CSC0056: Data Communication

Lecture 05: Point-to-Point Protocols (2)

Instructor: Chao Wang 王超

Department of Computer Science and Information Engineering

NATIONAL TAIWAN NORMAL UNIVERSITY

Course information

- Instructor: Chao Wang 王超 (https://wangc86.github.io/)
  - Email: cw@ntnu.edu.tw
  - Office: Room 511, Applied Science Building, Gongguan Campus
  - Office hours: Wednesdays and Fridays, 10am-noon, or by appointment
- Course website: https://wangc86.github.io/csc0056/
  - Homework submission: via NTNU Moodle (https://moodle.ntnu.edu.tw/)
- Course meetings: Mondays 9:10-12:10 in C007, Gongguan Campus

Acknowledgement: Some slides' materials in this course are borrowed with permission from the 2014 edition of the course taught by Prof. Yao-Hua Ho 賀耀華
Figures are obtained from the textbook available at http://web.mit.edu/dimitrib/www/datanets.html

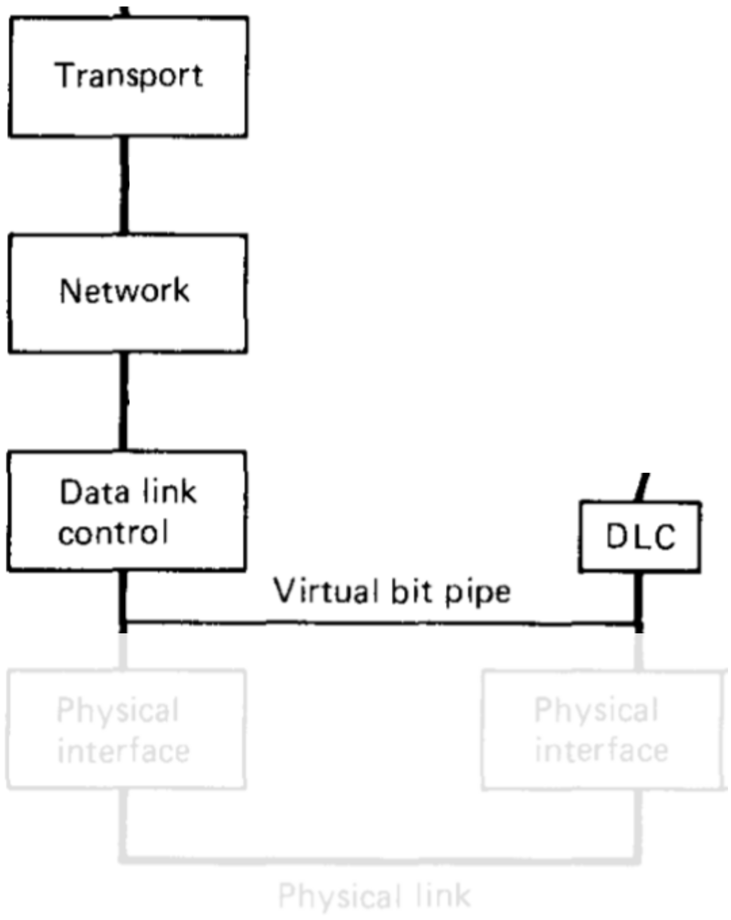
Outline of lecture05

- Lecture04 was cancelled due to typhoon

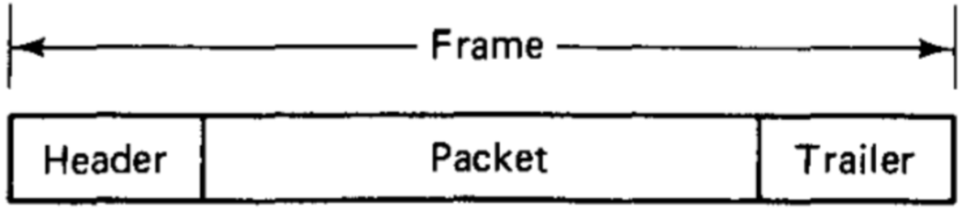
- Data retransmission strategies and analysis (data link layer)
    - Stop-and-wait ARQ
    - Go-back-N ARQ
    - Selective repeat

- Point-to-point protocols at higher layers

Data transmission viewed from the data link layer



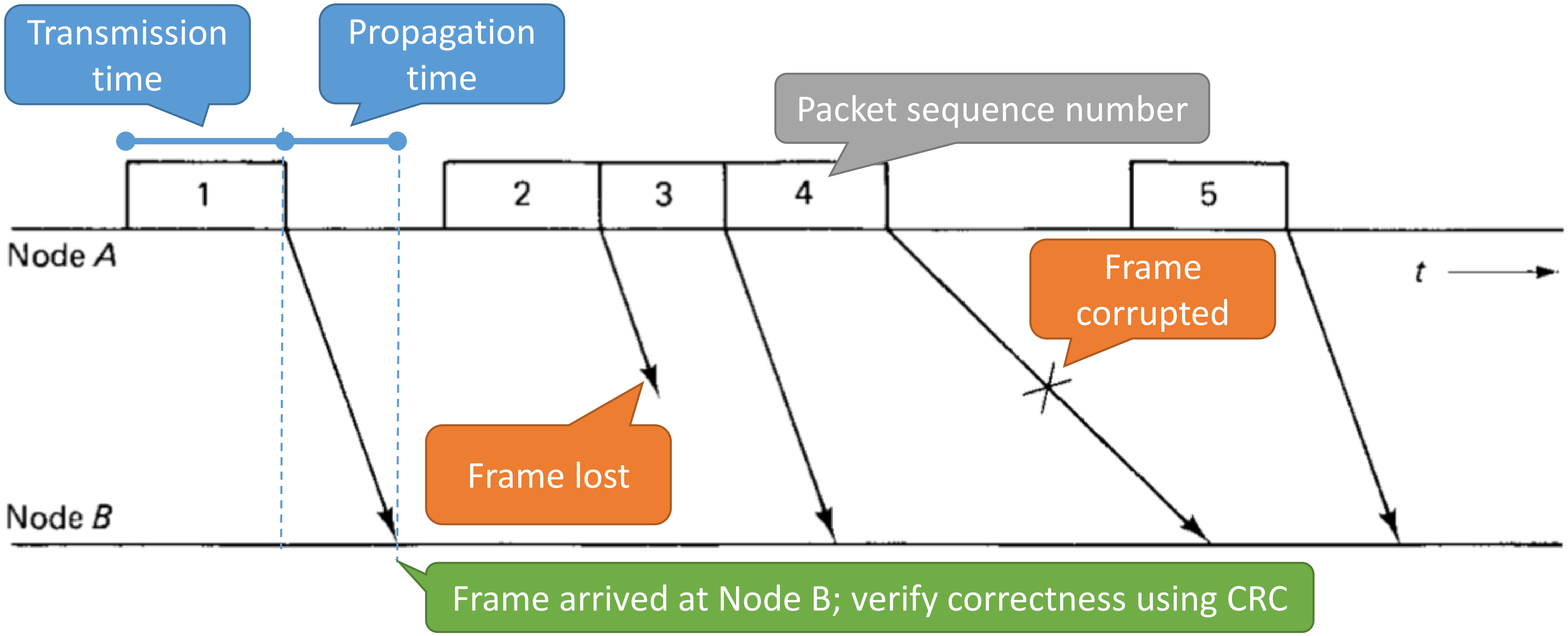
- Data link layer creates a *frame* by appending a header and trailer to each of the *packet* from the network layer



- Data link layer views the underlying point-to-point channel as an *unreliable* virtual bit pipe
- A frame could be lost or corrupted (i.e., contains errors) in the virtual bit pipe

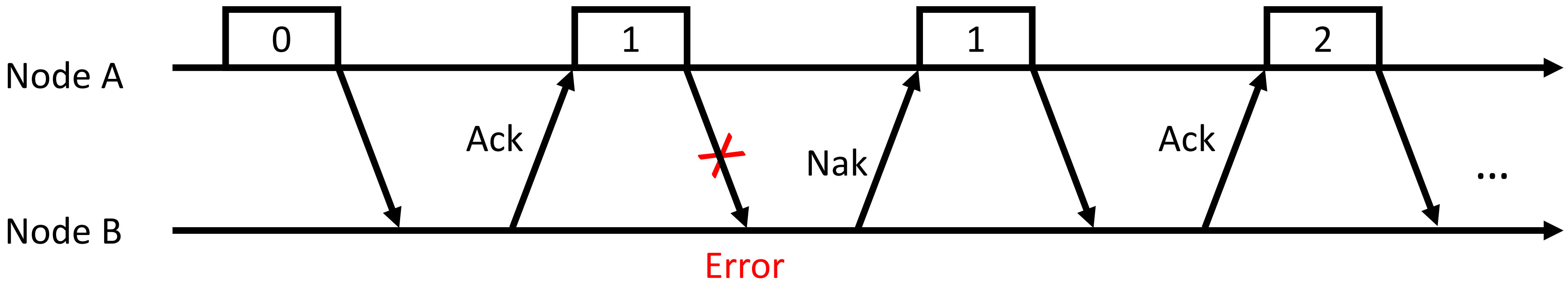
ARQ: Automatic Repeat Request

- Model and definition, assuming Node A sends frames to Node B



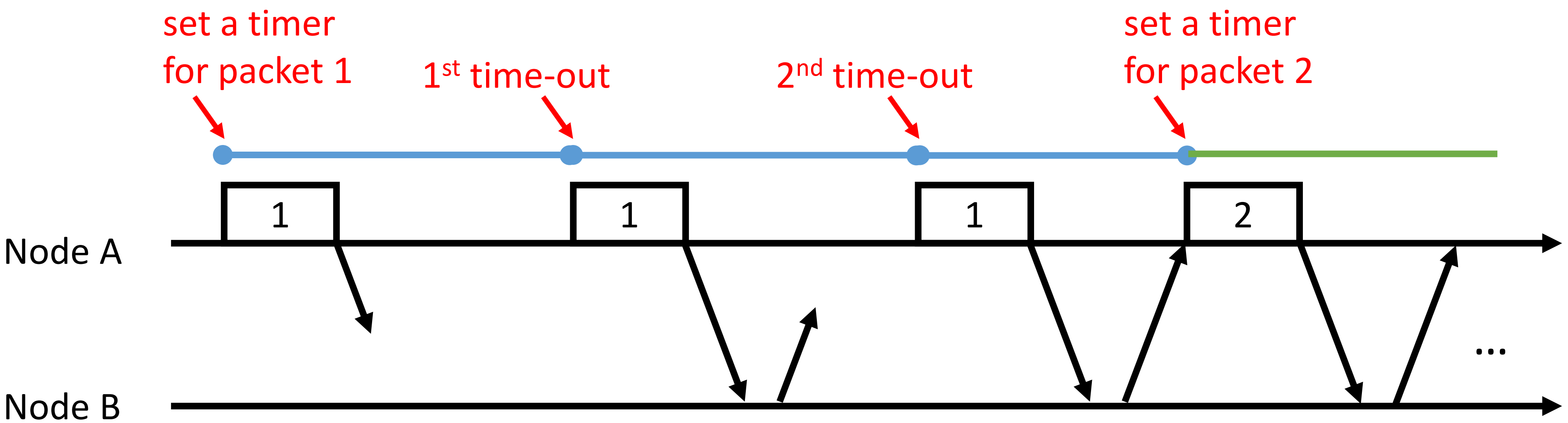
An initial idea of Stop-and-Wait ARQ

- Node A sends a new packet only if it received an acknowledgement from Node B (called an Ack)
- Node B sends an Ack if the received frame is error-free; otherwise, it sends a negative acknowledgement (called an Nak)



The need for time-out and re-transmission

- With no time-out, Node A may wait forever (for either Ack or Nak) if a frame is lost in either direction during transmission…
- Upon a time-out, Node A will re-transmit the same packet
- Design issue: what is a desirable length of a timer? (see later slides)

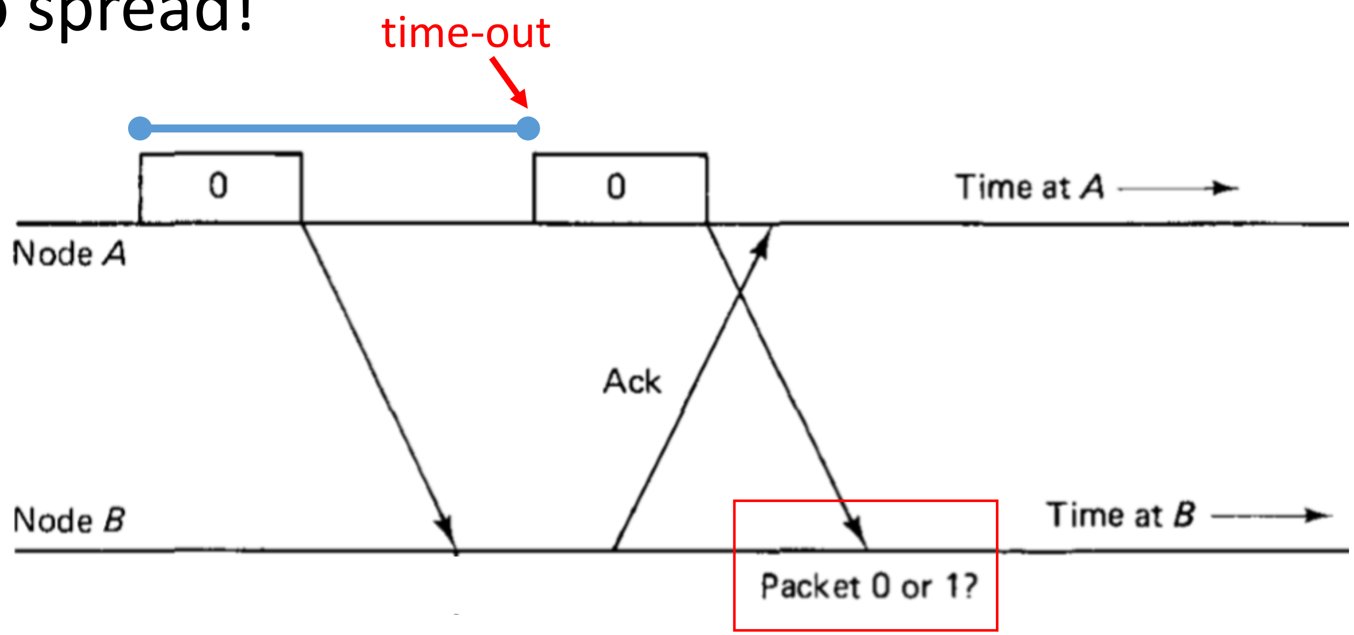


The need for packet sequence number

- Node B otherwise may not be able to tell if a frame contains a new packet or if a frame contains a previous packet

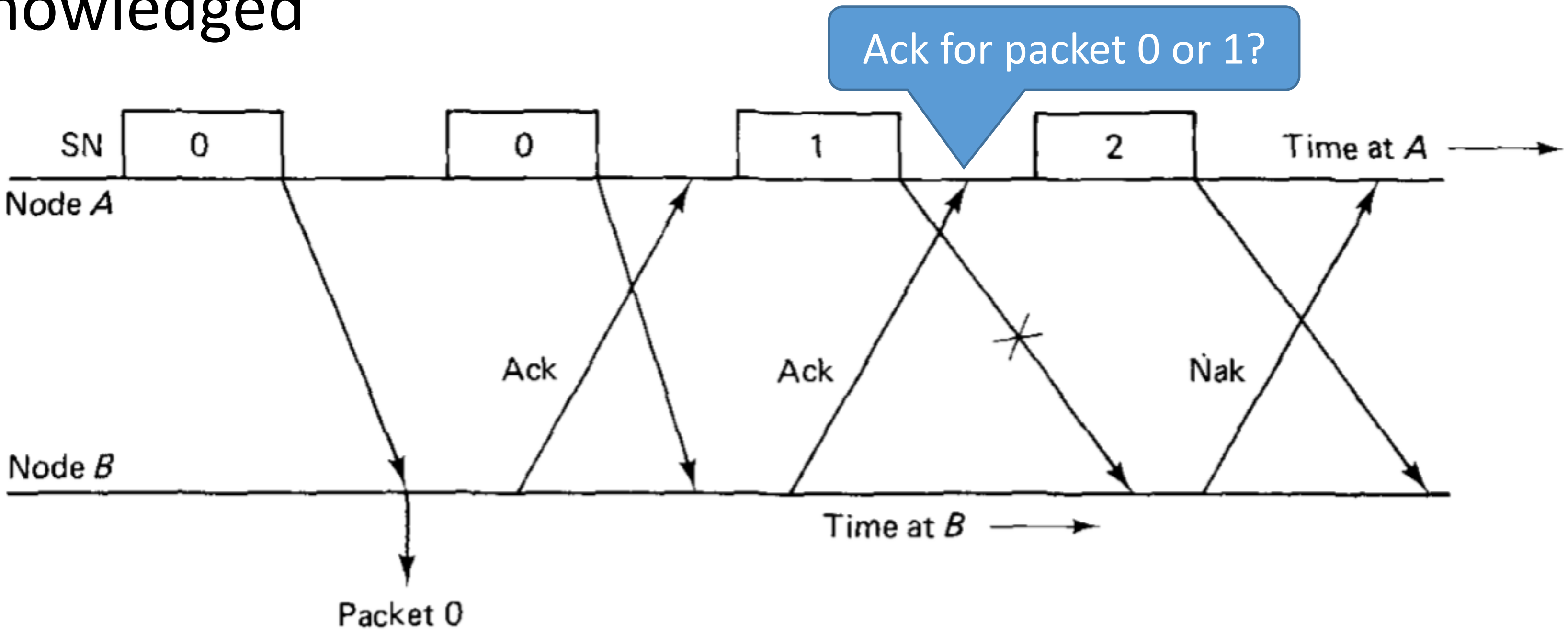
✓In general, in designing distributed algorithms, a challenge is that information takes time to spread!

Correctness and efficiency are two critical aspects in design of distributed algorithm.



The need for distinct acknowledgements

- With simple Ack/Nak, Node A might not be able to tell which packet Node B acknowledged

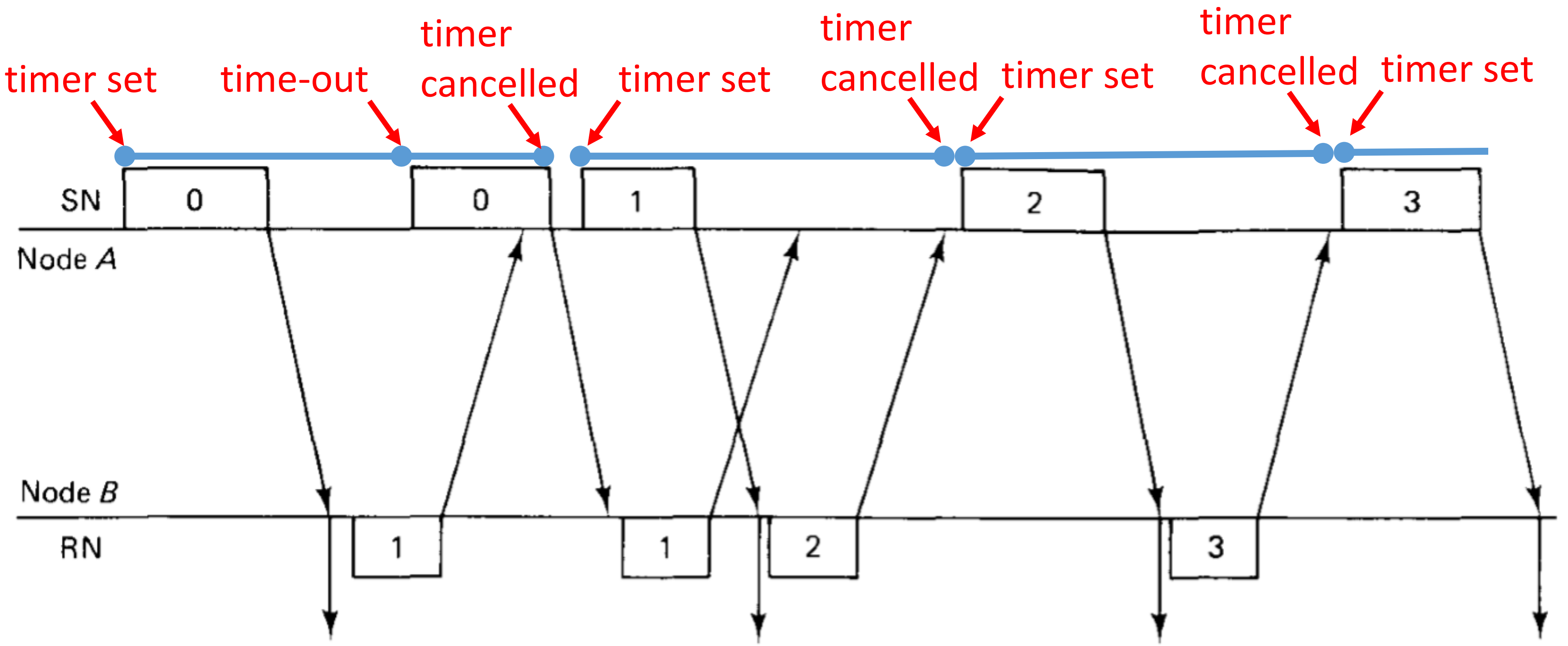


- Solution: Node B sends the number of frame currently *awaited*

A working version of stop-and-wait ARQ

- Using time-out, sequence number (SN), and request number (RN) to coordinate Nodes A and B

(Read the textbook for the assumptions and the algorithm)

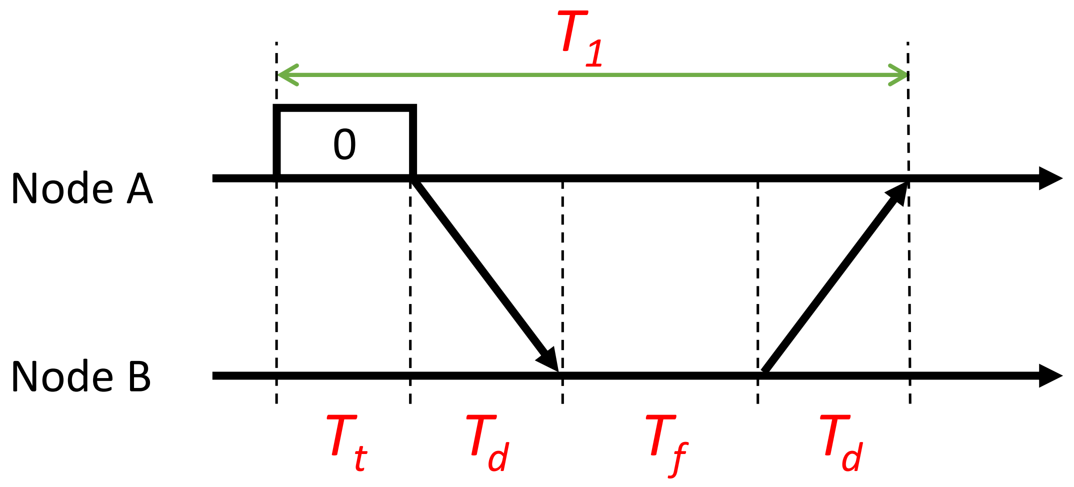


Analyzing efficiency of the stop-and-wait

- There are several ways to look at efficiency
- Way #1 (latency): let T be the total time between the transmission of a packet and reception of its Ack

✓ If error-free, $T = T_1 = T_t + 2T_d + T_f$

T_t : packet transmission time
T_d : delay in propagation and processing
T_f : feedback transmission time



Efficiency in terms of latency

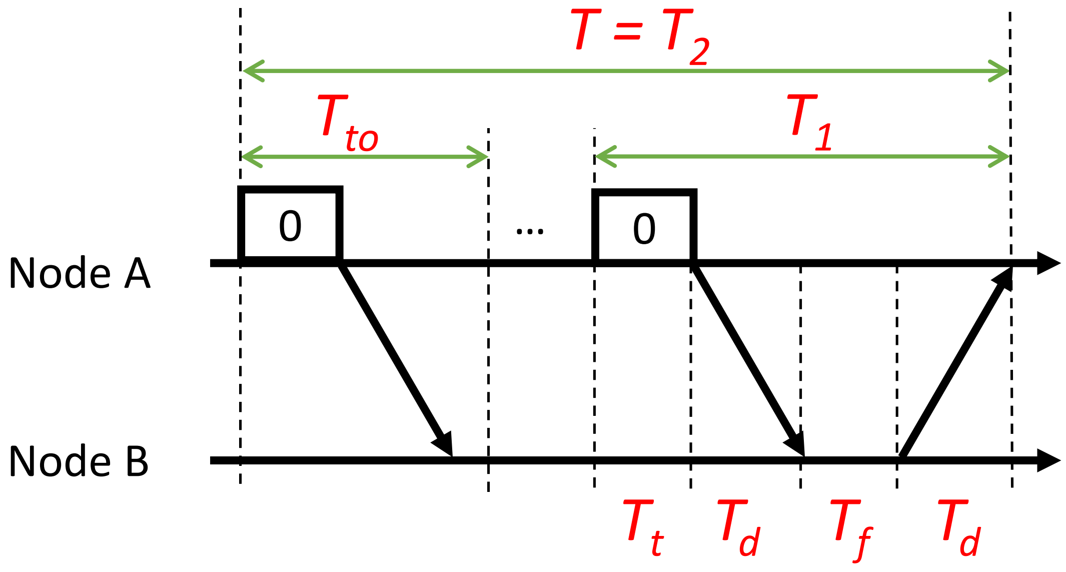
- ✓ With chances of error,

  $T = T_2 = T_1 + T_{to}(EX\text{-}1)$

  $EX = q^{-1}$

  (Refer to the note for a proof)

- ✓ **Latency depends on T_{to} and q**

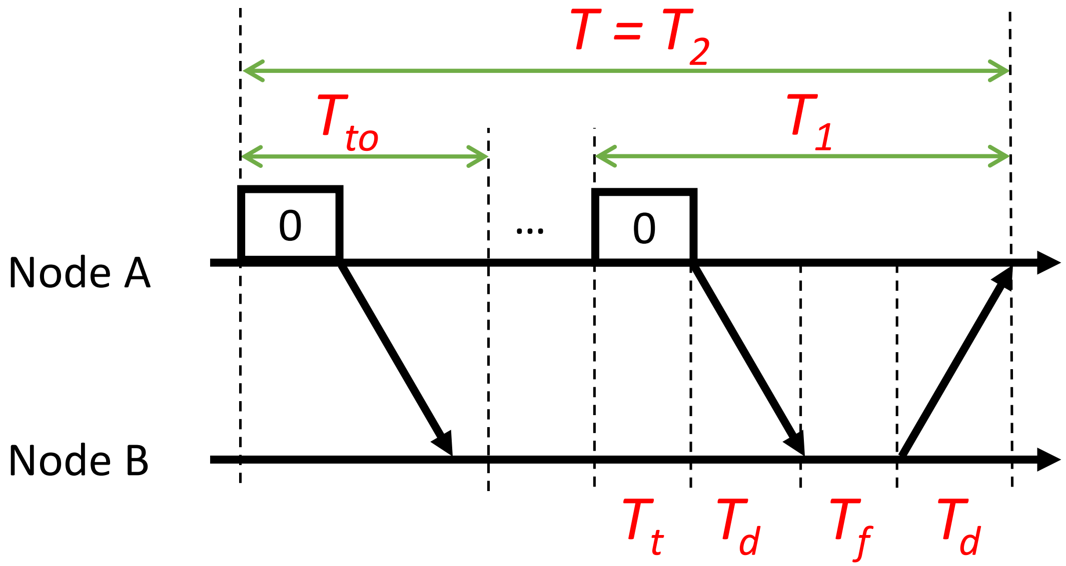


EX : the expected number of times a packet must be transmitted for a stop-and-wait system
q : the probability that a packet is correctly received and acked on a given transmission
T_{to} : length of a timer (i.e., duration before a time-out)
T_t : packet transmission time
T_2 : total time between the transmission of a packet and reception of its Ack

Effects of the length of timer

- Setting $T_{to} > T_1$, we have

  $T_2 = T_1 + T_{to}(q^{-1}-1) > q^{-1}T_1$



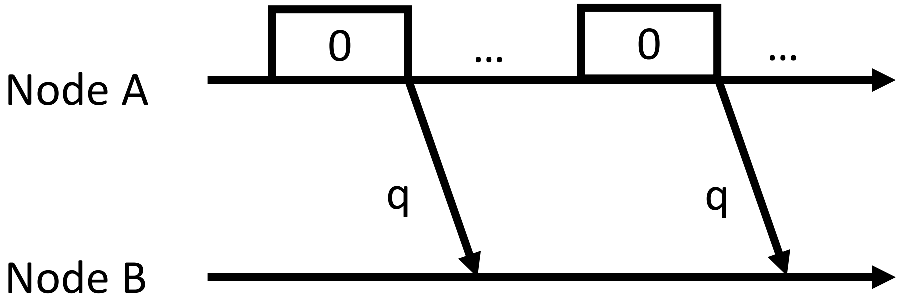
- setting $T_{to} < T_1$, make sense?
  - will cause additional redundancy
  - but then we have $T_2 < q^{-1}T_1$, which gives us a way to bound the latency.
  - The percentage of increase in latency is $(T_2 - T_1)/T_1 < (1 - q)/q$.

- Exercise: plug in different values of q and see how T changes

Another way to look at efficiency of the stop-and-wait : link goodput

- Way #2 (link goodput): (way #1 at slide #11)

  Define efficiency E as the expected number of packets delivered to node B per frame from A to B



✓ **Efficiency E = $\underline{1/EX = q}$ = $1\text{-}p$**

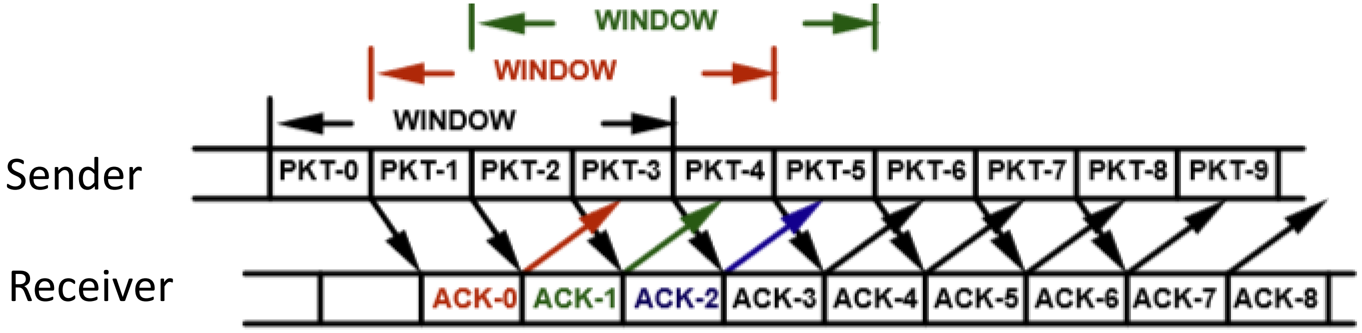
(Similar to the proof that we used for analyzing latency)

✓ **for all ARQ protocols, $E \leq 1\text{-}p$**

EX : the expected number of transmissions before the packet is correctly received *p* : the probability of frame error *q* : the probability of no frame error (***q=1-p***)

Improving over stop-and-wait: **the go-back-N protocol** (*widely used*)

- The **stop-and-wait** protocol is inefficient since the sender cannot send a new packet before the previous one is acknowledged
- The **go-back-N** protocol allows transmission of new packets before earlier ones are acknowledged
  - Maintain a *sliding window* where the sender can send packets that are within the window (range) of packets
  - The window advances as the sender received Acks for earlier packets



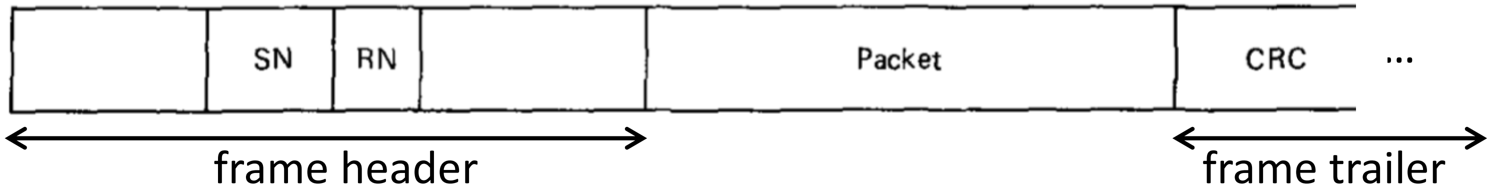
Algorithm of the go-back-N protocol

- Sender sets window size=*N*, and will not set packet *i+N* until it has received the acknowledgement for packet *i*
  - Let SN_{min} be the latest value of RN obtained from the receiver. The window includes packet indices ranging from SN_{min} to $SN_{min}+N-1$
  - The sender goes back and re-transmits packet SN_{min} when either it has reached the end of the window or a *time-out* occurred (the timer in this case is set for the time to send a full window of packets).
- Receiver operates just like that in the stop-and-wait protocol
  - Cannot accept packet out of order
  - Sending request number RN=*i+1* means that it acknowledged all packets up to and including *i*

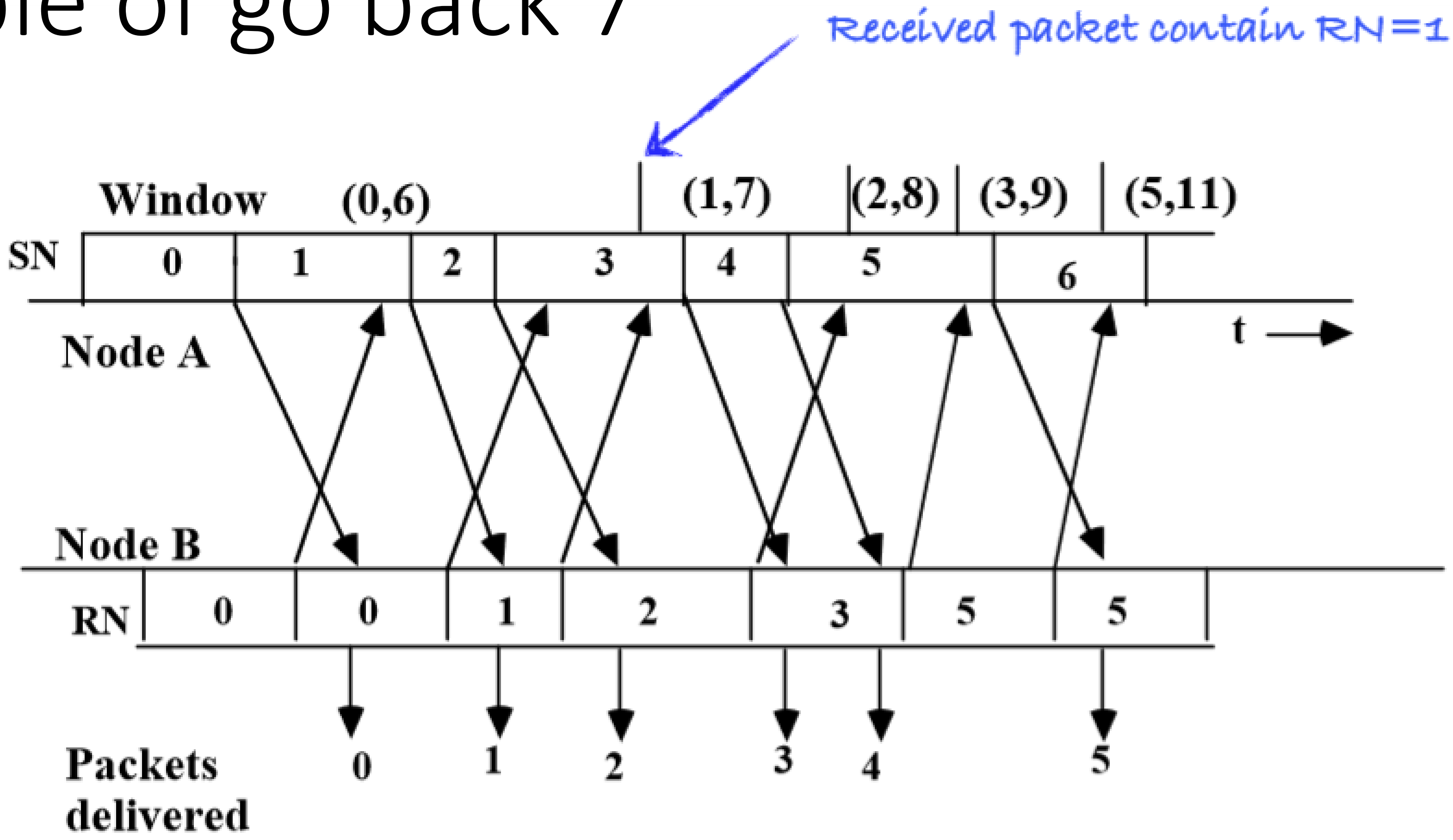
Consideration in sending request number RN

- In both stop-and-wait and go-back-N protocols, there are two ways a receiver may send request number RN:
    1. If traffic is unidirectional, the receiver sends a non-data frame containing the RN value
    2. If traffic is bi-directional, the receiver may *piggyback* RN in a frame going in the opposite direction; use non-data frame also, in the absence of traffic

  ✓ Either way, the receiver must send out RN within bounded time

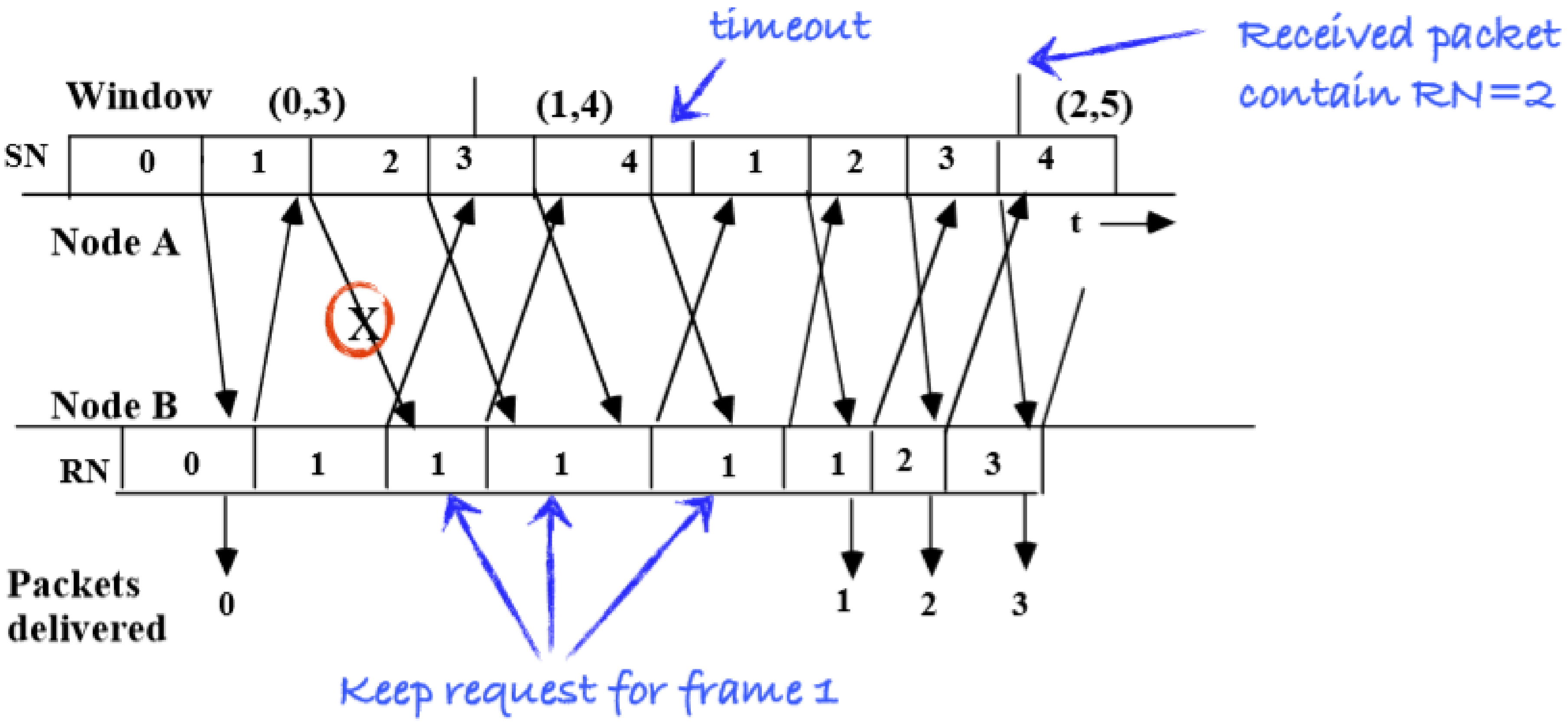


Example of go back 7



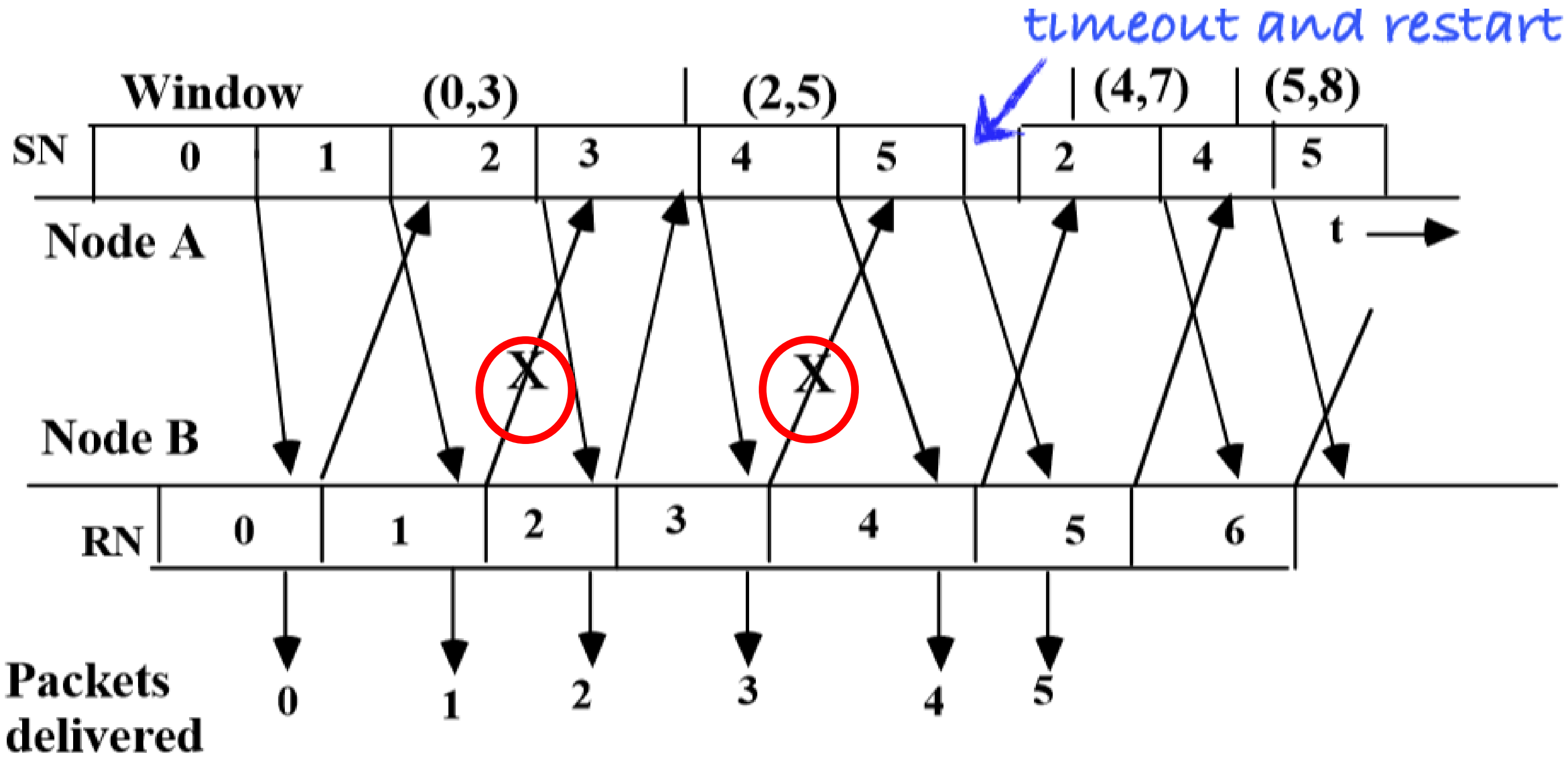
- Note that packet RN-1 must be accepted at Node B before Node B can send a request for packet RN

Handling transmission error (go back 4)



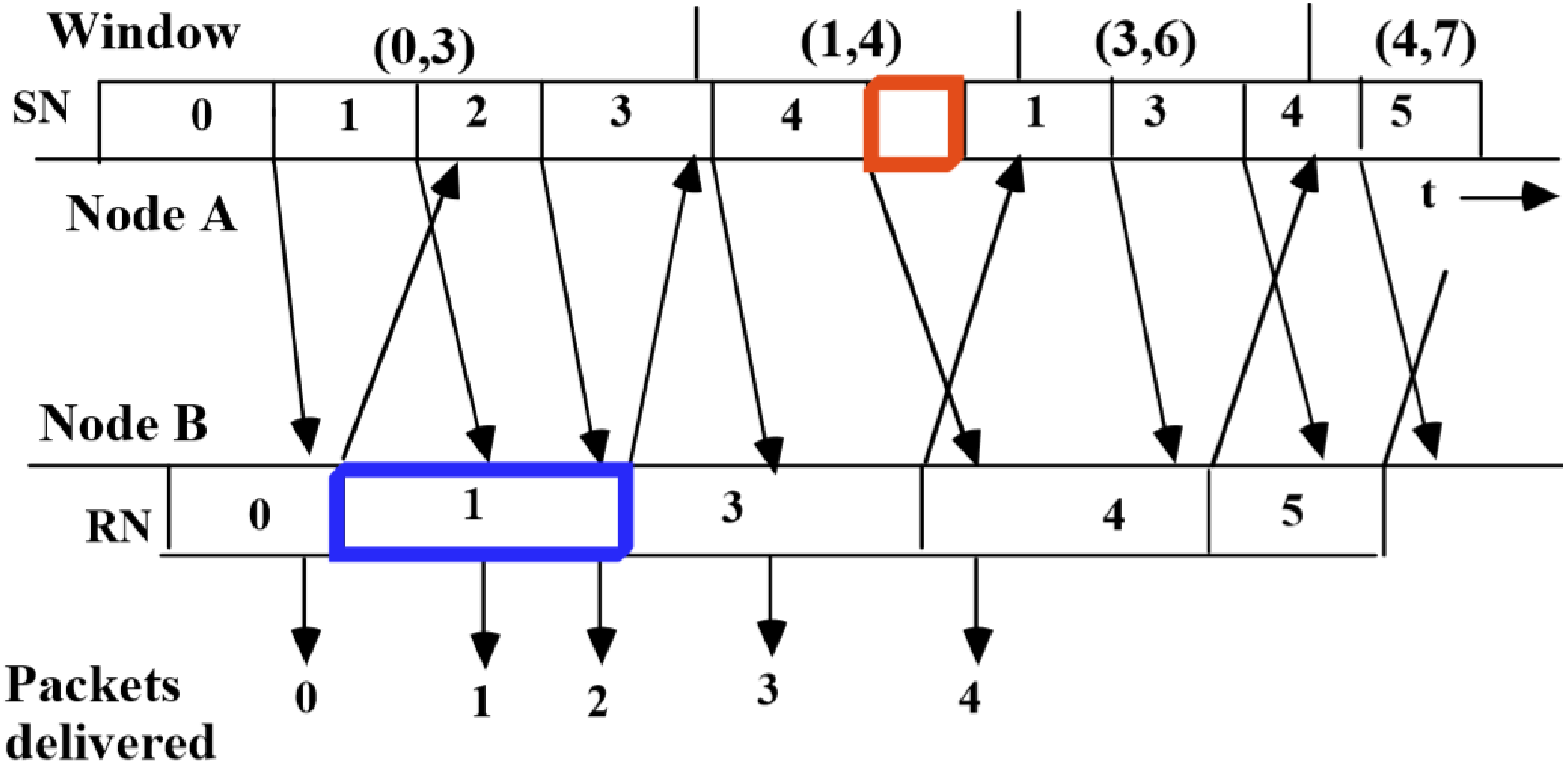
- After an error, the sender has to re-transmit the entire set of packets in a window

Handling feedback error (go back 4)



- Newer acknowledgements may prevent some re-transmissions :)
  - Examples: RN=2 and RN=4

Effect of long frames (go back 4)



- Long feedback frames may cause a sender to wait or go back, since they slow down acknowledgements

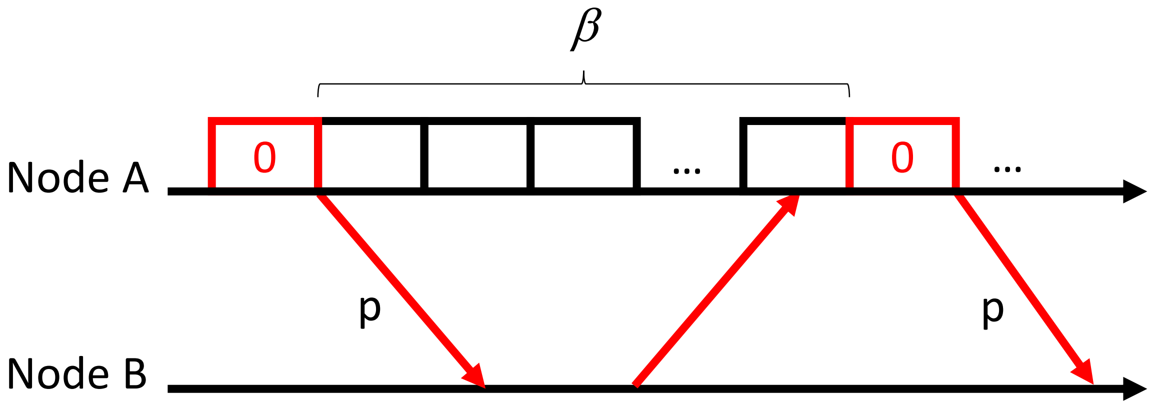
Efficiency of the go back N protocol (link goodput)

- Define efficiency E as the expected number of packets delivered to node B per frame from A to B

✓ **Efficiency E = $1/EX$**

= $\underline{(1-p)/(1+p\beta)}$

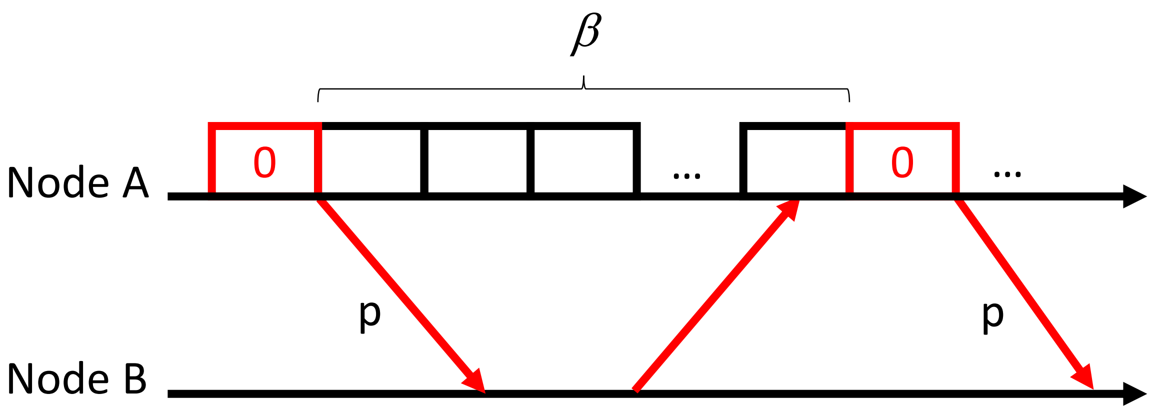
(See the lecture note)



EX : the expected number of transmitted frames from A to B per successfully accepted packet at B
p : the probability of frame error
β : the expected number of transmitted frames from A to B between the transmission of a given frame and the reception of feedback about that frame

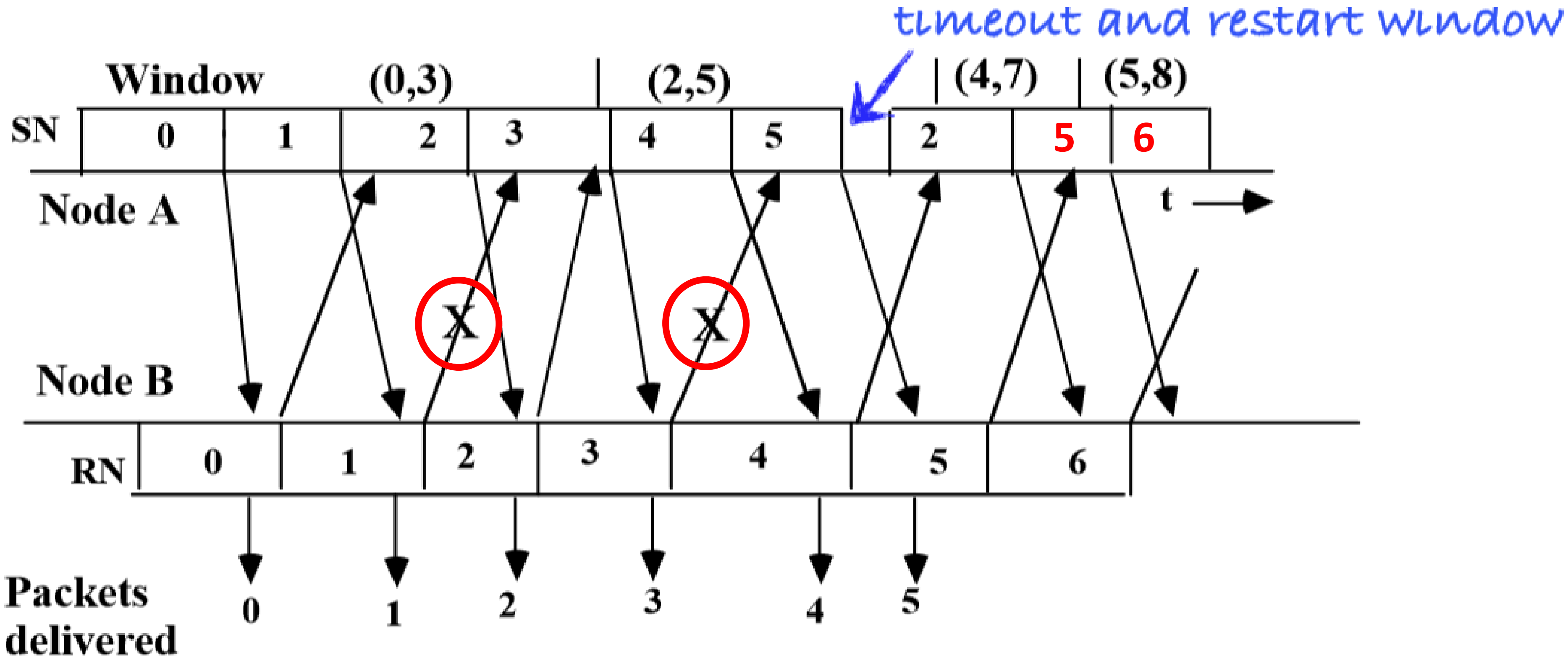
Improving over go-back-N: **the selective repeat protocol**

- Go-back-N has improved over stop-and-wait in terms of waiting time before transmission
- But in terms of link goodput *E*
  - In stop-and-wait: $E = 1\text{-}p$
  - In go-back-N: $E = (1\text{-}p)/(1+p\beta)$



- The selective repeat protocol:
  - Request re-transmission only for those packets that are not correctly received
  - In other words, ideally β will decrease over time, which gives $E \cong 1\text{-}p$

Example:
selective repeat with window size = 4



- Acknowledged packets will not need to be re-sent

Requesting for re-transmission in selective repeat

- Implicit
  - The receiver acknowledges every good packet, packets that are not acknowledged before a time-out are subject to re-transmission

- Explicit
  - An explicit NAK can request ret-transmission of just one packet
  - This option can expedite the re-transmission

✓One or both approaches are used in practice

Rules of the selective repeat protocol

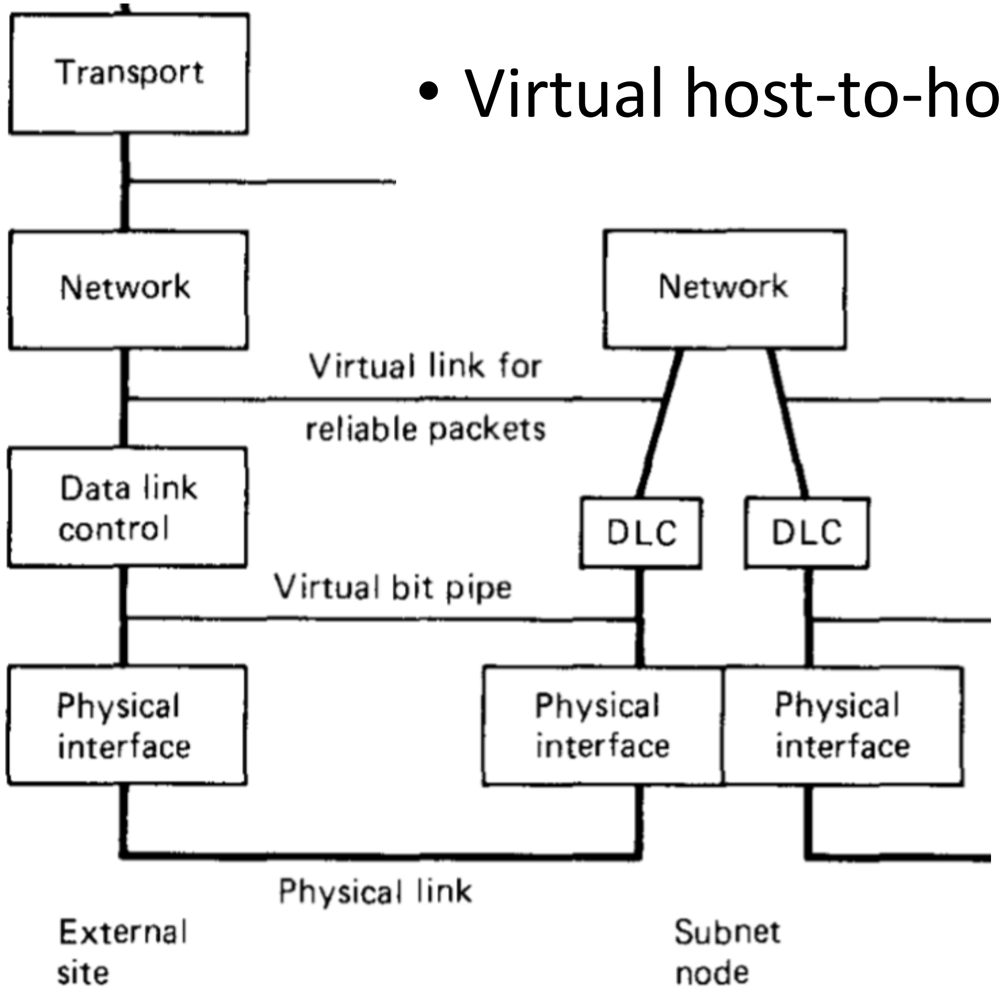
- Window protocol just like the go-back-N
  - Window size *W*
- Sender can transmit new packets as long as their number is with *W* of all unacknowledged packets
- Sender re-transmits unacknowledged packets after a time-out
  - or upon a NAK if NAK is employed
- Receiver acknowledges all correct packets
- Receiver in addition must buffer all correct packets until they can be deliver in order to the higher layer

Outline of lecture05

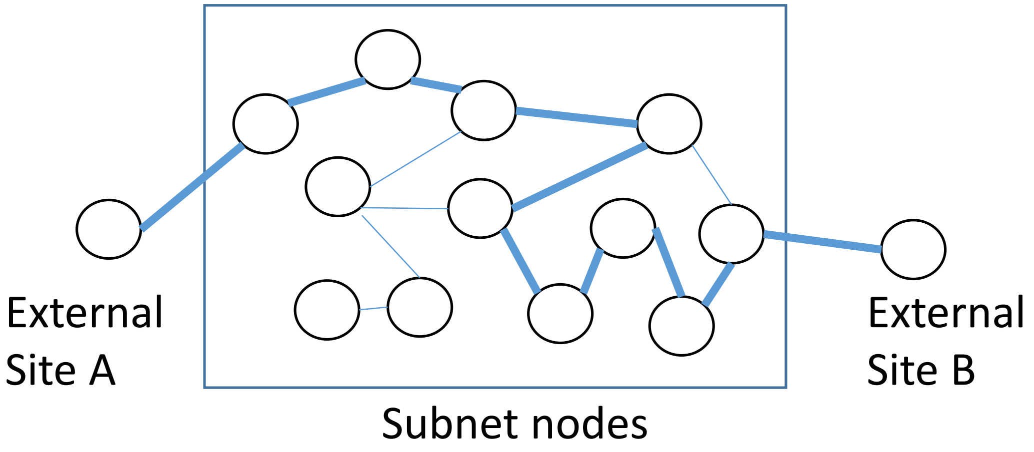
- Data retransmission strategies and analysis (data link layer)
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective repeat

- **Point-to-point protocols at higher layers**

Concept review of the network layer



- Virtual host-to-host packet service to the higher layers
- Each host (node) contains one network layer module
- Provide *routing* for the network:



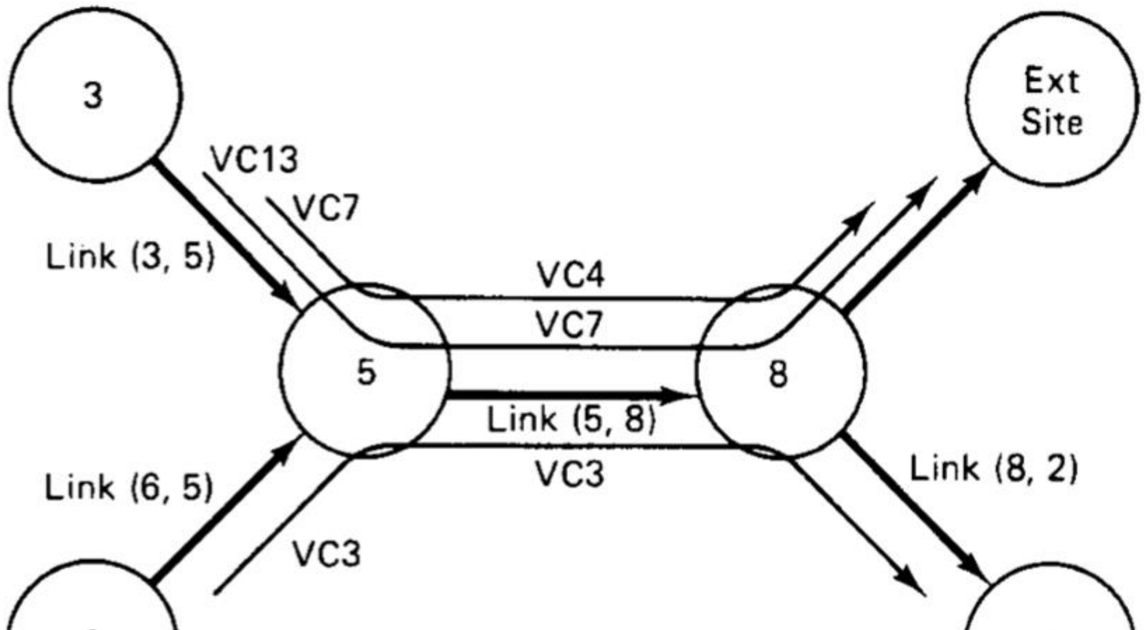
Session identification and addressing

- Definition of a *session*:

  A sequence of messages between two users over a network.

- At each node, need to distinguish packets of different sessions

- Example: virtual circuit routing
  - Use virtual channel numbers (VC#) to route packets over virtual circuits in the network



Node 5 table	
(3, 5) VC13	→ (5, 8) VC7
(3, 5) VC7	→ (5, 8) VC4
(6, 5) VC3	→ (5, 8) VC3

Error recovery at higher layers

- Conceptually very similar to ARQ at the data link layer
  - Point-to-point error recovery: two nodes and a link in between
  - End-to-end error recovery: two sites and a subnet in between
    - May operate on a go-back-n/selective repeat basis
- Key difference: at higher layers, packets might arrive out of order
  - Solution candidates:
    - Enforce ordering
    - Use a large modulus number for packet numbering
    - Destroy a packet after some time
- Error recovery in TCP (Sec. 2.9.3)