Communications and Information Sharing
CS 118
Computer Network Fundamentals

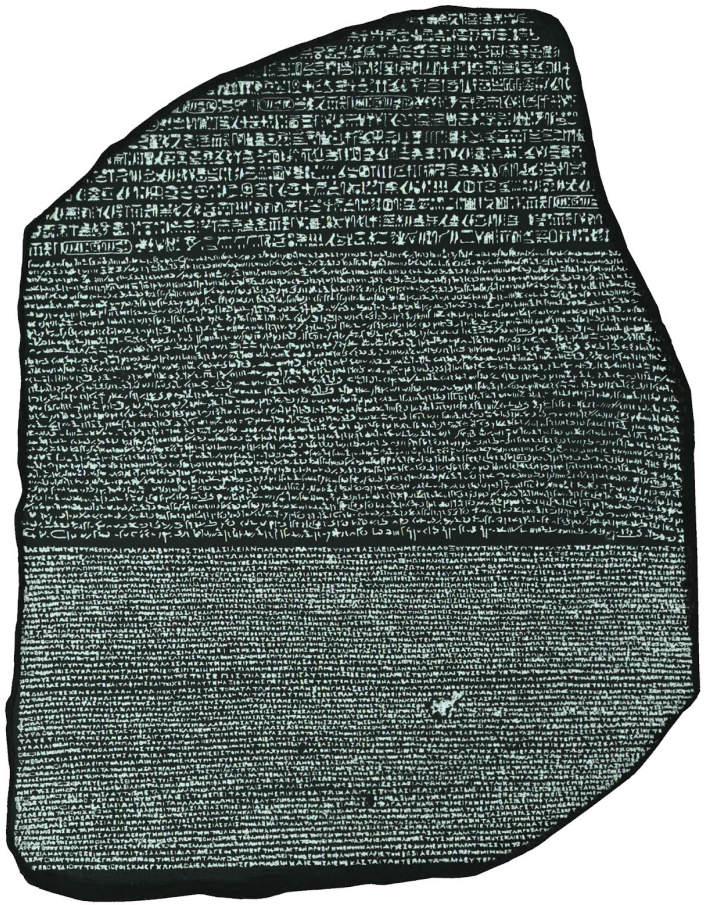
Peter Reiher

Shared Information

- Technical — What did you hear?
- Semantics — What did that mean?
- Effectiveness — What did I want you to understand?

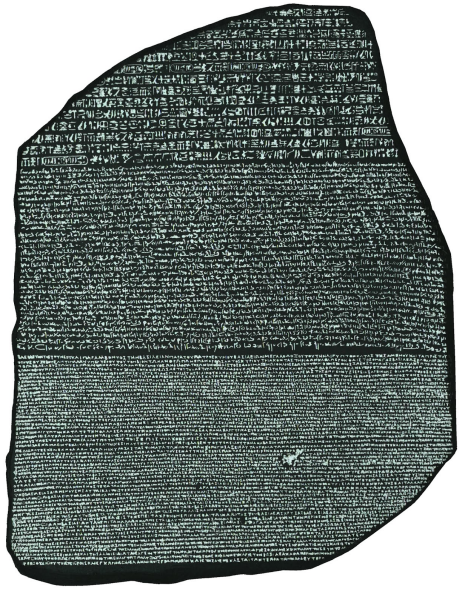
Syntax

- Symbols
  - Particular symbols
- Sequences
  - Order and arrangement



Semantics

- What does it mean?
  - Assigning values to symbols

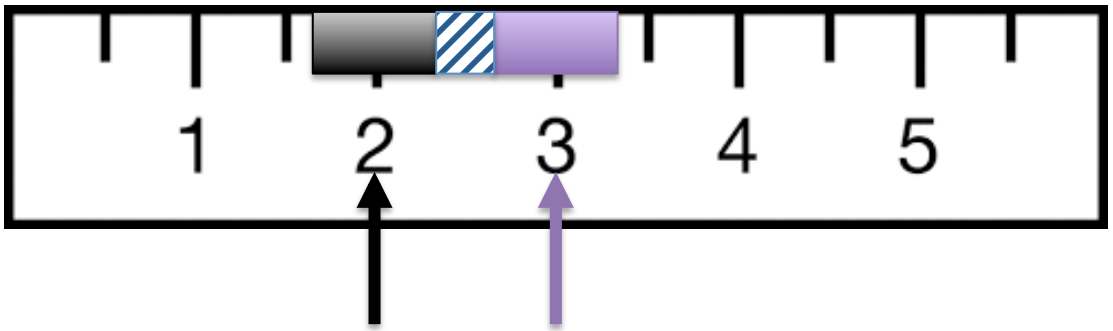


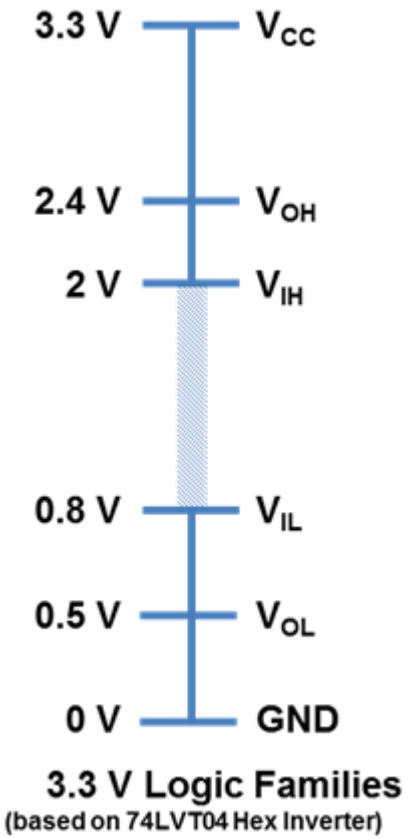
Analog vs. Digital

- Symbol type matters
- Analog symbols can be ambiguous
  - Is the curl at the end of the letter significant or not?
- Digital can be mapped to ONE meaning
- Computers don't do ambiguity
  - And they don't really do analog, anyway
- So we often map analog to digital

Discretization

- Pick specific analog values
  - Treat only those values as valid
  - Round unambiguous values ("restoration", "redigitization")





Effectiveness

- We want our messages to be understood by those who receive them
- Precision and accuracy are two important aspects of message effectiveness
- Would two receivers interpret a given message as the same state? (*precision*)
- Is it the state the transmitter intended? (*accuracy*)

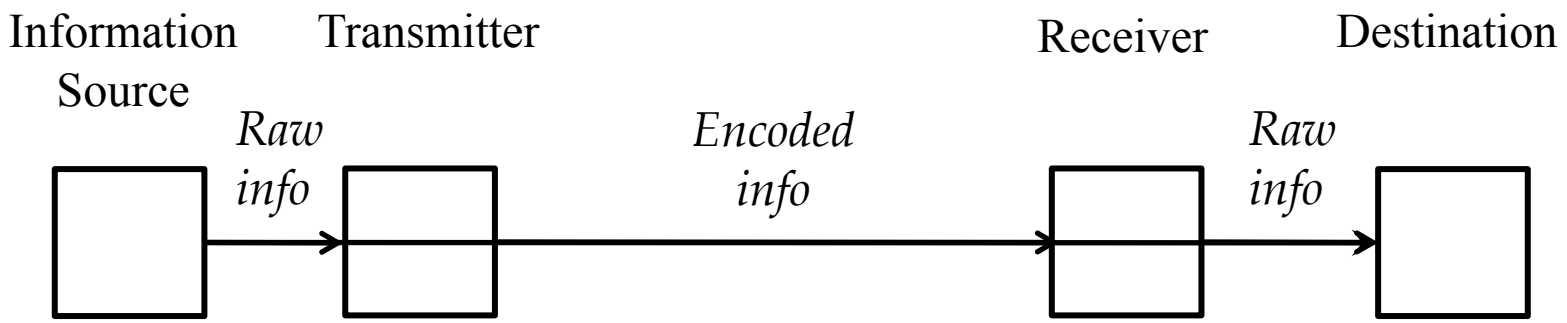
Which Shared Information Aspects Can We Handle?

- Syntax
  - Maybe, with enough rules, if we fix errors
- Semantics
  - Kitchen sink problem
  - Fruit flies *like* a banana, time flies *like* an arrow.
- Effectiveness
  - Often related to intent
  - Which is beyond technical means

Constraining the problem

- Syntax
  - Always check it
- Semantics
  - For control only
  - Not for message content
  - Meaning of loss, retransmission, flow control
- Effectiveness
  - Not really a communications problem
  - Assume effective or check results and redo

What Is Communications?



But we need to encode the information for transmission

- Both to match the physical characteristics of the transmission medium
- And for other purposes that will become clearer

Then it needs to be decoded after transmission

- Converting it back to something the destination can understand

Raw vs. encoded information

- Remember, we're at a general level here
  - Not just computer messages
  - Also speech, audio, and everything else
- Raw information is what the sender wants to communicate
  - And what the receiver wants to get
- Encoded information is what the transmission medium can deal with
  - Probably not the same as raw information

Raw information

- Not necessarily the same for sender and receiver
  - A German speaker sends a message to a Korean speaker
  - Probably starts out as German
  - And ends up as Korean
- Same may be true for computer communications
  - E.g., 32 bit word sender vs. 64 bit word receiver

Encoded information

- A characteristic of the communication medium
  - Not the nature of the raw information, usually
- If you're using Morse code, it's dots and dashes
  - Regardless of whether you send English or French
  - Though conversion to and from encoding could change
- If you're using electrical wires, it might be voltage levels

Messages and State

- The sending and receiving of a message specifies states
- What states?
- The states at the sender and receiver

Sender and Receiver States

- The sender sent a particular message
  - With particular syntax, semantics, and effectiveness
  - Those specify a state at the sender
- The receiver received a particular message
  - Again, with particular syntax, semantics, and effectiveness
  - That's the receiver's state
- But we can only send syntax
  - Not semantics or effectiveness

Entropy and Information

- A measure of disorder
  - Lack of predictability
- Entropy specifies the maximum amount of information in a message
- How many different messages could be sent?
- How many could be received?
- The more you could send or receive, the more possible states
  - And thus more entropy

For Example,

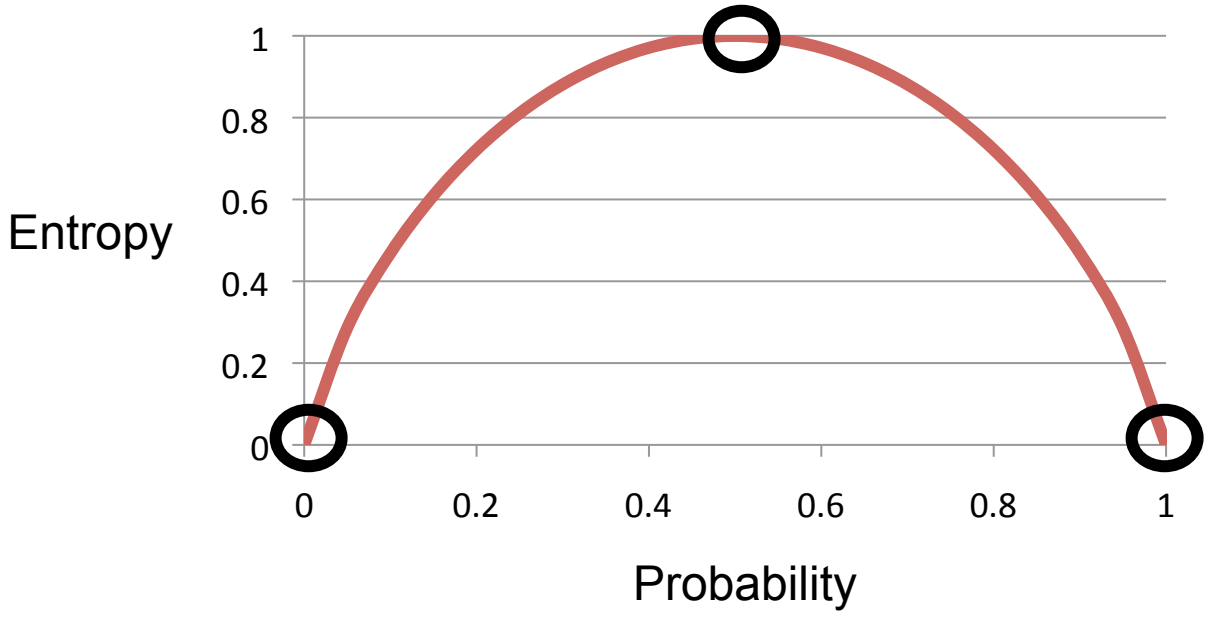
- The sender only sends YES or NO
  - And only sends one such message
- How much information is sent?
- One bit – a zero or a one
- What if the sender sends two such messages?
- Two bits – 00, 01, 10, or 11
- More choices, more states, more uncertainty
  ➔ more entropy

Communications and Entropy

- Shannon developed the basic equation describing communications in terms of entropy
- Consider the source as a finite state machine
- For each source state *i* there is an entropy H_i equal to $-p_i log\ p_i$
- Also, there is a probability $p_i(j)$ of producing symbol *j* from state *i*
- So the source's entropy is $-\Sigma p_i(p_i(j) log\ p_i(j))$
  - Stationary assumption gives us $H = -\Sigma p_i log\ p_i$

Max and min entropy

- Consider a two-state sender
  - Max entropy is when the choice is 50/50
  - Min entropy is when there is no choice



Returning to Our Example

- The sender only sends YES or NO
  - And only sends one such message
- When is the sender entropy at max?
  - When either message is equally likely
- When is the sender entropy at min?
  - When he always sends YES
  - Or always sends NO

Generalizing the Example

- What if we can send *N* different symbols?
  - Rather than just YES or NO
- When is entropy of the sender minimized?
  - When only one of the *N* symbols is ever sent
- When is entropy of the sender maximized?
  - When any one of the *N* symbols is equally likely

So What?

- We can now say something about how much information you can push through a channel
- Let the source have entropy H (bits per symbol)
- Let the channel have capacity C (bits per second)
- Then we can transmit $C/H - \varepsilon$ symbols per second
  - For arbitrarily small ε
- But never more than C/H

Predictability

- What if we're not sending random bits?
- Maybe it's English text in ASCII
- Maybe it's Morse code
- Then the $p_i(j)$'s are not uniform
  - Some symbols are more likely, given the symbols already sent
  - Entropy is lower than the max
  - Meaning we can squeeze more information through

What if choices aren't equal?

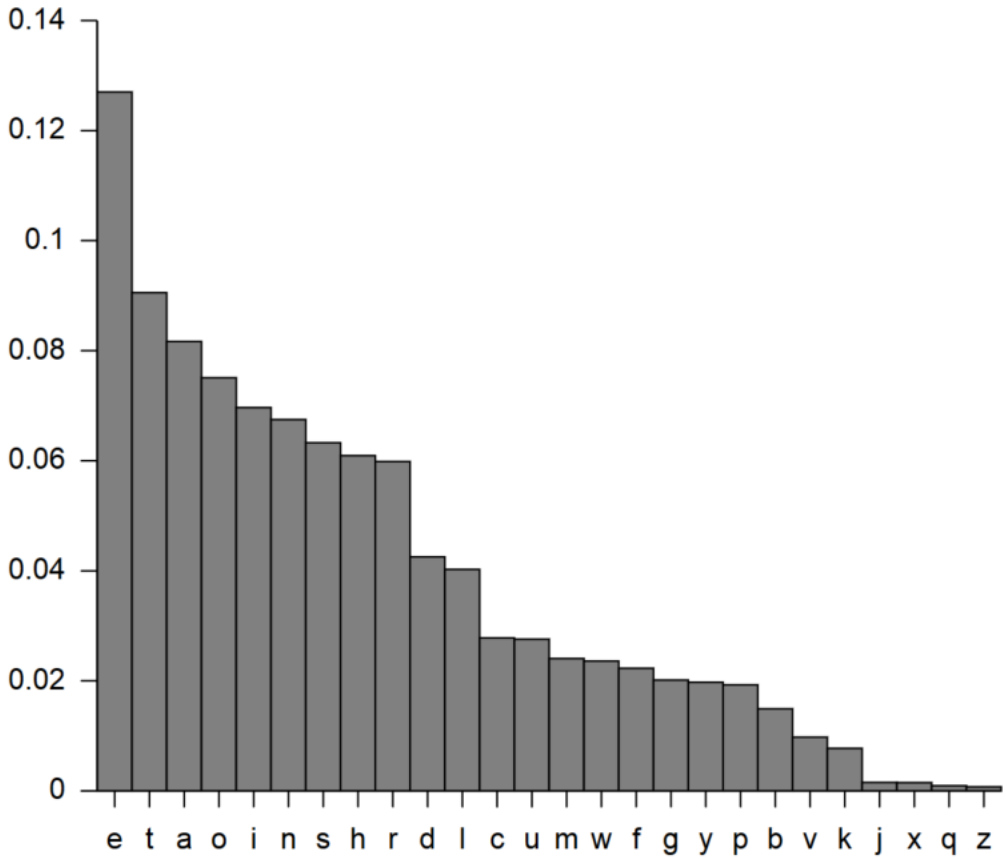
- YELLO_
  - What comes next?
- PIT_
  - What comes next?
- "Next letter" in English isn't 1 of 26
  - Roughly 50% redundant

A look at Morse code again…

- Time units:
  - Dot = t
  - Dash = $3t$
  - Inter-symbol gap within a letter = t

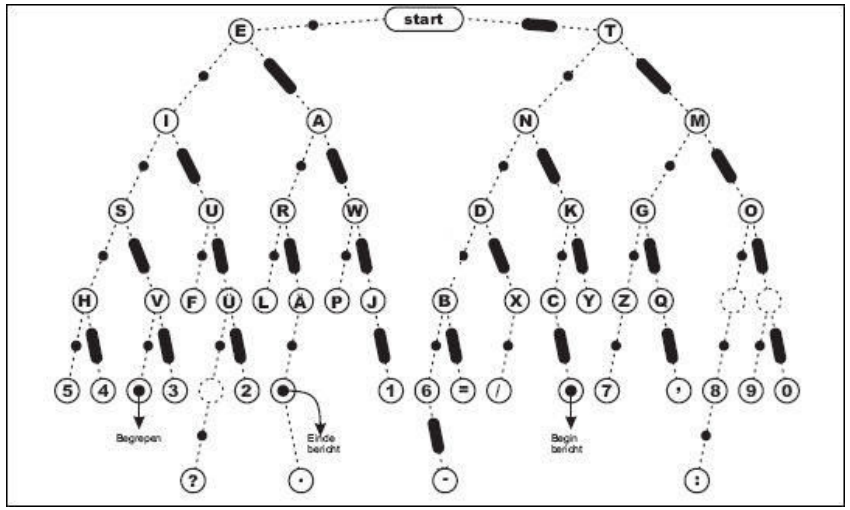
American English letter frequencies

- Basic order:
  - E
  - T
  - A
  - O
  - I
  - N
  - S



Morse code

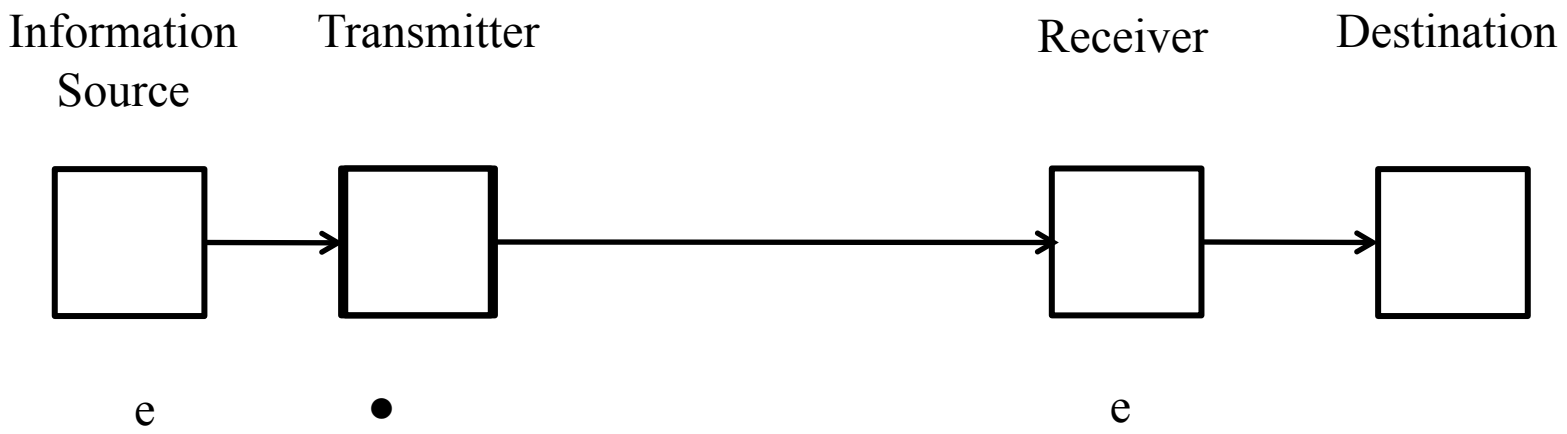
- Code representation:
  - E ● 1
  - T — 3
  - A ●— 5
  - O ——— 11
  - I ●● 3
  - N —● 5
  - S ●●● 5



How Do We Get More Through?

- Encoding it properly
- In essence, “short” signals for common things
- Long signals for uncommon things
- E.g., Morse code
  - Common letters are few dots and dashes
  - Uncommon letters require more dots and dashes
  - Each dot or dash takes up time on the line
  - They didn’t do it perfectly . . .

Who Does This Coding?

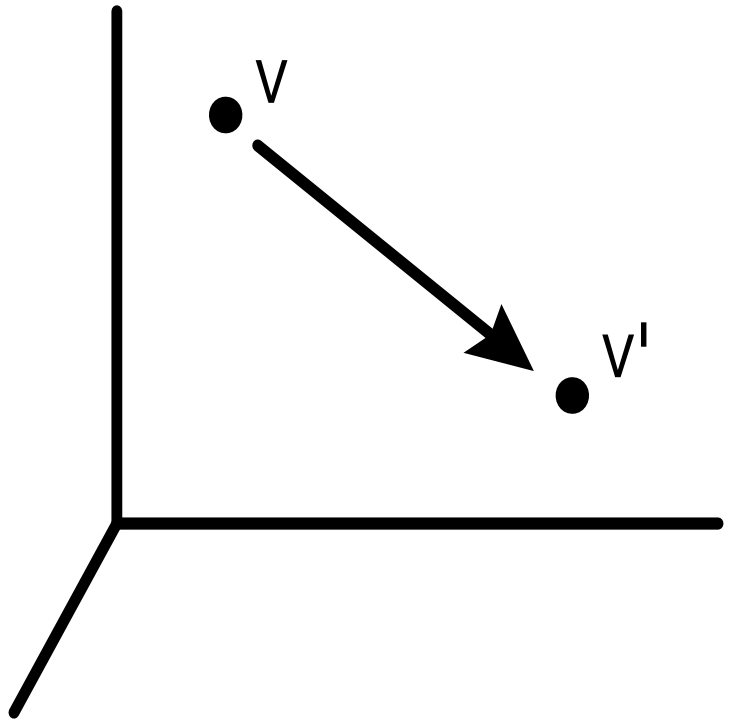


And the receiver decodes

The perils of sharing

- Shared state may be inaccurate
  - Channel errors
  - Time (i.e., 'staleness')
- Capacity is finite
  - Nobody can know everything

Simple state

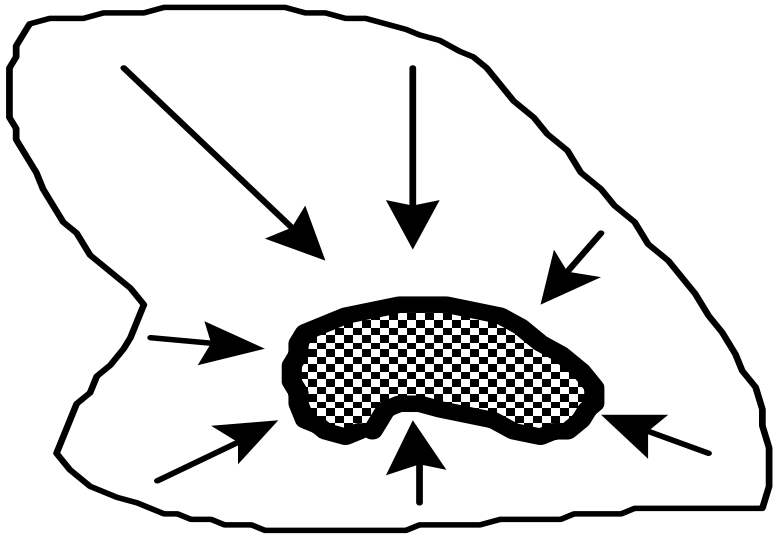


How does communication affect state?

- Knowledge doesn't stay still…

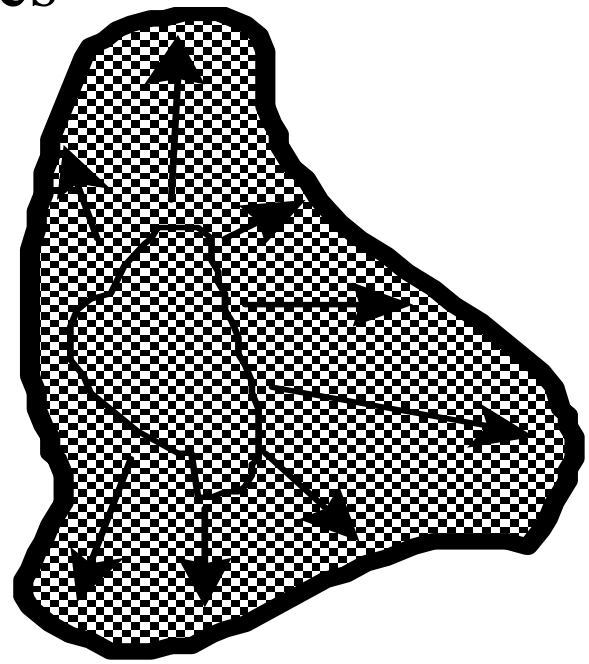
Effect of receiving

- Entropy decreases
  - Receiver knows more about the transmitter



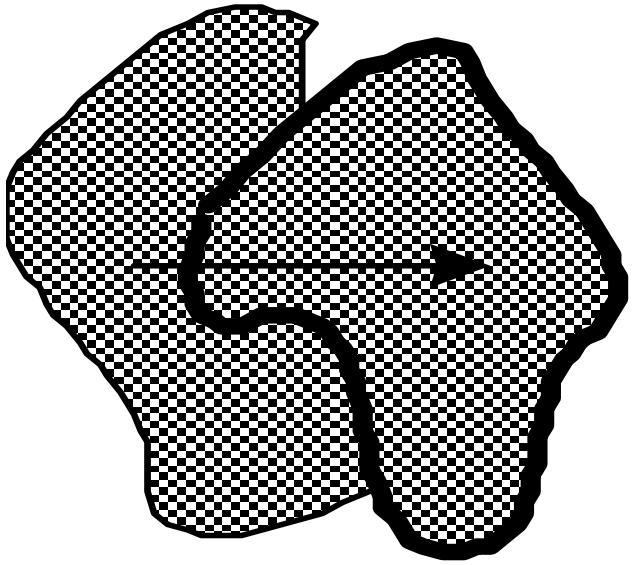
Effect of time

- Entropy never decreases over time
  - Usually increases



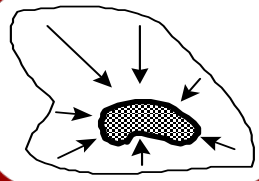
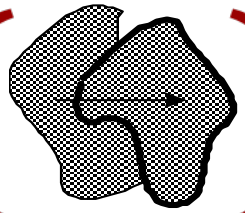
Effect of sending (1)

- Sending information about your state
  - Makes your view of receiver state fuzzier



Effect of sending (2)

- Entire system entropy never decreases
  - Receiver's model of transmitter entropy decreases in entropy, so sender's model of receiver MUST increase in entropy

$$S(\text{  }) + S(\text{  }) \geq 0$$

Putting it all together – CTP

- Character transfer protocol
  - (we're creating this as an example)
  - Sending a message one letter at a time
- Assume a perfect channel
  - No errors in transmission, ever

CTP events

- Starting condition
  - Both sides share rules (protocol)
  - And share endpoint info (link)

CTP rules

- Both endpoints start in the state:
  - NOT-CONNECTED
- Use phone-call protocol to get to:
  - CONNECTED
- Connected transmitter
  - sends characters one a time
- Connected receiver
  - receives characters one at a time until CLOSED
- At EOF
  - transmitter closes connection

From not-connected to connected

- Simple phone-call protocol
  - Transmitter initiates, waits for response
  - Receiver responds when asked
  - Once confirmed, both sides enter CONNECTED state

Remember – perfect channel,
so no need for timeouts or exceptions

Simple transfer

- Character at a time
  - Transmitter sends characters in order
  - Receiver collects characters and places them in order
- What state is shared?
  - CONNECTED state
  - The character

From connected to closed

- Final change in shared state
- Lets receiver know
  - Transfer can be stopped
  - Message is complete

CTP evolution

- Successive increased shared state:
  - Agree to change from not-connected to connected
    - Using the phone-call protocol
  - Agree on each character sent
    - In a perfect channel, nothing is lost or reordered, so transmitter knows what receiver gets (i.e., agreement always happens)
  - Agree to end transfer

Limits of CTP

- Assumes a perfect channel
  - Ignores loss, reordering, flow control, etc.
- Inefficient
  - The agreed state is augmented one character at a time

Once closed, what do we know?

- The message!
  - WHY? – the succession of shared state

Back to predictability

- We know more than we think
  - Can send groups of characters
  - Can confirm using “checksums”

Putting it all together – FTP

- A more efficient version of CTP

Starting point

- Share protocol rules
- Already know each other's endpoint
- Know we're using those rules
  - TCP port 22

From not-connected to connected

- Uses TCP's version of the phone-call protocol:
  - Transmitter sends SYN
  - Receiver sends SYN-ACK
  - Once confirmed, both sides enter ESTABLISHED TCP state
- What's the difference?
  - Over a perfect channel, NOTHING

What about an imperfect channel?

- Some characters might be dropped
  - "H e l l o" becomes "H e l l"
- Some characters might be changed
  - "F a r" becomes "F u r"
- Some characters might be added
  - "H e a t" becomes "H e a r t"
- Or maybe it came through unchanged
- How can the receiver know what happened?

Better transfer

- Block at a time
  - Transmitter sends characters in order inside blocks, labeled with a checksum
  - Receiver collects blocks, verifies checksums, and places them in order
- What state is shared?
  - Connection state
  - The CHECKSUMS!

FTP evolution

- Successive increased shared state:
  - Agree to change from IDLE to ESTABLISHED
    - Using the TCP protocol
  - Agree on each checksum sent
  - Agree to end transfer

From connected to closed

- Final change in shared state
- Lets receiver know
  - Transfer can be stopped
  - Message is complete – WHY?

FTP's leap of faith

- Correct file transfer IFF:
  - Sequence of correct blocks
  - Each block is correct IFF checksum is correct

Examples of FTP-like exchanges

- File transfer
- Web request/response
- Netflix

What other states can we add?

- Pacing
  - How fast does the transfer go?
  - At constant or changing tempo?
- Number of outstanding messages
  - Messages sent but not yet received
  - Or perhaps received, but not yet acknowledged
- Amount of reordering

What should you assume?

- As little as possible!

Postel Principle

- Be conservative in what you do…
  - Transmit only what you think will help
  - Transmit only what you expect will be understood
- Be liberal in what you tolerate.
  - If a message could mean multiple things, allow as many as possible
  - Do not assume malice where incompetence will suffice (errors happen)

Summary

- Communication is less than most think
  - Just syntax – not semantics or intent
- Information is based on states
  - Which is based on entropy (disorder)
- We can model how state evolves
  - Each side models the other
  - Successive steps in models are how we go from sharing state to transferring files