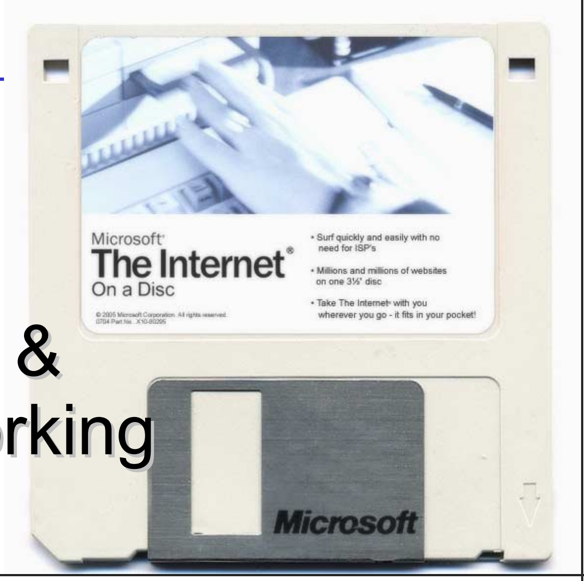
Types & Networking


Microsoft
The Internet®
On a Disc
• Surf quickly and easily with no need for ISP's
• Millions and millions of websites on one 3½" disc
• Take The Internet with you wherever you go - it fits in your pocket!
Microsoft

One-Slide Summary

- A **type** is a (possibly infinite) set of values. Each type supports a set of **valid operations**. Types can be latent or manifest, static or dynamic, strong or weak.
- We can change the Charme interpreter to support manifest (program visible) types.
- A **network** is a group of three or more communicating entities.
- **Bandwidth** is the throughput of a communication resource, measured in bits per second. **Latency** is the time delay between the moment when communication is initiated and the moment the first bit arrives, measured in seconds.
- In **circuit switching**, a path through a network is reserved (high quality-of-service, used in telephones). In **packet switching**, each packet is routed individually (internet, postal service).

Outline

- Administration
- StaticCharme Typechecking
- Networking History
- Latency, Bandwidth, Switching
- The Internet
- Dynamic Web Sites

Structured Lab in Small Hall Today! (bring a laptop if you have one)

Administrivia

- **Start PS8 and PS9 Now**
  - PS9 Team Requests due Friday April 10th
- **1st CS Department Fireside Chat**
  - TODAY (Wednesday Apr 8) 5:30pm MEC 205
  - Worth 2 points of Extra Credit on Exam 2
- **Kinga's Web Fault Research Survey**
  - http://www.cs.virginia.edu/~kld5r/webfault/
  - Worth 2 points of Extra Credit on Exam 2
  - Plus possibly $$$ ...
- **2009 Computing and Communication Scholarship for Undergraduate Women**
  - http://www.cs.virginia.edu/ccscholarship
  - $1000 merit scholarship, due June 30th

Student Comments

- I am displeased with the course in general. I was expecting a course that showed how computing concepts relate to the liberal arts. While that is true to some extend, this class feels more like a straight programming class. DrScheme is not intuitive, and makes this course much harder than 101 and 201 without really giving more information.
  - *Managing expectations is the key to happiness!*

More Student Comments

- I am displeased that the answer to question eight in this problem set require a lot of thinking and writing code for only one point of the assignment.
  - *Irony! The goal was to make it reasonable to not do all of the problem set.*
- I am displeased that I have to make a dynamic website, which will take a lot of time and likely be our hardest assignment.
  - *Yes, the final project will be hard.*

Even More Student Comments

- I am displeased that some people are opting not to do the problem sets. I am not a computer science major and this is the first cs class I have ever taken, but I still think it is important for everyone to branch out and learn new things. Although some people may say that they will never use this stuff again, you never know.
  - *Currently **zero** students have opted out of the problem sets.*

Displeased

21	Course and problem sets are hard/long/frustrating
6	Reading quizzes
5	Two exams + final = too much work at end
5	Switch languages (learning on our own)
4	Book remains dry, confusing, and without answers
3	Don't know what to do for PS9
2	It still takes too long to get help in office hours
2	Cannot drop lowest PS grade
2	IDLE sucks
8	Other

Changes?

- Vote:
  - "Ignore all reading quiz grades. I will show my knowledge of the material elsewhere."
- We will add answers to the textbook next year.
  - By hiring students ...
- Come talk to me and I will give you six free final project ideas.
- You *can* drop the lowest PS grade.

CPrimitiveType

```
class CPrimitiveType(CType):
  def __init__(self, s):
    self._name = s
  def __str__(self):
    return self._name
  def isPrimitiveType(self):
    return True
  def matches(self, other):
    return other.isPrimitiveType() \
           and self._name == other._name
```

class X(Y):
means **X** is a subclass of **Y**

CProcedureType

```
class CProcedureType(CType):
  def __init__(self, args, rettype):
    self._args = args
    self._rettype = rettype
  def __str__(self):
    return "(" + str(self._args) + " -> " \
           + str(self._rettype) + ")"
  def isProcedureType(self): return True
  def getReturnType(self): return self._rettype
  def getParameters(self): return self._args
  def matches(self, other):
    return other.isProcedureType() \
```

???

CProcedureType

```
class CProcedureType(CType):
  def __init__(self, args, rettype):
    self._args = args
    self._rettype = rettype
  def __str__(self):
    return "(" + str(self._args) + " -> " \
           + str(self._rettype) + ")"
  def isProcedureType(self): return True
  def getReturnType(self): return self._rettype
  def getParameters(self): return self._args
  def matches(self, other):
    return other.isProcedureType() \
      and self.getParameters().matches(other.getParameters()) \
      and self.getReturnType().matches(other.getReturnType())
```

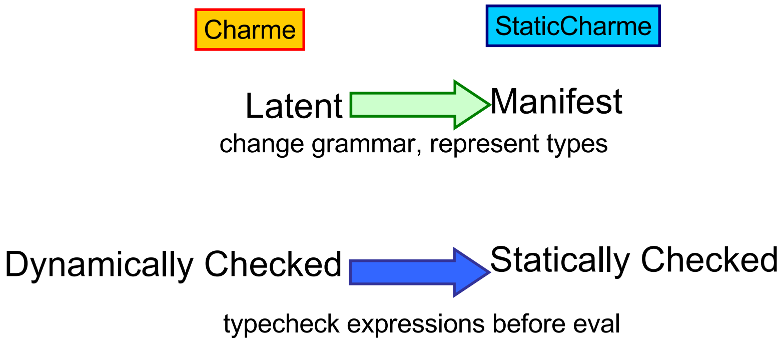
CProductType

```
class CProductType(CType):
  def __init__(self, types): self._types = types
  def __str__(self): ...
  def isProductType(self): return True
  def matches(self, other):
    ???
```

CProductType

```
class CProductType(CType):
  def __init__(self, types): self._types = types
  def __str__(self): ...
  def isProductType(self): return True
  def matches(self, other):
    if other.isProductType():
      st = self._types
      ot = other._types
      if len(st) == len(ot):
        for i in range(0, len(st)):
          if not st[i].matches(ot[i]): return False
        # reached end of loop ==> all matched
        return True
    return False
```

Types of Types



Charme

StaticCharme

Latent → Manifest

change grammar, represent types

Dynamically Checked → Statically Checked

typecheck expressions before eval

Liberal Arts Trivia: Dance

- This closed position, ¾ time standard ballroom dance featuring gliding steps and rotations. It became fashionable in Vienna in the 1780s and shocked many when it was first introduced: unlike the popular folk dances of the time, it was a couples dance that involved the leader clasping the follower about the waist. This gave it a dubious moral status in the eyes of the gentry.
- Bonus: My uncle Walter goes ...

Liberal Arts Trivia: Linguistics

- This Chinese language dialect (Yuet Yu or Yue Yu) is popular in Hong Kong, Macau and southern mainland China. It retains more tones and consonant endings from older varieties of Chinese that have been lost to other modern Chinese dialects. Its rarely-used written form contains many characters not used in standard written Chinese. See 2nd here:

Adding Type Checking

```
def evalLoop():
  initializeGlobalEnvironment()
  while True:
    ...
      for expr in exprs:
        typ = typecheck(expr, globalEnvironment)
        if typ and typ.isError():
          print "Type error:" + typ.getMessage()
        else:
          res = meval(expr, globalEnvironment)
          if res != None:
            print str(res)
```

Static Type Checking

```
def typecheck(expr, env):
  if isPrimitive(expr):
    return typePrimitive(expr)
  elif isConditional(expr):
    return typeConditional(expr, env)
  elif isLambda(expr):
    return typeLambda(expr, env)
  elif isDefinition(expr):
    typeDefinition(expr, env)
  elif isName(expr):
    return typeName(expr, env)
  elif isApplication(expr):
    return typeApplication(expr, env)
  else: evalError ("Unknown expression: " + str(expr))
```

```
class Environment:
  # Store a [type, value] pair for each variable.
  ...
  def addVariable(self, name, typ, value):
    self._frame[name] = (typ, value)
  def lookupPlace(self, name):
    if self._frame.has_key(name): return self._frame[name]
    elif (self._parent): return self._parent.lookupPlace(name)
    else: return None
  def lookupVariableType(self, name):
    place = self.lookupPlace(name)
    if place: return place[0]
    else: return CErrorType("Name not found")
  def lookupVariable(self, name):
    return self.lookupPlace(name)[1]
   ...
```

Typechecking Names

```
def typeName(expr, env):
   return env.lookupVariableType(expr)
```

```
def evalDefinition(expr, env):
   name = expr[1]
   value = meval(expr[4], env)
   typ = CType.fromParsed(expr[3])
   env.addVariable(name, typ, value)
```

Static Type Checking

```
def typecheck(expr, env):
  if isPrimitive(expr):
    return typePrimitive(expr)
  elif isConditional(expr):
    return typeConditional(expr, env)
  elif isLambda(expr):
    return typeLambda(expr, env)
  elif isDefinition(expr):
    typeDefinition(expr, env)
  elif isName(expr):
    return typeName(expr, env)
  elif isApplication(expr):
    return typeApplication(expr, env)
  else: evalError ("Unknown expression: " + str(expr))
```

```
def typeDefinition(expr, env):
   assert isDefinition(expr)
   if len(expr) != 5:
      evalError ("Bad definition: %s" % str(expr))
   name = expr[1]
   if isinstance(name, str):
      if expr[2] != ':':
         evalError ("Definition missing type: %s" % str(expr))
      typ = CType.fromParsed(expr[3])
      etyp = typecheck(expr[4], env)
      if not typ.matches(etyp):
         evalError("Mistyped definition: ..." % (name, typ, etyp))
   elif isinstance(name, list):
      evalError ("Procedure definition syntax not implemented")
   else: evalError ("Bad definition: %s" % str(expr))
```

Example: (define x : Number “hello”)
Example: (define y : Number (+ 2 3))

Static Type Checking

```
def typecheck(expr, env):
  if isPrimitive(expr):
    return typePrimitive(expr)
  elif isConditional(expr):
    return typeConditional(expr, env)
  elif isLambda(expr):
    return typeLambda(expr, env)
  elif isDefinition(expr):
    typeDefinition(expr, env)
  elif isName(expr):
    return typeName(expr, env)
  elif isApplication(expr):
    return typeApplication(expr, env)
  else: evalError ("Unknown expression: " + str(expr))
```

```
(define square : (Number -> Number)
  (lambda (x : Number) (* x x)))
```

```
class Procedure:
   def __init__(self, params, typ, body, env):
      self._params = params
      self._body = body
      self._typ = typ
      self._env = env
   def getParams(self):
      return self._params
   def getParamTypes(self):
      return self._typ
   def getBody(self): return self._body
   def getEnvironment(self): return self._env
   def __str__(self):
      return "<Procedure %s / %s>" \
        % (str(self._params), str(self._body))
```

Add type to Procedure

```
def evalLambda(expr, env):
   assert isLambda(expr)
   if len(expr) != 3:
      evalError ("Bad lambda expression: %s" % (str(expr)))
   params = expr[1]
   paramtypes = []
   paramnames = []
   assert len(params) % 3 == 0
   for i in range(0, len(params) / 3):
      name = params[i*3]
      assert params[(i*3)+1] == ':'
      paramnames.append(name)
      typ = CType.fromParsed(params[(i*3)+2])
      paramtypes.append(typ)
   return Procedure(paramnames, paramtypes, expr[2], env)
```

```
(lambda (x : Number
         y : Number)
   (* x y))
```

```
def typeLambda(expr, env):
   assert isLambda(expr)
   if len(expr) != 3: evalError ("Bad lambda expression: %s" % str(expr))
   # this is a bit tricky - we need to "partially" apply it
   # to find the type of the body
   newenv = Environment(env)
   params = expr[1]
   paramnames = []
   paramtypes = []
   assert len(params) % 3 == 0
   for i in range(0, len(params) / 3):
      name = params[i*3]
      assert params[(i*3)+1] == ':'
      typ = CType.fromParsed(params[(i*3)+2])
      paramnames.append(name)
      paramtypes.append(typ)
      newenv.addVariable(name, typ, None)
   resulttype = typecheck(expr[2], newenv)
   return CProcedureType(CProductType(paramtypes), resulttype)
```

Study me for Exam 2!

Static Type Checking

```
def typecheck(expr, env):
   if isPrimitive(expr):
      return typePrimitive(expr)
   elif isConditional(expr):
      return typeConditional(expr, env)
   elif isLambda(expr):
      return typeLambda(expr, env)
   elif isDefinition(expr):
      typeDefinition(expr, env)
   elif isName(expr):
      return typeName(expr, env)
   elif isApplication(expr):
      return typeApplication(expr, env)
   else: evalError ("Unknown expression: " + str(expr))
```

Typechecking an Application

```
def typeApplication(expr, env):
   proctype = typecheck(expr[0], env)
   if not proctype.isProcedureType():
      evalError("Application of non-procedure: " + str(expr[0]))
   optypes = map (lambda op: typecheck(op, env), expr[1:])
   optype = CProductType(optypes)
   if not optype.matches(proctype.getParameters()):
      evalError("Parameter type mismatch: ..." \
            % (proctype.getParameters(), optype))
   return proctype.getReturnType()
```

square : Number -> Number
Example: (+ 1 (square 5))
Example: (+ 2 (square “hello”))

Static Type Checking

```
def typecheck(expr, env):
   if isPrimitive(expr):
      return typePrimitive(expr)
   elif isConditional(expr):
      return typeConditional(expr, env)
   elif isLambda(expr):
      return typeLambda(expr, env)
   elif isDefinition(expr):
      typeDefinition(expr, env)
   elif isName(expr):
      return typeName(expr, env)
   elif isApplication(expr):
      return typeApplication(expr, env)
   else: evalError ("Unknown expression: " + str(expr))
```

Typechecking Primitives

```
def typePrimitive(expr):
   if isNumber(expr):
      return CPrimitiveType('Number')
   elif isinstance(expr, bool):
      return CPrimitiveType('Boolean')
   elif callable(expr):
      return findPrimitiveProcedureType(expr)
   else:
      assert False
```

This is a kludgey procedure that looks through the global environment to find the matching procedure, and returns its type

Static Type Checking

```
def typecheck(expr, env):
  if isPrimitive(expr):
    return typePrimitive(expr)
  elif isConditional(expr):
    return typeConditional(expr, env)
  elif isLambda(expr):
    return typeLambda(expr, env)
  elif isDefinition(expr):
    typeDefinition(expr, env)
  elif isName(expr):
    return typeName(expr, env)
  elif isApplication(expr):
    return typeApplication(expr, env)
  else: evalError ("Unknown expression: " + str(expr))
```

Left as possible Exam 2 question!

StaticCharme

```
StaticCharme> (+ 1 #t)
Error: Parameter type mismatch:
expected (Number Number), given (Number Boolean)
StaticCharme> (define square:((Number) -> Number)
  (lambda (x:Number) (* x x)))
StaticCharme> (square #t)
Type error: Parameter type mismatch:
expected (Number), given (Boolean)
StaticCharme> (define badret:((Number) -> Number)
                (lambda (x: Number) (> x 3)))
Error: Mistyped definition:
badret declared type ((Number) -> Number),
actual type ((Number) -> Boolean)
```

Who Invented the Internet?

Who Invented Networking?

What is a Network?

A **network** is a group of three or more connected communicating entities.

Beacon Chain Networking

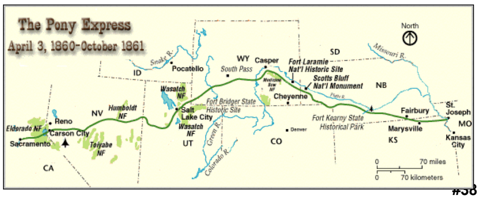
Thus, from some far-away beleaguered island, where all day long the men have fought a desperate battle from their city walls, the smoke goes up to heaven; but no sooner has the sun gone down than the light from the line of beacons blazes up and shoots into the sky to warn the neighboring islanders and bring them to the rescue in their ships.

Iliad, Homer, 700 BC

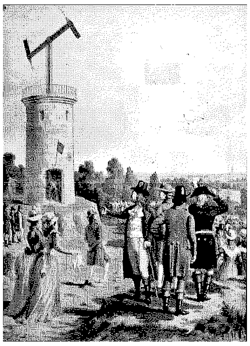
Chain of beacon's signaled Agammemnon's return (~1200BC), spread on Greek peaks over 600km.

Pony Express

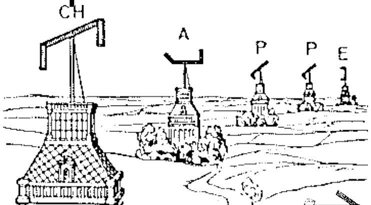
- April 1860 – October 1861
- Missouri to California
  - 10 days
  - 10-15 miles per horse, ~100 miles per rider
- 400 horses total



Chappe's Semaphore Network



First Line (Paris to Lille), 1794



Mobile Semaphore Telegraph
Used in the Crimean War 1853-1856

Government and Networking

Chappe wanted a commercial network

The use of novel methods that modify established habits, often hurts the interests of those who profit the most from the older methods. Few people, with the exception of the inventors, are truly interested in helping projects succeed while their ultimate impact is still uncertain. . . . Those in power will normally make no effort to support a new invention, unless it can help them to augment their power; and even when they do support it, their efforts are usually insufficient to allow the new ideas to be fully exploited. (Claude Chappe, 1824)

Anyone performing unauthorized transmissions of signals from one place to another, with the aid of telegraphic machines or by any other means, will be punished with an imprisonment of one month to one year, and a fine of 1,000 to 10,000 Francs.

French Law passed in 1837 made private networking illegal

Liberal Arts Trivia: Mathematics

- The *this* of a function at a chosen input value describes the best linear approximation of the function near that input point. If *this* can be applied to a function infinitely many times, the function is called smooth. The *this* is also given by the limit, as the difference in input approaches zero, of the ratio of the difference between the function values of two nearby inputs to the difference between those two nearby inputs.

Liberal Arts Trivia: Religious Studies

- Among the truths said to have been realized by Siddhartha Gautama Buddha during his experience of enlightenment are these:
  1) The Nature of Suffering (hint: almost everything)
  2) Suffering's Origin (hint: desire)
  3) Suffering's Cessation (hint: freedom from craving)
  4) The Way Leading to the Cessation of Suffering (hint: Noble Eightfold Path)

  What are these things collectively know as?

Measuring Networks

- **Latency**

  Time from sending a bit until it arrives

  seconds (or *seconds per geographic distance*)

- **Bandwidth**

  How much information can you transmit per time unit

  bits per second

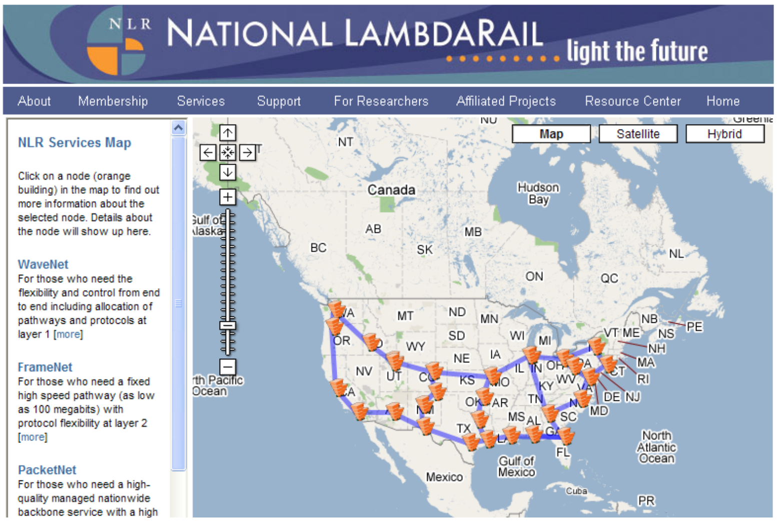
Latency and Bandwidth

- Napoleon's Network: Paris to Toulon, 475 mi
- Latency: 13 minutes (1.6s per mile)
  - What is the delay at each signaling station, how many stations to reach destination
  - At this rate, it would take ~1 hour to get a bit from California
- Bandwidth: 2 symbols per minute (98 possible symbols, so that is ~13 bits per minute)
  - How fast can signalers make symbols
  - At this rate, it would take you about 9 days to get *ps8.zip*

Improving Latency

- Fewer transfer points
  - Longer distances between transfer points
  - Semaphores: how far can you see clearly
    - Curvature of Earth is hard to overcome
  - Use wires (electrical telegraphs, 1837)
- Faster transfers
  - Replace humans with machines
- Faster travel between transfers
  - Hard to beat speed of light (semaphore network)
  - Electrons in copper: about 1/3rd speed of light

How many transfer points between here and California?



NLR Services Map

Click on a node (orange building) in the map to find out more information about the selected node. Details about the node will show up here.

WaveNet

For those who need the flexibility and control from end to end including allocation of pathways and protocols at layer 1 [more]

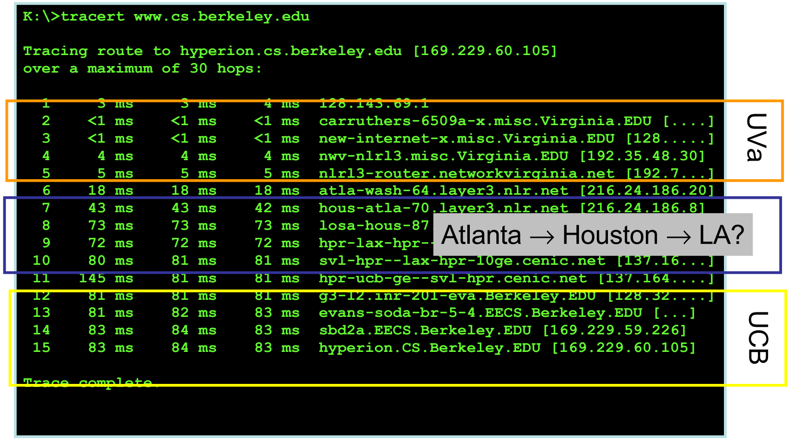
FrameNet

For those who need a fixed high speed pathway (as low as 100 megabits) with protocol flexibility at layer 2 [more]

PacketNet

For those who need a high-quality managed nationwide backbone service with a high

tracert



```
K:\>tracert www.cs.berkeley.edu

Tracing route to hyperion.cs.berkeley.edu [169.229.60.105]
over a maximum of 30 hops:

  1     3 ms     3 ms     4 ms  128.143.69.1
  2    <1 ms    <1 ms    <1 ms  carruthers-6509a-x.misc.Virginia.EDU [....]
  3    <1 ms    <1 ms    <1 ms  new-internet-x.misc.Virginia.EDU [128.....]
  4     4 ms     4 ms     4 ms  nwv-nlrl3.misc.Virginia.EDU [192.35.48.30]
  5     5 ms     5 ms     5 ms  nlrl3-router.networkvirginia.net [192.7...]
  6    18 ms    18 ms    18 ms  atla-wash-64.layer3.nlr.net [216.24.186.20]
  7    43 ms    43 ms    42 ms  hous-atla-70.layer3.nlr.net [216.24.186.8]
  8    73 ms    73 ms    73 ms  losa-hous-87
  9    72 ms    72 ms    72 ms  hpr-lax-hpr--
 10    80 ms    81 ms    81 ms  svl-hpr--lax-hpr-10ge.cenic.net [137.16...]
 11   145 ms    81 ms    81 ms  hpr-ucb-ge--svl-hpr.cenic.net [137.164....]
 12    81 ms    81 ms    81 ms  g3-12.inr-201-eva.Berkeley.EDU [128.32....]
 13    81 ms    82 ms    83 ms  evans-soda-br-5-4.EECS.Berkeley.EDU [...]
 14    83 ms    84 ms    83 ms  sbd2a.EECS.Berkeley.EDU [169.229.59.226]
 15    83 ms    84 ms    83 ms  hyperion.CS.Berkeley.EDU [169.229.60.105]

Trace complete.
```

```
>>> cvilleberkeley = 3813 # kilometers
>>> seconds = 84.0/1000
>>> speed = cvilleberkeley / seconds
>>> speed
45392.857142857138
>>> light = 299792.458 # km/s
>>> speed / light
0.15141427321316114
```

Packets are traveling average at 15% of the speed of light (includes transfer time through 15 routers)

Bandwidth

How much data can you transfer in a given amount of time?

Improving Bandwidth

- Faster transmission
  - Train signalers to move semaphore flags faster
  - Use something less physically demanding to transmit
- Bigger pipes
  - Have multiple signalers transmit every other letter at the same time
- Better encoding
  - Figure out how to code more than 98 symbols with semaphore signal
  - Morse code (1840s)

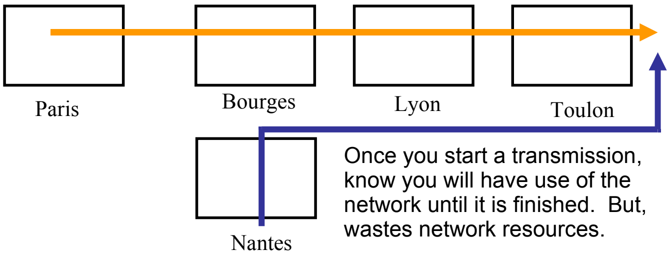
Morse Code

Represent letters with series of short and long electrical pulses

A	B	C	D
• –	– • • •	– • – •	– • •
E	F	G	H
•	• • – •	– – •	• • • •
I	J	K	L
• •	• – – –	– • –	• – • •
M	N	O	P
– –	– •	– – –	• – – •
Q	R	S	T
– – • –	• – •	• • •	–
U	V	W	X
• • –	• • • –	• – –	– • • –
Y	Z		
– • – –	– – • •		

Circuit Switching

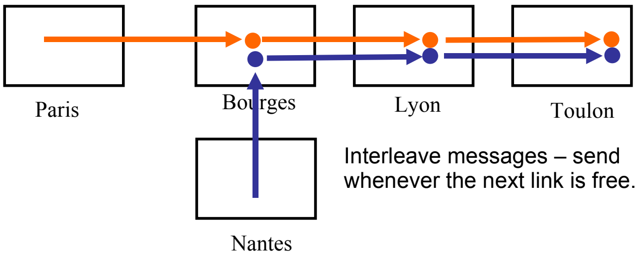
- Reserve a whole path through the network for the whole message transmission



Once you start a transmission, know you will have use of the network until it is finished. But, wastes network resources.

Packet Switching

- Use one link at a time



Interleave messages – send whenever the next link is free.

Circuit and Packet Switching

- (Land) Telephone Network (back in the old days)
  - Circuit: when you dial a number, you have a reservation on a path through the network until you hang up
- The Internet
  - Packet: messages are broken into small packets, that find their way through the network link by link

internetwork

An **internetwork** is a collection of multiple networks connected together, so messages can be transmitted between nodes on different networks.

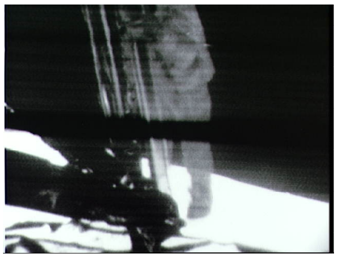
The First internet

- 1800: Sweden and Denmark worried about Britain invading
- Edelcrantz proposes link across strait separating Sweden and Denmark to connect their (signaling) telegraph networks
- 1801: British attack Copenhagen, network transmit message to Sweden, but they don't help.
- Denmark signs treaty with Britain, and stops communications with Sweden

First Use of Internet

- October 1969: First packets on the ARPANet from UCLA to Stanford. Starts to send "LOGIN", but it crashes on the G.
- 20 July 1969:

Live video (b/w) and audio transmitted from moon to Earth, and to millions of televisions worldwide.



Okay, so *who* invented the Internet?

The Modern Internet

- Packet Switching: Leonard Kleinrock (UCLA) thinks he did, Donald Davies and Paul Baran, Edelcrantz's signalling network (1809)
- Internet Protocol: Vint Cerf, Bob Kahn
- Vision, Funding: J.C.R. Licklider, Bob Taylor
- Government: Al Gore (first politician to promote Internet, 1986; act to connect government networks to form "Interagency Network")

The World Wide Web

Available within the network will be functions and services to which you subscribe on a regular basis and others that you call for when you need them. In the former group will be investment guidance, tax counseling, selective dissemination of information in your field of specialization, announcement of cultural, sport, and entertainment events that fit your interests, etc. In the latter group will be dictionaries, encyclopedias, indexes, catalogues, editing programs, teaching programs, testing programs, programming systems, data bases, and – most important – communication, display, and modeling programs. **All these will be – at some late date in the history of networking - systematized and coherent; you will be able to get along in one basic language up to the point at which you choose a specialized language for its power or terseness.**

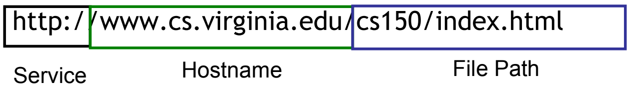
J. C. R. Licklider and Robert W. Taylor, *The Computer as a Communication Device*, April **1968**

The World Wide Web

- Tim Berners-Lee, CERN (Switzerland)
- First web server and client, 1990
- Established a *common language* for sharing information on computers
- Lots of previous attempts (Gopher, WAIS, Archie, Xanadu, etc.)

World Wide Web Success

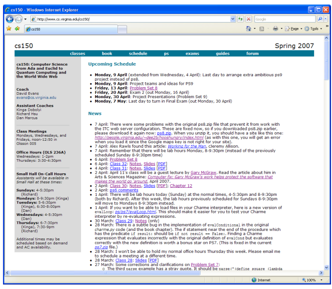
- World Wide Web succeeded because it was **simple**!
  - Didn't attempt to maintain links, just a common way to name things
  - Uniform Resource Locators (URL)



http://www.cs.virginia.edu/cs150/index.html

HyperText Transfer Protocol

HyperText Transfer Protocol



cs150 - Windows Internet Explorer
http://www.cs.virginia.edu/cs150/
Google
cs150
Page
Tools
Internet
100%

cs150 Spring 2007

classes book schedule ps exams guides forum

cs150: Computer Science from Ada and Euclid to Quantum Computing and the World Wide Web

Coach
David Evans
evans@cs.virginia.edu

Assistant Coaches
Kinga Dobolyi
Richard Hsu
Dan Marcus

Class Meetings
Mondays, Wednesdays, and Fridays, noon-12:50 in Olsson 005

Office Hours (OLS 236A)
Wednesdays: 1-2pm
Thursdays: 3:30-4:30pm

Small Hall On-Call Hours
Assistants will be available in Small Hall at these times:

Sundays: 4-5:30pm (Richard)
Mondays: 8-9:30pm (Kinga)
Tuesdays: 5-6:30pm (Kinga), 6:30-8:00pm (Dan)
Wednesdays: 4-5:30pm (Dan)
Thursdays: 6-7:30pm (Kinga), 7:30-9pm (Richard)

Additional times may be scheduled based on demand and AC availability.

Upcoming Schedule

- **Monday, 9 April** (extended from Wednesday, 4 April): Last day to arrange extra ambitious ps9 project instead of ps8.
- **Monday, 9 April**: Project teams and ideas for PS9
- **Friday, 13 April**: Problem Set 8
- **Friday, 20 April**: Exam 2 (out Monday, 16 April)
- **Monday, 30 April**: Project Presentations (Problem Set 9)
- **Monday, 7 May**: Last day to turn in Final Exam (out Monday, 30 April)

News

- 7 April: There were some problems with the original ps8.zip file that prevent it from work with the ITC web server configuration. These are fixed now, so if you downloaded ps8.zip earlier, please download it again now: ps8.zip. When you unzip it, you should have a site like this one: *http://people.virginia.edu/~dee2b/hooshungry/index.html* (as with this one, you will get an error when you load it since the Google maps key is not right for your site).
- 7 April: Alex Rawls found this article: *Working for the Man*, cJeremy Allison.
- 7 April: Remember that there will be lab hours Monday, 8-9:30pm (instead of the previously scheduled Sunday 8-9:30pm time)
- 6 April: Problem Set 8
- 6 April: Class 32: Notes, Slides [PDF]
- 4 April: Class 31: Notes, Slides [PDF]
- 2 April: April 11's class will be a guest lecture by Gary McGraw. Read the article about him in Arts & Sciences Magazine: *Computer fix: Gary McGraw's work helps protect the software that makes the world go 'around*, April 2007.
- 2 April: Class 30: Notes, Slides [PDF]; Chapter 12
- 2 April: ps6 comments
- 1 April: There will be lab hours today (Sunday) at the normal times, 4-5:30pm and 8-9:30pm (both by Richard). After this week, the lab hours previously scheduled for Sundays 8-9:30pm will move to Mondays 8-9:30pm instead.
- 1 April: If you want to be able to load files in your Charme interpreter, here is a new version of `evalLoop`: *ps/ps7/evalLoop.html*. This should make it easier for you to test your Charme interpreter by re-evaluating expressions.
- 30 March: Class 29: Notes (only)
- 28 March: There is a subtle bug in the implementation of `evalConditional` in the original *charme.py* code (and the book chapter). The if statement near the end of the procedure which has the predicate `if result:` should be `if not result == False:`. Finding a Charme expression that evaluates incorrectly with the original definition of `evalCond` but evaluates correctly with the new definition is worth a bonus star on PS7. (This is fixed in the current *ps7.zip* file.)
- 28 March: I won't be able to hold my normal office hours Thursday this week. Please email me to schedule a meeting at a different time.
- 28 March: Class 28: Slides [PDF]
- 27 March: Some corrections and clarifications on Problem Set 7:
  - The third `parse` example has a stray quote. It should be `parse("(define square (lambda`

Client (Browser)

Apache
HTTP SERVER PROJECT

Server

GET /cs150/index.html HTTP/1.0

```
<html>
<head>
…
```

Contents of file

HTML
HyperText Markup Language

HTML: HyperText Markup Language

- Language for controlling presentation of web pages
- Uses formatting tags
  - Enclosed between < and >
- Not a universal programming language

Proof: no way to make an infinite loop

HTML Grammar Excerpt

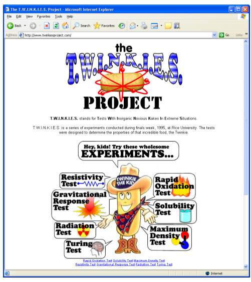
Document ::= **<html>** *Header Body* **</html>**
Header ::= **<head>** *HeadElements* **</head>**
HeadElements ::= *HeadElement HeadElements*
HeadElements ::=
HeadElement ::= **<title>** *Element* **</title>**

Body ::= **<body>** *Elements* **</body>**
Elements ::= *Element Elements*
Elements ::=
Element ::= **<p>** *Element* **</p>**
Make *Element* a paragraph.
Element ::= **<center>** *Element* **</center>**
Center *Element* horizontally on the page.
Element ::= **<b>** *Element* **</b>**
Display *Element* in **bold**.
Element ::= Text

What is a HTML interpreter?

Popular Web Site: Strategy 1 Static, Authored Web Site

Web Programmer, Content Producer



The T.W.I.N.K.I.E.S. Project - Microsoft Internet Explorer

http://www.twinkiesproject.com/

the T.W.I.N.K.I.E.S. PROJECT

T.W.I.N.K.I.E.S. stands for Tests With Inorganic Noxious Kakes In Extreme Situations.

T.W.I.N.K.I.E.S. is a series of experiments conducted during finals week, 1995, at Rice University. The tests were designed to determine the properties of that incredible food, the Twinkie.

Rapid Oxidation Test Solubility Test Maximum Density Test
Resistivity Test Gravitational Response Test Radiation Test Turing Test

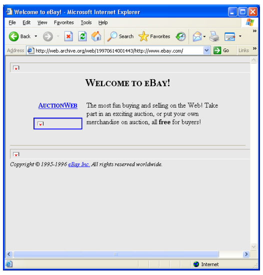
http://www.twinkiesproject.com/

Drawbacks:
- •Have to do all the work yourself
- •The world may already have enough Twinkie-experiment websites

Popular Web Site: Strategy 2 Dynamic Web Applications

Web Programmer, Content Producer

Seed content and function



Welcome to eBay! - Microsoft Internet Explorer

http://web.archive.org/web/19970614001443/http://www.ebay.com/

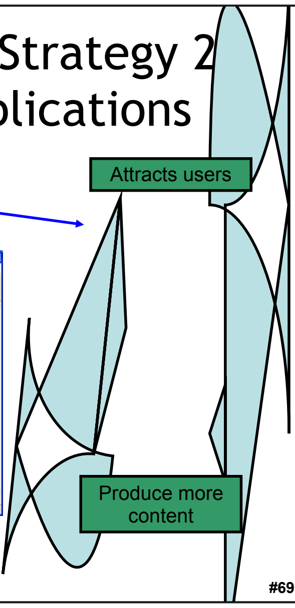
WELCOME TO EBAY!

AUCTIONWEB

The most fun buying and selling on the Web! Take part in an exciting auction, or put your own merchandise on auction, all **free** for buyers!

Copyright © 1995-1996 eBay Inc. All rights reserved worldwide.

eBay in 1997
http://web.archive.org/web/19970614001443/http://www.ebay.com/



Popular Web Site: Strategy 2 Dynamic Web Applications

Seed content and function

Attracts users

Advantages:
- Users do most of the work
- If you're lucky, they might even pay you for the privilege! (not using UVa's servers)

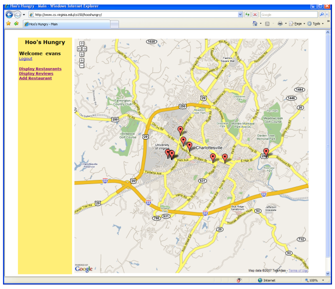
Disadvantages:
- Lose control over the content (you might get sued for things your users do)
- **Have to know how to program a web application**

eBay in 2007

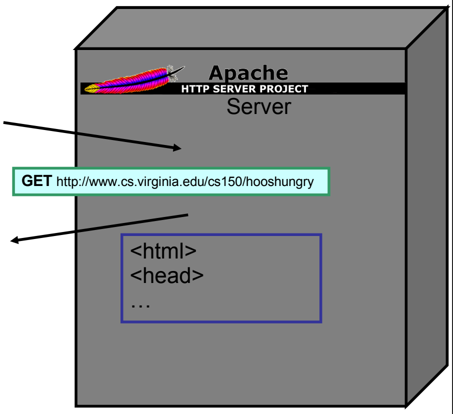
Dynamic Web Sites

- Programs that run on the client's machine
  - Java, JavaScript, Flash, etc.: language must be supported by the client's browser (so they are usually flaky and don't work for most visitors)
  - Used mostly to make annoying animations to make advertisements more noticeable
  - Occasionally good reasons for this: need a fancy interface on client side (like Google Maps)
- Programs that run on the web server
  - Can be written in any language, just need a way to connect the web server to the program
  - Program generates regular HTML – works for everyone
  - (Almost) Every useful web site does this

Dynamic Web Site



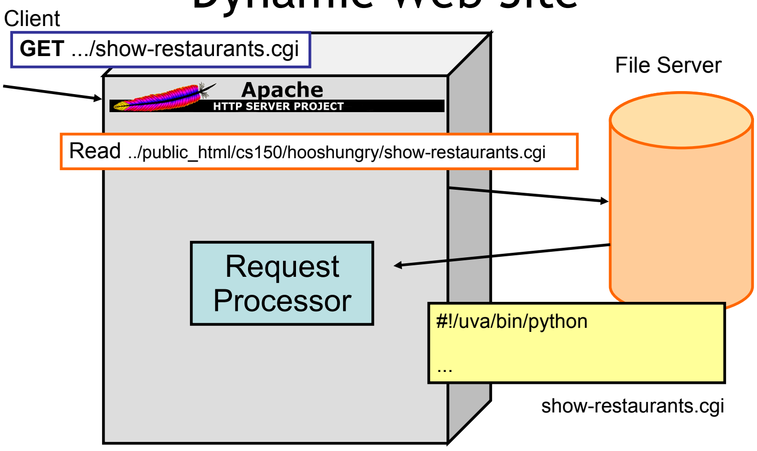
Client (Web Browser)
"HTML Interpreter"



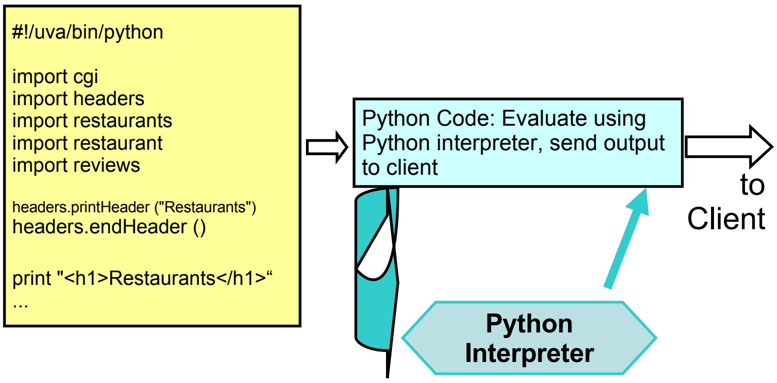
GET http://www.cs.virginia.edu/cs150/hooshungry

```
<html>
<head>
…
```

Dynamic Web Site



Processing a GET Request

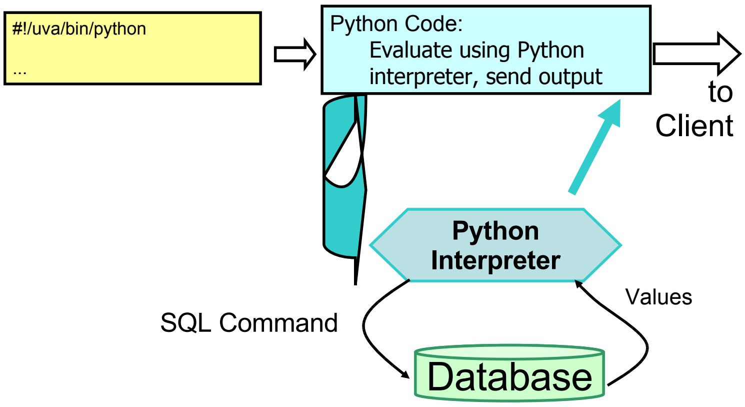


```
#!/uva/bin/python

import cgi
import headers
import restaurants
import restaurant
import reviews

headers.printHeader ("Restaurants")
headers.endHeader ()

print "<h1>Restaurants</h1>“
...
```

Python Code: Evaluate using Python interpreter, send output to client
to Client
Python Interpreter

Using a Database

- HTTP is **stateless**
  - No history of information from previous requests
- We probably need some state that changes as people visit the site
- That’s what databases are for - store, manipulate, and retrieve data



```
#!/uva/bin/python
...
```

SQL

- Structured Query Language (**SQL**)
  - (Almost) all databases use it
- Database is tables of fields containing values
- All fields have a type (and may have other attributes like UNIQUE)
- Similar to procedures from PS5

Homework

- Problem Set 9 Team Requests Friday Apr 10th
- Problem Set 8 due Monday April 13th