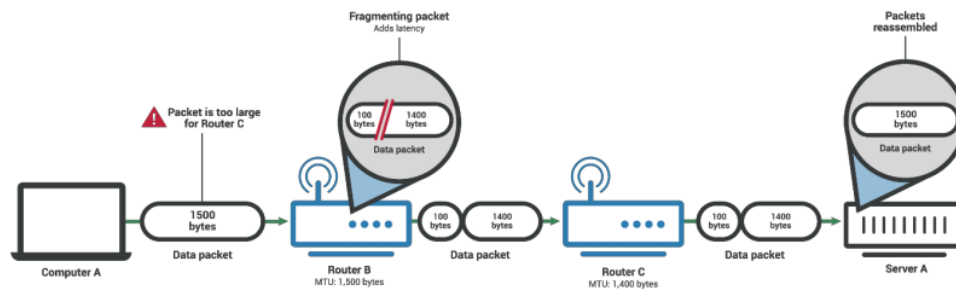


Figure 9: [15] Fragmenting of packets.

Sometimes fragmentation is not possible, this is when IPv6 is used, or when packet headers activate the "Don't fragment" flag; in these scenarios, the packet is dropped as they cannot be fragmented [15].

Network latency can also be reduced by minimising the number of packets, whilst also minimising the size of the packets to prevent fragmentation [17].

Bandwidth depends on the MTU size and is also an important factor to consider as developers; data transmission size and frequency should be minimised as not to cause network congestion leading to increased latency [19].

Synchronous and Asynchronous Transmission

I/O Dispatch Models

I/O dispatch models describe how computer systems handle input/output (I/O) operations. Different models, such as synchronous and asynchronous models, can be applied [17]:

- In a synchronous approach, the software waits for the I/O operation to respond before going on to the subsequent job. This is helpful in situations when response time is a top priority, like in real-time systems [17].
- In an asynchronous model, the I/O action is carried out in the background while the programme still runs. This is helpful for applications where a reaction time delay is acceptable or when several I/O operations must be carried out simultaneously [17].

A relevant I/O dispatch model for network applications is the select/poll model. This model allows multiple file descriptors (such as network sockets) to be monitored simultaneously, allowing the program to respond to events such as incoming network data, without blocking the execution of other tasks [18].

Synchronous and Asynchronous

There are two main protocols for transferring data over a network: synchronous and asynchronous transmission [17]. Synchronicity and asynchronicity represent distinct communication modalities employed within network protocols. Synchronous communication is a communication mode characterised by the exchange of data between the sender and receiver in real-time. On the other hand, asynchronous communication is a mode of communication that allows the sender and receiver to exchange data at their own pace.

Both devices must have a synchronised clock signal during transmission so that the receiver can read the bits in at the same rate that the sender reads them.

Synchronous

Synchronous works by establishing a common clock signal between the sender and receiver [18]. Data is then sent at the same point in the clock cycle, meaning the sender and receiver both know when data transfer will begin. Data can then be sent from sender to receiver without intervals and can be used to continuously send large amounts of data alongside timing signals [19].

Synchronous data is transferred via synchronous transmission in the form of blocks or frames [17] which stream consistently without gaps between bytes. SYN characters are sent at the start of blocks to synchronise the connection and begin reading bits [20]. The SYN characters proceed all blocks of data to keep the clocks synched [21].

There could be a delay in establishing a common clock signal, but once complete, data is sent quickly at regular intervals in real time [18], this reducing timing errors [22].

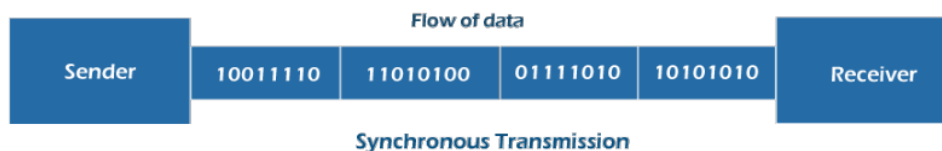


Figure 10: [22] Synchronous data transmission - Format of data flow.



Figure 11: [18] Synchronous data transmission – clock: active block sending.

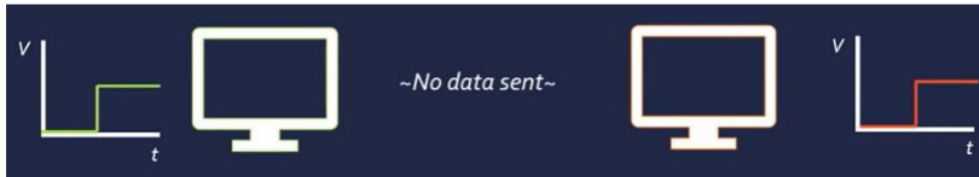


Figure 12: [18] Synchronous data transmission – clock: inactive block sending.

Asynchronous

In asynchronous transmission there is no consistent timing signal used because asynchronous does not need to permanently synchronise devices, except when transmission is taking place. This makes asynchronous cheaper and easier to implement as it does not need to share a clock.

Data is transferred through asynchronous transmission as characters or bytes [17] with gaps between data. Bytes are capped with a start and stop bit and each transmission is separated by gaps to show a clear stop to each byte. Due to the gaps and extra bits, transmissions are slower.

Direction of message sent →		
Stop	Data (byte)	Start
0	10110110	1

Figure 13: Example of sent data using asynchronous transmission.

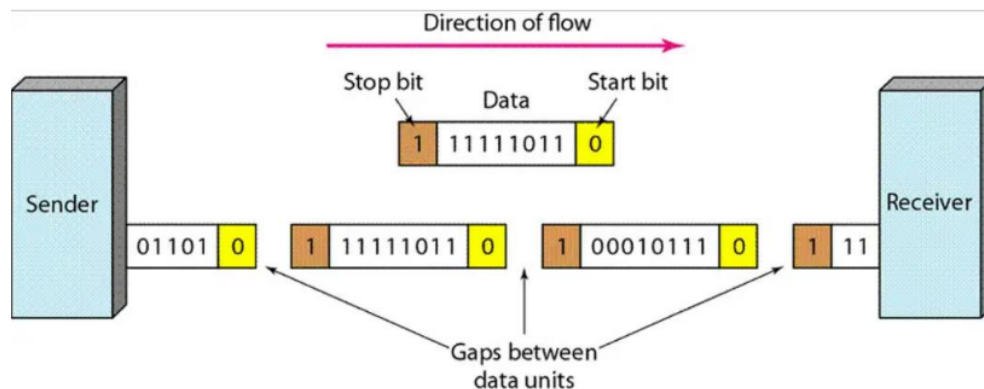


Figure 14: [23] Asynchronous data transmission - Format of data flow.

Uses of Synchronous and Asynchronous

Synchronous communication applications encompass technologies such as voice over IP (VoIP), video conferencing, and real-time online gaming. The transmission of real-time voice data in VoIP necessitates synchronous communication with minimal latency [24]. The utilisation of video conferencing technology necessitates the real-time transmission of both audio and video data, thereby requiring synchronous communication to prevent any potential delays or synchronisation issues [25]. Synchronous communication is a crucial requirement for real-time online gaming to ensure that players are participating in real-time and to prevent any potential delays or lag [26].

Conversely, applications that employ asynchronous communication encompass email, file transfer protocols (FTP), and Simple Message Transfer Protocol (SMTP). Asynchronous communication refers to the transmission of data without the requirement of immediate interaction between the sender and receiver. Email messages possess the capability of being transmitted and received at varying intervals, and the exchange of information does not necessitate immediate response [27]. FTP is an instance of asynchronous communication that facilitates the transfer of files between systems without necessitating real-time interaction [28]. The Simple Mail Transfer Protocol (SMTP) is a communication protocol that facilitates the transmission and reception of electronic mail messages. SMTP employs asynchronous communication to enable messages to be enqueued and dispatched at a deferred time [29].

An understanding of I/O dispatch models helps in optimizing network activity and thus improves application performance.

Scalability is important to consider when making a networked application so that the system can handle an increase in user traffic. Asynchronous transmissions are better for this as they use less resources [29].

Interoperability for networked applications

Interoperability

Interoperability is the fundamental capability of various systems, devices, applications, or products to connect over LANs or WANs, interact and exchange data with one another without hindrance, in either implementation or access [30]; meaning computerised products would be able to interact with each other seamlessly without the end user's attention [31].

These systems and software applications are able to interoperate in a way it can interpret and manipulate the data being passed back and forth and used effectively [32]. In interoperable framework, nodes can exchange, interpret, and present shared data in a format that the next node can also interpret, etc [30].



Figure 15: [30] Interoperability framework diagram.

There are three main types of interoperability [30]:

1. Syntactic – The first step towards interoperability, using common data format and data structures. Information acquired must be extremely valuable and meaningful in order for there to be semantic interoperability [33]; however, the interfaces do not require to be in the same language [34].
2. Semantic – The next step towards interoperability makes use of a controlled, shared vocabulary connected to each data element through the addition of metadata [33] so that each system understands the data [34].
3. Structural – this defines the data interchange format, which outlines the standards used to structure messages transmitted from one system to another; crucial for users to be able to comprehend the information's purpose clearly [34].

Benefits of interoperability include adaptability, better productivity, data unity, data protection, reduced errors, and lower costs.

How to achieve Interoperability

The use of open standards [35], API integration [36], middleware software [37], a service-oriented architecture (SOA) strategy [38] or the implementation of an enterprise service bus (ESB) [39] are a few techniques that can be used to achieve interoperability in network applications.

Open standards that specify the protocols, formats, and processes for data sharing across various systems, such as HTTP, SMTP, and XML, provide smooth communication [35]. Using standardised protocols like RESTful APIs, APIs allow many apps to communicate with one another by exchanging data and functionality [36]. Data translation, security, and authentication are all handled by middleware as a mediator between several applications [37]. Using open standards or APIs, SOA divides large systems into smaller services that may be utilised by many applications [38]. ESB manages data routing, transformation, security, logging, and auditing while providing a centralised communication infrastructure for many applications [39].

As developers our applications need to have platform compatibility, which refers to compatibility between the platform and product. our design should be platform independent using interoperable methods such as RESTful web applications that can operate with many different languages [45]. This way, developers can create applications that smoothly integrate into various networks, devices, and applications.

Conclusion

In order to create a good network application, many decisions need to be made on the best approach for this. Network applications been to be scalable, secure, reduced latency, interoperable and reliable in order for visitors to have a smooth experience and re-visit the site. These factors can be improved in many different ways such as:

- Use TCP to establish a secure connection whilst keeping communications minimal to reduce latency.
- Bandwidth can get congested, so it is important to consider the expected MTU size to ensure packets are not too large and create an efficient communication where possible without fragmenting.
- By using asynchronous communication methods, the site can be scalable and accommodate a flexible size of foot traffic to the site.
- Finally, interoperability is grate at creating compatibility between systems and platforms even when using differing languages.

FINAL GRADE

70/100

GENERAL COMMENTS

Instructor

N.B. the reference list was not attached to this document when submitted - it was sent separately via e-mail by request. I have not penalised this in the marking.

A good report, with a large number of sources, albeit trending a little web-based/non-authoritative in places.

A good effort at the implementation - clearly a lot of effort went in here, though I suspect it's not quite as finished as you would've liked. Nonetheless, some good work done.

Part 1 (report): 77

Part 2 (implementation): 62

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8
