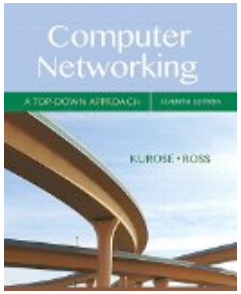
COMP 375: Lecture 15

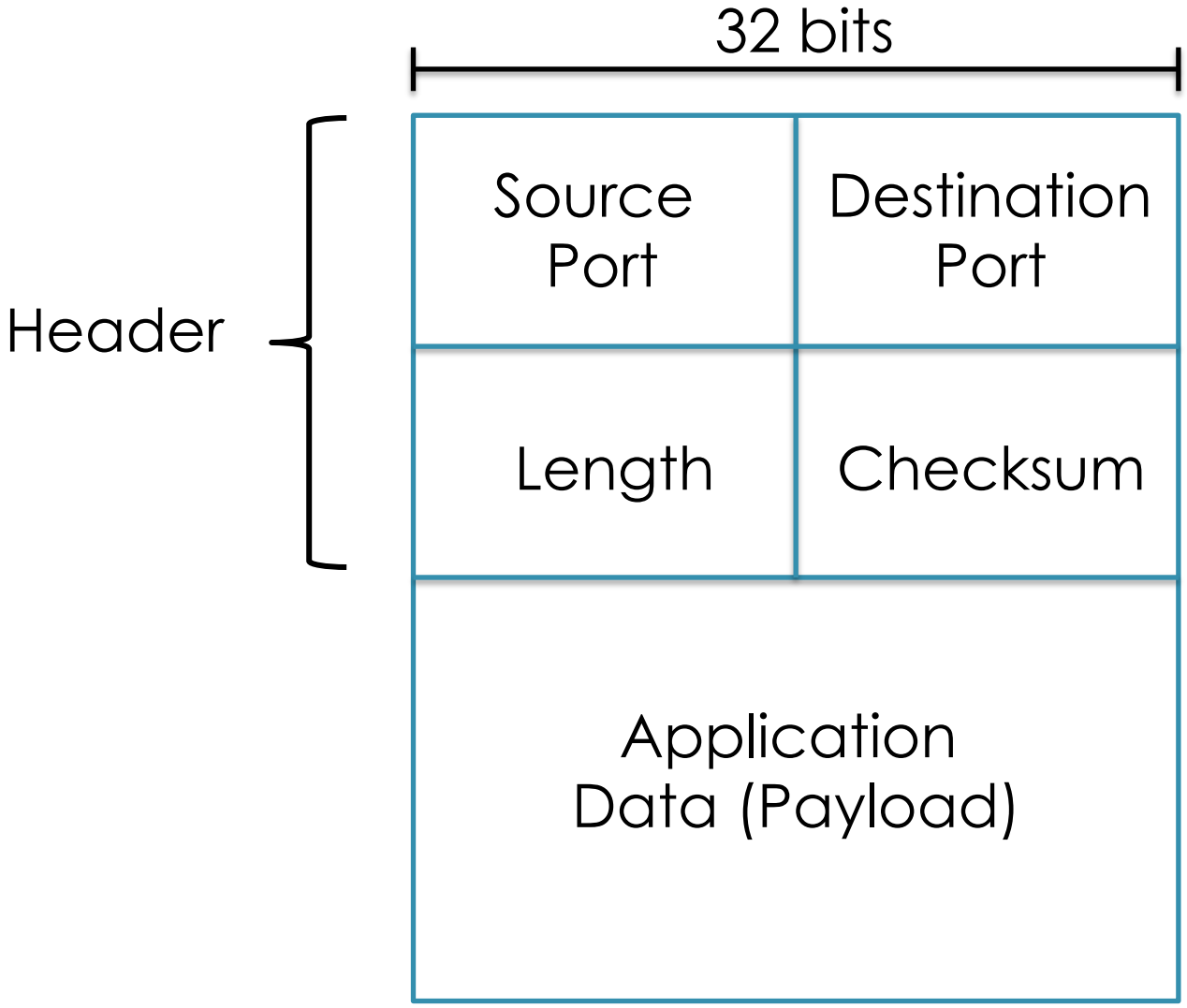


- **News & Notes:**
  - Midterm #1 in class Monday
  - Project #2 due @ 10PM
    - 1 project buck for 24 hour extension, 2 project bucks extends until Tuesday @ 10PM
  - Cool new website: www.hackterms.com
- **Reading (Wed, Mar. 7)**
  - (*Tentative*) Sections 3.4.{2-4}

Section 3.3

UDP

UDP is **connectionless**, with **best effort** service and **minimal overhead**.



UDP's **checksum** is designed to detect transmission errors.

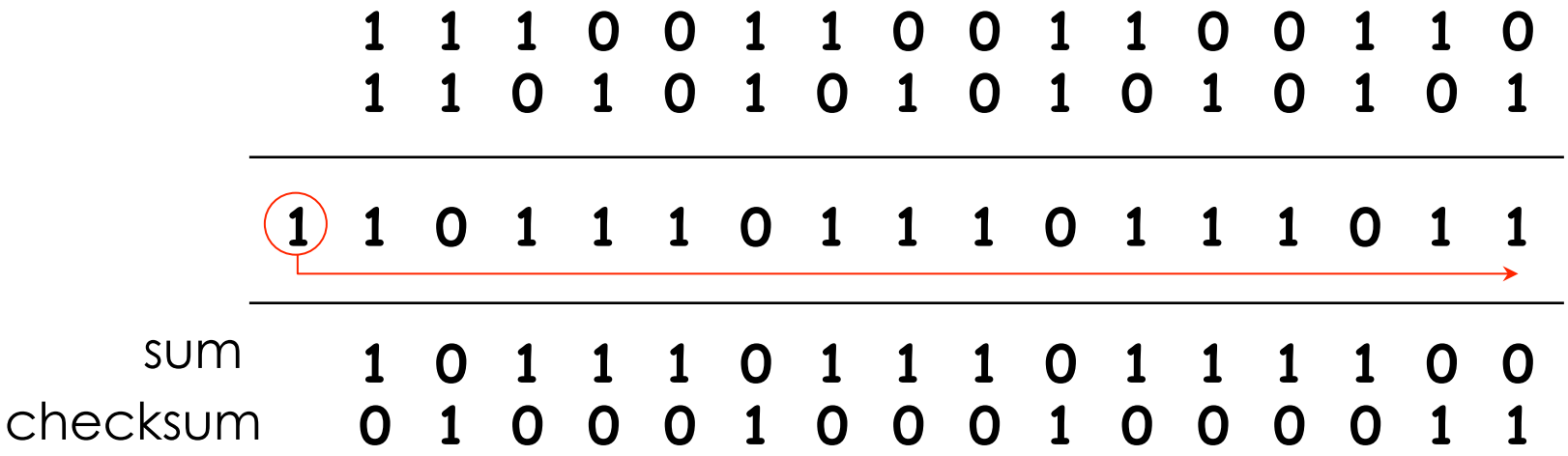
- Checksum Calculation:
  1. Divide data into 16-bit chunks
  2. Calculate the sum of all of these chunks
  3. Take the 1's complement of sum

One's Complement is simply the bitwise not (~) operator.

1111000011110000

0000111100001111

What is the checksum of the UDP message 0xE666D555?



Note: When adding numbers, a carryout from the most significant bit needs to be added to the result.

Verifying the checksum follows a similar process to calculating it.

- Validation Algorithm:
  1. Divide received data into 16-bit chunks
  2. Calculate the sum of all these chunks
  3. Add the sum to the received checksum
     - If result is not 0xFFFF, there were errors
     - Otherwise, …?

Does sum + checksum == 0xFFFF guarantee there were no transmission errors?

A.	Yes!
B	No!

There are three classes of programs that benefit from using UDP.

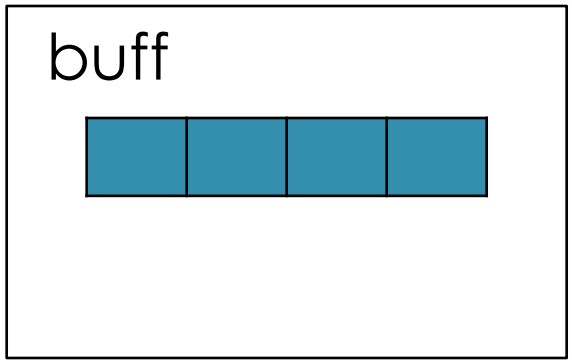
1. Latency sensitive
2. Error correction unnecessary
3. Communicating with *lots* of others

What if you want something more reliable than UDP, but faster/not as full featured as TCP?

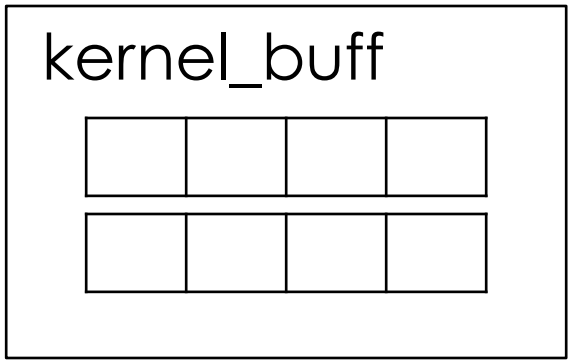
A.	Sorry, you're out of luck. ☹
B.	Write your own transport protocol.
C.	Add in the features you want at the application layer.

The kernel (OS) buffers data before sending it out over the network.

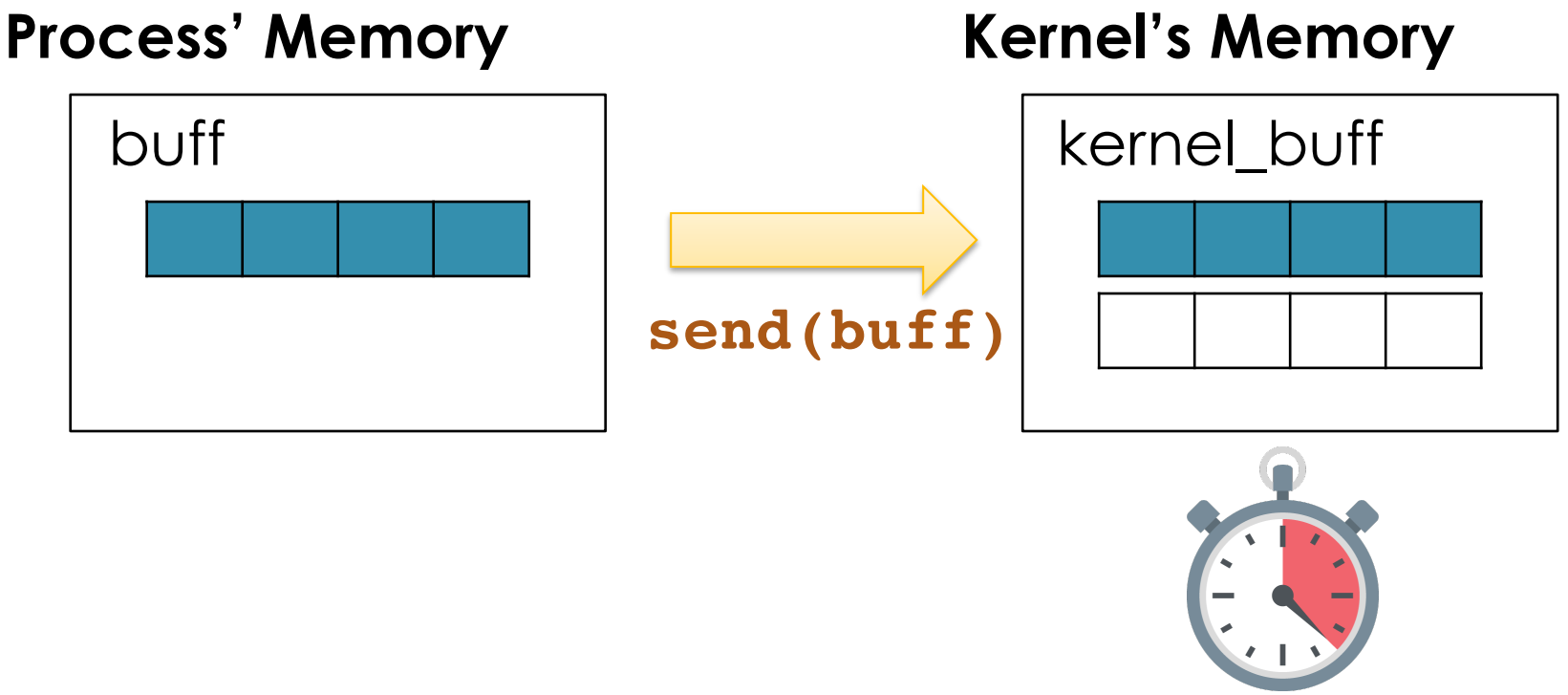
Process' Memory



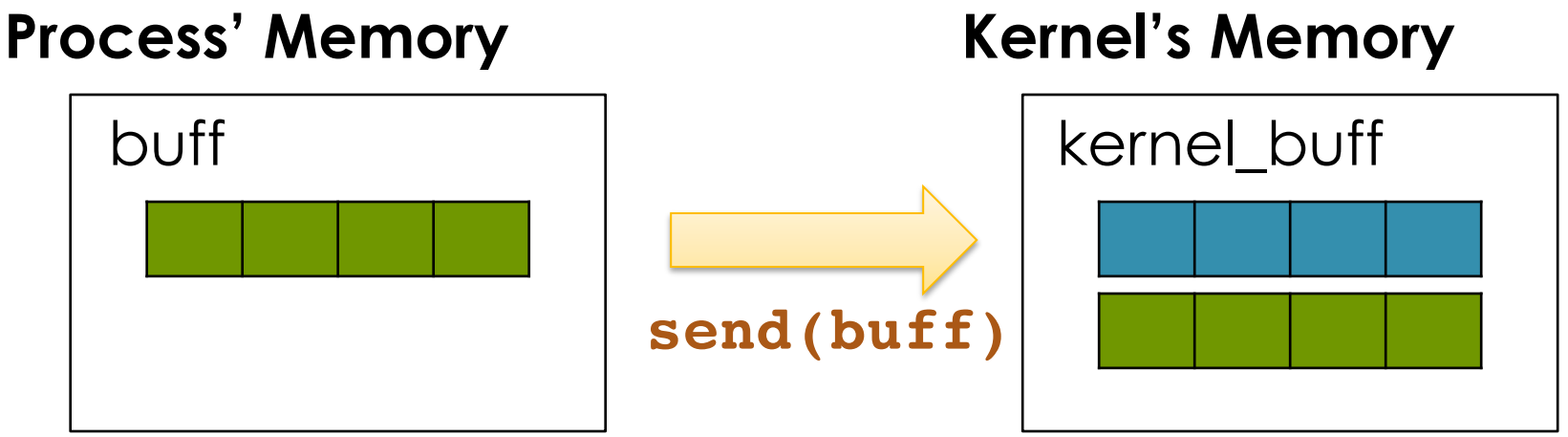
Kernel's Memory



The kernel (OS) buffers data before sending it out over the network.

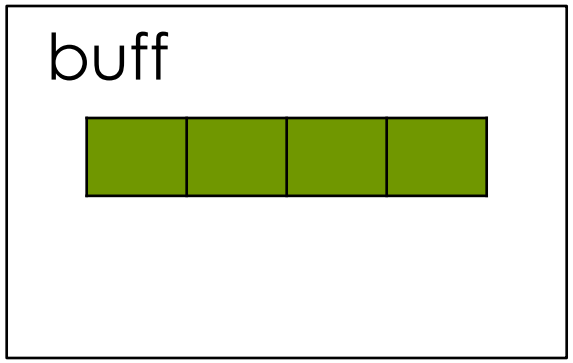


By waiting, can reduce number of network transmissions.

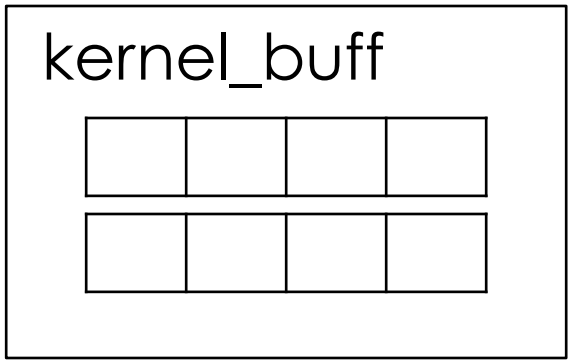


By waiting, can reduce number of network transmissions.

Process' Memory



Kernel's Memory



Send over network

TCP blocks (waits) if kernel buffer is full. Should UDP do the same?

A.	**Yes**. It needs to and therefore it should.
B.	**Yes**. It doesn't need to, but it might be useful.
C.	**No**. It does not need to and should not do so.

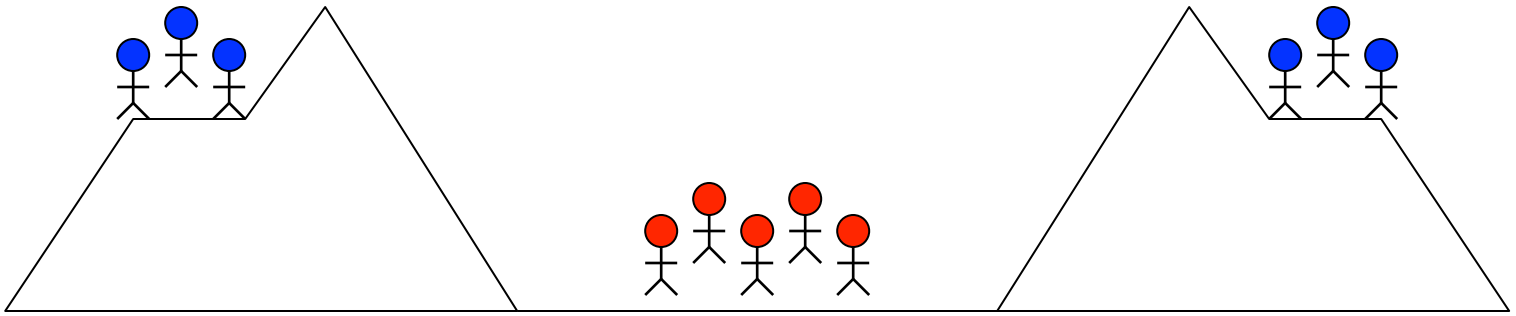
Project tip: Split your HTTP responses into multiple sends.

```
void sendFile(int sock, path file)
{
  size_t fsize = fs::file_size(file);
  sendOKHeader(sock, fsize);
  readAndSendFileData(sock, file);
}
```

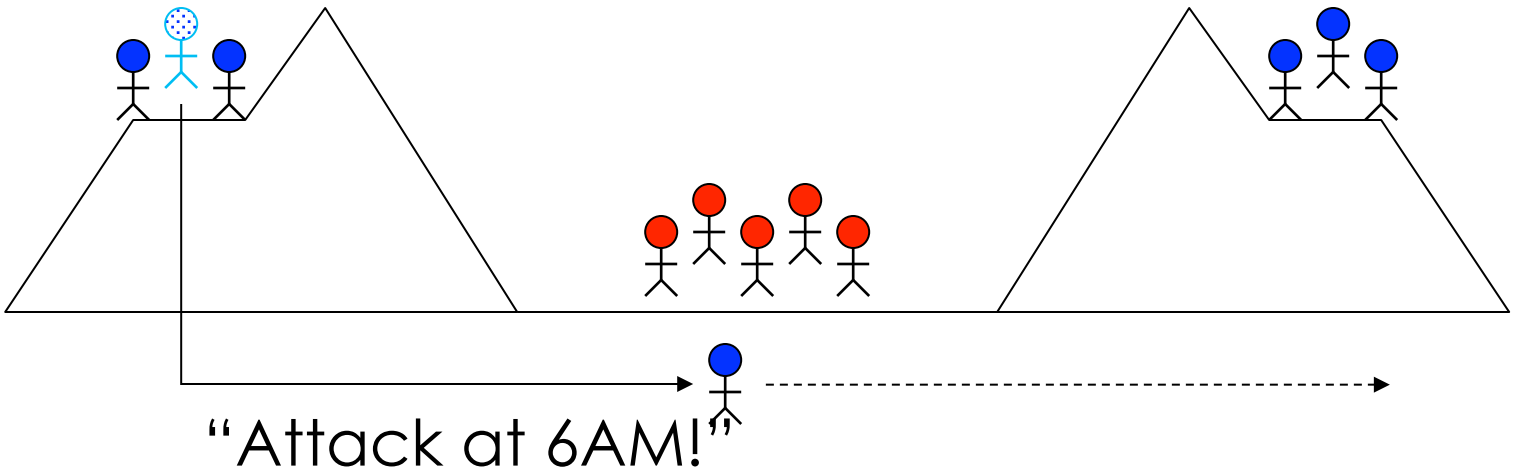
Section 3.4

RELIABLE DATA TRANSFER

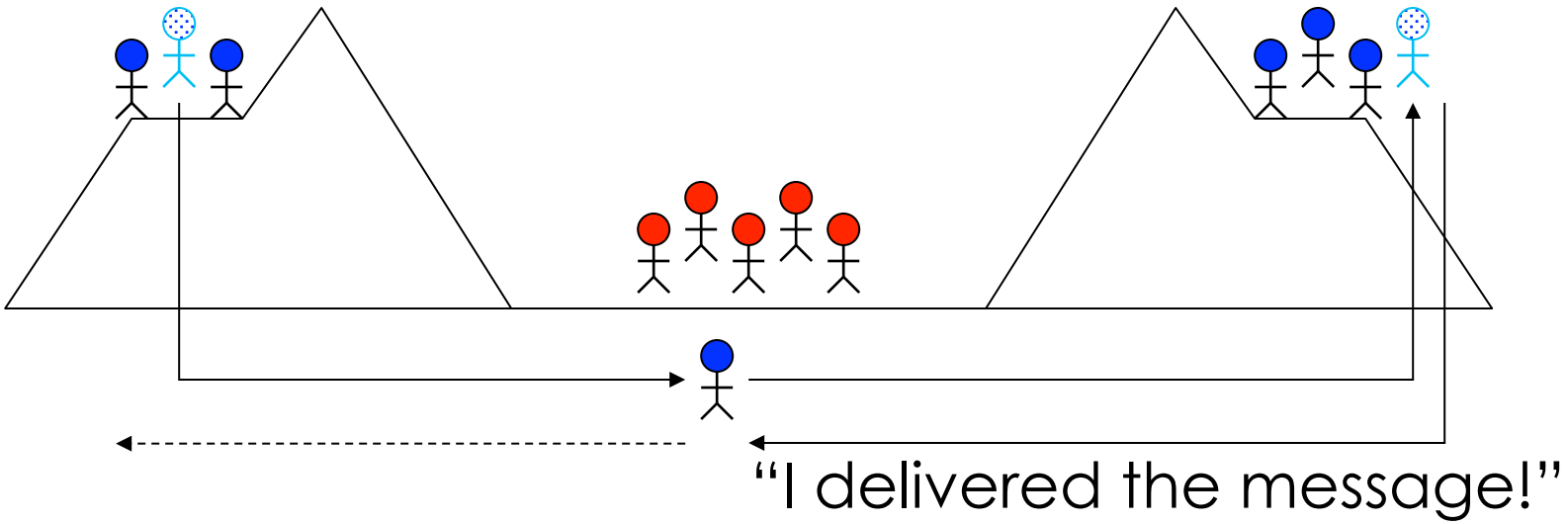
In the **Two Generals problem** two army divisions surround enemy, but must jointly attack to be successful.



The divisions can only communicate via a messenger, who may not make it.



To be sure the messenger made it, we await confirmation.

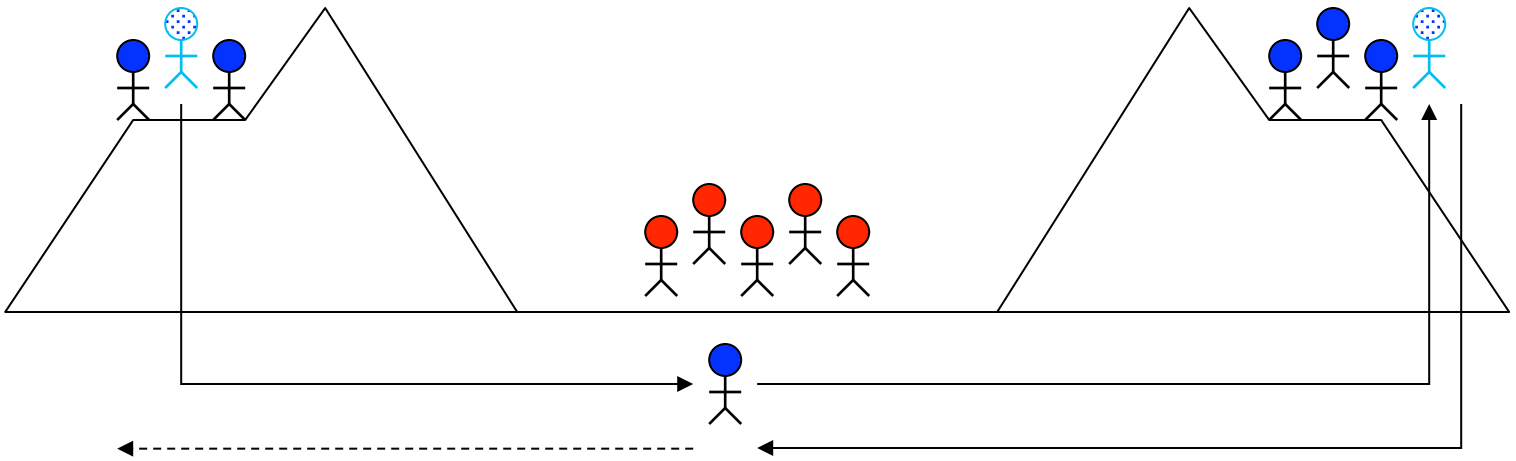


In the "two generals problem", can the two armies reliably coordinate their attack?

A.	Yes!
B.	No!

Be ready to explain why!

This problem shows us we can improve success rate, but can't make guarantees.



Give up? No way! We're pretty good at engineering solutions!

We have several tools for implementing reliable communication.

- **Our Concerns:**
  - Message corruption
  - Message duplication
  - Message loss
  - Message reordering
  - Performance
- **Our Toolbox:**
  - Checksums
  - Timeouts
  - ACKs & NACKs
  - Sequence numbering
  - Pipelining