ARQ techniques
(window protocols)

Gruppo Reti TLC
nome.cognome@polito.it
http://www.telematica.polito.it/

Copyright

Quest'opera è protetta dalla licenza Creative Commons NoDerivs-NonCommercial. Per vedere una copia di questa licenza, consultare:
http://creativecommons.org/licenses/nd-nc/1.0/
oppure inviare una lettera a:
Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.

This work is licensed under the Creative Commons NoDerivs-NonCommercial License. To view a copy of this license, visit:
http://creativecommons.org/licenses/nd-nc/1.0/
or send a letter to
Creative Commons, 559 Nathan Abbott Way, Stanford, California 94305, USA.
Techniques to protect against transmission errors

- **FEC** (Forward Error Correction)
- **ARQ** (Automatic Retransmission reQuest)

- Both need additional bits in the packet header
- FEC requires more bits because correcting is more difficult than detecting
- Both have limited capability (if too many errors, nothing can be done)

Block codes for error control

- n bit
- k bit for user data
- n-k parity bit
- \(2^k\) possibile combinations
Examples of error protection

• Parity bit
  – detects bit errors in odd numbers
  – 1 error more likely than k errors

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0</td>
<td>1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 1 0</td>
<td>0 1</td>
</tr>
</tbody>
</table>

• Repetition code
  – Majority decision
  – Error correction possible
  – Requires more overhead

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
</tbody>
</table>

Examples of error protection

• Row and column parity
  – Corrects single bit errors
  – Requires more overhead than row (or column parity) but not as the repetition code

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0 0 1</td>
<td>0 1</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 0 0</td>
<td>0 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 1 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 0 0</td>
<td>1 0</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0 1 1</td>
<td>0 0</td>
</tr>
</tbody>
</table>
**Packet header**

- Parity (or error protection) bits are available in PDU header (PCI).
- Classical example of parity bit are CRC (cyclic redundancy check).
- Coding theory: select the code that given an amount of bit best suits the specific channel.

**ARQ**

- Joint control over a connection of:
  - Error (losses)
  - Sequence
  - Flow
- Packet numbering bit are needed in the PDU PCI.
- Basic idea: send a packet and wait for a confirmation (ACK)
  - If not received, retransmit.
ARQ

- Three ARQ flavours
  - Stop and wait (Alternating bit)
  - Go back N
  - Selective repeat
- We focus on a unidirectional communication
- In bidirectional connections, two independent mechanisms run in the two opposite directions

![Diagram showing TX and RX with PCI and SDU](image)

ARQ

- Data packet header
  - Parity bit for error control over data AND header
  - N(T), transmitted packet number
  - Addresses

![Diagram showing TX and RX with V(T) and V(R)](image)
ARQ

- Acknowledgement packet header
  - Parity bit on the header only
  - N(R), next expected packet number (convention)
  - Addresses

Stop and wait

- Transmitter
  - Makes a copy of the PDU (for possible retransmission) and stores it in the tx buffer
  - Sends the PDU
  - Activates a timer (timeout)
  - Waits for the ACK
  - If timeout expires before ACK reception, repeats the transmission of the PDU, reactivating the timeout
Stop and wait

• Transmitter
  – On ACK reception
    • Checks ACK correctness
    • Checks sequence number
    • If the ACK number refers as expected to the next PDU that has to be transmitted, enables the transmission cycle for the next PDU
    • Otherwise, the ACK is ignored

Stop and wait

• Receiver
  – On PDU reception
  – Checks PDU correctness
    • If the PDU is correct (regardless the sequence number) sends the proper ACK
  – Check PDU sequence number
    • If the PDU has the expected number, it is moved to the higher layer protocols for further processing
Stop and wait

- Initialization
  - Connection set up at connection opening
    - TX and RX agree on protocol parameters
  - $V(T) = 0$ at the transmitter
  - $V(R) = 0$ at the receiver

---

Stop and wait

- PDU transmission with $N(T) = V(T)$
- Timer is started
Stop and wait

• PDU reception
  – Check for correctness
  – Check sequence: \( N(T) = V(R) \) ?

Stop and wait

• Increase \( V(R) \)
• Send and ACK with \( N(R) = V(R) \)
Stop and wait

- ACK reception
  - Sequence control
    - \( N(R) = V(T) + 1 \) ?
  - Stop the timer
  - Erase PDU copy from the buffer

\[
\begin{align*}
\text{TX} & : V(T) = 0 \\
N(R) = 1 & \\
\text{RX} & : V(R) = 1
\end{align*}
\]

Stop and wait

- Increase \( V(T) \)

\[
\begin{align*}
\text{TX} & : V(T) = 1 \\
N(R) = 1 & \\
\text{RX} & : V(R) = 1
\end{align*}
\]
Stop and wait

• $V(T) = 1$ at the transmitter
• $V(R) = 1$ at the receiver
Stop and wait

• PDU reception
  – Check for correctness
  – Check sequence: \( N(T) = V(R) \) ?

\[
\begin{array}{ccc}
& \rightarrow & \\
\text{TX} & \quad \text{N(T)=1} \quad \text{SDU} & \quad \text{RX} \\
V(T)=1 & \quad & V(R)=1
\end{array}
\]

Stop and wait

• Increase \( V(R) \)
• Send and ACK with \( N(R) = V(R) \)

\[
\begin{array}{ccc}
& \rightarrow & \\
\text{TX} & \quad \text{N(R)=2} & \quad \text{RX} \\
V(T)=1 & \quad & V(R)=2
\end{array}
\]
Stop and wait

- ACK reception
  - Sequence control
    - $N(R) = V(T) + 1$?
  - Stop the timer
  - Erase PDU copy from the buffer

Stop and wait

- Increment $V(T)$
Stop and wait

- $V(T) = 2$ at the transmitter
- $V(R) = 2$ at the receiver

Numbering PDUs is mandatory
Numbering scheme is cyclic
  - fixed number of bit in the header and finite counter size
One PDU sent per cycle
  - The cycle duration is named round trip time (RTT)
Stop and wait

- One single bit for numbering: Alternating bit protocol

```
<table>
<thead>
<tr>
<th>TX</th>
<th>RX</th>
</tr>
</thead>
<tbody>
<tr>
<td>V(T)</td>
<td>V(R)</td>
</tr>
</tbody>
</table>
```

- $V(T) = 0$  $N(T) = 0$  $V(R) = 0$
- $V(T) = 0$  $N(R) = 1$  $V(R) = 1$
- $V(T) = 1$  $N(T) = 1$  $V(R) = 1$
- $V(T) = 1$  $N(R) = 0$  $V(R) = 0$
- $V(T) = 0$  $V(R) = 0$

Alternating bit protocol

```
<table>
<thead>
<tr>
<th>TX</th>
<th>RX</th>
</tr>
</thead>
<tbody>
<tr>
<td>V(T)</td>
<td>V(R)</td>
</tr>
</tbody>
</table>
```

- $V(T) = 0$  $N(T) = 0$  $V(R) = 0$
- $V(T) = 0$  $N(R) = 1$  $V(R) = 1$
- $V(T) = 1$  $N(T) = 1$  $V(R) = 1$
- $V(T) = 1$  $N(R) = 0$  $V(R) = 0$
- $V(T) = 0$  $V(R) = 0$
Reception of a PDU with errors (PDU loss)

- TX
  - V(T)
- RX
  - V(R)

V(T) = 0  N(T) = 0  V(R) = 0
V(T) = 0  N(R) = 1  V(R) = 1
V(T) = 1  N(T) = 1  V(R) = 1
V(T) = 1  N(R) = 0  V(R) = 1

- Timeout value setting is critical

Copyright Gruppo Reti – Politecnico di Torino

COMPUTER NETWORKS - Window protocols - 35

COMPUTER NETWORKS - Window protocols - 36
Reception of an ACK with errors

- Need to number PDUs
  - Otherwise the PDU duplication would not be detected
Non sequential channel

**TX**

V(T)

------------

**RX**

V(R)

V(T) = 0

N (T) = 0

V(R) = 0

V(T) = 0

N (R) = 1

V(R) = 1

V(T) = 0

N (T) = 0

V(R) = 1

V(T) = 1

N (R) = 1

V(R) = 1

V(T) = 1

N (T) = 0

V(R) = 1

V(T) = 0

N (R) = 0

V(R) = 0

V(T) = 0

N (R) = 1

V(R) = 0

V(T) = 1

V(R) = 0
Non sequential channel

\[
\begin{align*}
V(T) = 0 & \quad N(T) = 0 & \quad V(R) = 0 \\
V(T) = 0 & \quad N(R) = 1 & \quad V(R) = 0 \\
V(T) = 1 & \quad N(T) = 1 & \quad V(R) = 0 \\
V(T) = 1 & \quad N(R) = 0 & \quad V(R) = 0 \\
V(T) = 0 & & \quad V(R) = 0
\end{align*}
\]
Non sequential channel

- Over a non sequential channel the alternating bit protocol may not be able to correctly detect
  - PDU losses
  - PDU duplication

Non sequential channel: modulo 4 numbering
Non sequential channel: modulo 4 numbering

\[ V(T) = 1 \quad N(T) = 1 \quad V(R) = 1 \]
\[ V(T) = 1 \quad N(R) = 2 \quad V(R) = 2 \]
\[ V(T) = 2 \quad N(T) = 2 \quad V(R) = 2 \]
\[ V(T) = 2 \quad N(R) = 3 \quad V(R) = 3 \]
\[ V(T) = 3 \quad N(T) = 3 \quad V(R) = 3 \]
\[ V(T) = 3 \quad N(R) = 0 \quad V(R) = 0 \]
\[ V(T) = 0 \quad N(T) = 0 \quad V(R) = 0 \]
\[ V(T) = 0 \quad N(R) = 1 \quad V(R) = 0 \]
Non sequential channel: modulo 4 numbering

• Protocol is blocked
  – Needed mechanism to restart (reset) the connection

• Problems occurrence become unlikely if using
  – More bit to number PDUs and ACKs
  – A maximum lifetime for PDUs and ACKs
Go Back N

• The stop and wait protocols in absence of errors permits to successfully send one packet per RTT (round trip time)
• If the RTT is large, the transmitter throughput (amount of bit/s that can be sent) is limited by the protocol
• To increase throughput, in the Go Back N version the transmitter can send up to $k$ PDUs before stopping while waiting for ACK reception

Transmission window

• The transmission window $W_T$ is the maximum number of PDUs that can be transmitted (in sequence) without receiving any ACK
• Limited by the amount of buffer space reserved at the transmitter
• It is also the maximum number of PDUs (or ACKs or PDUs+ACKs) propagating over the channel or in the network at a given time
Receiver window

- The receiver window $W_R$ defines the sequence of PDUs that the receiver is willing to store in a given time.
- Limited by the amount of memory space reserved for the connection at the receiver.

Transmission window

- Correctly Acked PDUs
- PDUs waiting for an ACK
- PDUs that can be transmitted
- PDUs that cannot be transmitted

$W_T$
### Receiver window

**Set to one in Go Back N**

![](image)

- Correctly ACKed PDUs
- Expected PDU
- Out of sequence PDU

---

### Go Back N

- **The transmitter**
  - Sends up to $N = W_T$ PDUs, storing a copy of them in the TX memory
  - Starts a timer (usually one timer per window)
    - The timer is reset ad any PDU transmission
  - Waits for ACK reception
  - If timeout expires before ACK reception
    - It repeats the transmission of ALL PDUs that have not been correctly ACKed
    - “Goes back” by N positions
Go Back N

- The receiver, when a PDU is received
  - Checks PDU correctness
    - If the PDU is correct, sends the ACK
  - Checks the PDU sequence number
    - If the sequence number is the number of the first PDU that has not been received yet, it moves the PDU to higher layer protocols for further processing

ACK semantics

- Several ACK meaning can be used
  - Selective ACK:
    - Each ACK notifies the correct reception of a single PDU
    - ACK(n) means "I received PDU n"
  - Cumulative ACK
    - Each ACK notifies the correct reception of ALL PDUs with a sequence number smaller than n
    - ACK(n) means "I received all PDUs up to n"
  - Negative ACK (NAK)
    - Request to retransmit a specific packet
    - NAK(n) means "retransmit PDU n"

- TX and RX must agree on ACK semantic
Piggybacking

- For bi-directional information flows it is possible to include the ACK number on the PDU header travelling in the opposite direction
- This technique is known as piggybacking
PDU numbering

- PDU numbering is cyclic
  - With k bit in the header
  - Numbering modulo $2^k$

- Example
  - 3 bit
  - $W_R=1$
  - $W_T=3$

Go Back N

- Transmitter is significantly more complex than the Stop and wait transmitter
  - More memory
  - Timer management is more complex
  - PDU numbering more difficult
- The receiver is the same (window size 1)
- It is possible to group ACK (thanks to the cumulative meaning)
  - One ACK for several PDUs
  - Need a timer at the receiver
- Constraint on the maximum window size $W_T < 2^k$
Selective repeat

• We define the selective repeat (SR) as the window protocol where the receiver window is larger than one
  – In Go back N, only in sequence PDUs can be received
• Accepting out of sequence PDUs may increase performance
• Usually transmitter and receiver window are of the same size
• Several options exist to implement the protocol, which differ for:
  – Cumulative vs selective ACK
  – Single timer per PDU or per window
  – Slightly different behaviour of Tx and RX
• We focus on cumulative ACK and one timer per window

Selective repeat receiver window

PDUs to be passed to higher layer protocols

PDU that were correctly ACKed and moved to higher layer protocols for further processing

\[ W_R \]

PDUs that can be accepted

Out of sequence PDUs that cannot be accepted
Selective repeat

• The transmitter
  – Sends up to $N = W_T$ PDUs, storing a copy of them in the TX memory
  – Starts a timer
    • The timer is reset at any PDU transmission
  – Waits for ACK reception
  – If timeout expires before ACK reception
    • It repeats the transmission of ALL PDUs that have not been correctly ACKed

Selective repeat

• The receiver, upon PDU reception
  – Checks PDU correctness
  – Check sequence number
  – If the PDUs is correct and in sequence
    • moves the PDUS (and all other PDU in sequence) to higher layer protocols
  – If the PDU is correct but not in sequence
    • It stores the PDU if it is within the window
    • It discards the PDU if it is out of the window
  – Sends an ACK for the last PDU correctly received in sequence
Selective repeat - Observations

- If a single PDU is lost, the SR behaves as the Go back N in terms of
  - Throughput
  - Channel occupation
- Performance advantages if
  - RTT < window transmission time
    - The new ACK permits to move ahead the TX window without the need to retransmit all PDUs in the transmission window
  - For multiple PDU losses
    - It is enough to receive a single copy of each packet
- By modifying the TX behavior (constraining it to retransmit only the first lost packet in the window) the channel occupation in case of losses can be reduced
  - One PDU per RTT is recovered
- Better performance when adopting selective ACKs
  - Sending a copy of the whole receiver window gives the TX the exact picture of the receiver status (requires large ACKs)

Selective repeat

- Constraint on the window size to ensure correct behavior
  - $W_T + W_R \leq 2^k$ where $k$ is the number of bit used to number PDUs and ACKs
- In the following example we highlight potential problems when this constraint is violated
  - $W_T = 3$, $W_R = 2$, $k = 2$
  - Hypothesis of one timer per window
  - We will show that
    - 1 PDU will be accepted twice
    - 1 PDU will be erroneously dropped
In absence of errors, the throughput of a window protocol is ruled by:

\[
\min \left\{ \frac{\text{transmission}_\text{window}}{\text{RTT}}, \frac{\text{link}_\text{speed}}{\text{RTT}} \right\}
\]

For a given RTT there is an optimal window size that maximize throughput.

Larger windows translate in a RTT increase.

“Short” connections get higher throughput for a given window size.

To control TX throughput, it is possible to act on:
- RTT (delaying ACK transmission)
  - May induce retransmissions
- Window size