A small problem, better solved concurrently

Jayadev Misra

Department of Computer Science
University of Texas at Austin

May 7–11, 2018
WG 2.3, Providence

To: Doug McIlroy

Friend, mentor, inspiration

Packet Reassembly Problem

- Video packets for a movie arrive along an input channel.
- Packet have consecutive sequence numbers, starting at 0.
- Incoming packets may be out of order, but by a bounded amount.
- Goal: Output the packets in the order of sequence numbers.

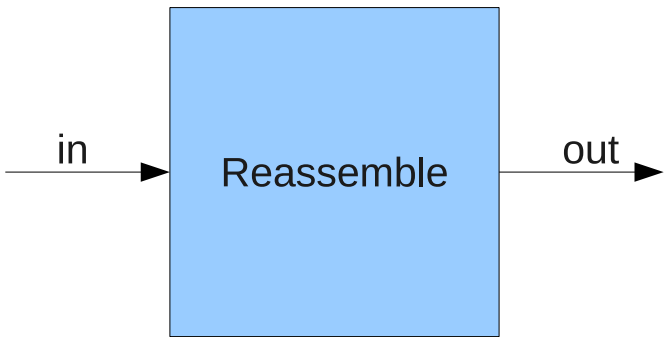

Figure: Packet Reassembler

Packet Reassembly Specification

- Channel in carries all packets, without duplication, in some order.
  Packet with sequence number i is at position p_i in channel in, $i \geq 0$.

- Given: For all i, $i \geq 0$, $|p_i - i| \leq k$, for some positive integer k. (S)

- Output the packets in order of sequence numbers along channel out.

Note: For $k = 0$, no reassembly required.

Conclude from (S)

For any i:

$p_i \leq i + k$	(S1)
$i \leq p_i + k$	(S2)
$i + 2 \times k \leq p_{i+2\times k} + k$	replace i by $i + 2 \times k$ in (S2)
$i + k \leq p_{i+2\times k}$	rewriting
$p_i \leq p_{i+2\times k}$	(S1) and above
$p_i \leq p_{i+d}$	using d for $2 \times k$
$p_i < p_{i+d}$	$i \neq i + d$ for $k > 0$, so $p_i \neq p_{i+d}$

Conclude:

Packets with equal sequence numbers modulo d are received in order.

Conclude from (S)

For any i:

$p_i \leq i + k$	(S1)
$i \leq p_i + k$	(S2)
$i + 2 \times k \leq p_{i+2\times k} + k$	replace i by $i + 2 \times k$ in (S2)
$i + k \leq p_{i+2\times k}$	rewriting
$p_i \leq p_{i+2\times k}$	(S1) and above
$p_i \leq p_{i+d}$	using d for $2 \times k$
$p_i < p_{i+d}$	$i \neq i + d$ for $k > 0$, so $p_i \neq p_{i+d}$

Conclude:

Packets with equal sequence numbers modulo d are received in order.

Program Strategy

- Packets with equal sequence numbers modulo d are received in order.
- Setup d internal channels, numbered 0 through $d-1$. On receiving packet with sequence number n store it in channel numbered $n \bmod d$.
- Then packets in each channel are in order.
- Output one item from each channel, in order starting at channel 0. Repeat this step.

Packet Reassembly Program

$def\ \ reassembly(read, write, d)\ =\ \ --$ d must be positive
$--$ "read" reads the next item along input channel;
$--$ "write" writes on the output channel.

$val\ \ ch\ =\ \ Table(d, lambda(\_) = Channel())$
$--$ Create an array ch of d channels.

$def\ \ input()\ =\ \ read()\ \ >(n, v)>\ ch(n\%d).put(v)\ \ \gg input()$

$def\ \ output(i)\ =\ \ ch(i).get()\ \ >v>\ write(v)\ \ \gg output((i+1)\%d)$

$input() \mid output(0)$ $--$ Goal expression

Note: $n\%d$ is n mod d.

A Variation

- A few packets may actually violate constraint (S).
- Modify the solution to account for straggler packets.