Advanced Lecture on Internet Applications

6. Text based Communication:

Character Code and Internationalization (3)

e-mail, SMTP, MIME

Masataka Ohta

mohta@necom830.hpcl.titech.ac.jp

ftp://ftp.hpcl.titech.ac.jp/appli6e.ppt

Character Code

- an encoding (digitization) rule for strings using characters of a character set
    - not merely assign code (number) to characters
        - with finite state, can be simply so
- the number of characters of a character set matters
    - if large, many bits are necessary
    - if small, many characters can’t be represented
        - small differences between similar characters can’t be represented

ISO 8859/1

- western European 96 Latin and symbol characters are added to ASCII
  - forcibly extend ISO 2022
    - byte value range from 33~126 to 32~ 127
- is strange in various ways
  - only one currency symbol, NBSP
  - no capital letter for “ÿ”

ISO-2022-JP (JUNET Code, rfc1468)

- developed to use Kanji in JUNET (UUNET)
- conformant to ISO 2022
- transmit all the characters with 7bit byte
- G0 character set is switched by escape sequences
  - initially ASCII
  - must reset to ASCII (or JIS X 0201) at line end
    - state maintenance between lines unnecessary

Character Sets of ISO-2022-JP

- ASCII
- JIS X 0201 (Latin)
- JIS 0208 (78 and 83 vertions)
- JIS X 0201 (Kana) a.k.a. hankaku kana is not included

Complexity and Simplicity of JIS X 0208 (1)

- large number of characters
- horizontal and vertical
  - vertical was not supported so seriously
- single (left to right) directional only
- ligature (variation of character shape by previous/next characters) is not necessary
  - though circle mark for composition exists
    - not really used for composition

Complexity and Simplicity of JIS X 0208 (2)

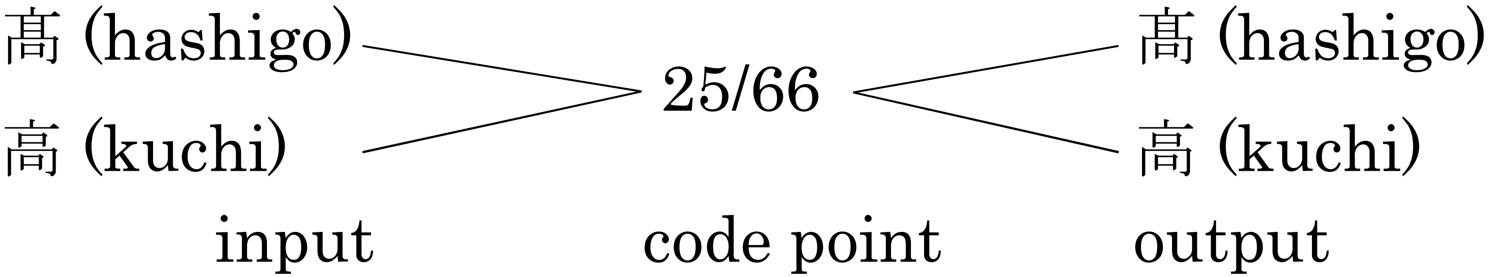
- no commonly shared recognition for character identifications and character shapes
  - is the serious problem
- correspondence between hiragana/katakana characters is not so clear and regular「ブ」
- diacritical (?) marks「゛゜」are precombined
- character width can be constant
- widely spread and usable everywhere

Ambiguity of Character Identification (Unification)

- JIS X 0208 does not specify small differences of character shapes
  - 「国」 and 「國」 are different characters
  - 「竜」 and 「龍」 are different characters
  - 「高」 and 「髙」 are the same character
  - 「Ａ」 and 「ａ」? 「Ａ」 and 「Α(alpha)」?
- character shapes of 「高」 and 「髙」 are unified in JIS X 0208

What is Unification?

- in JIS Kanji specification
  - one code point contains multiple character shapes
    - notable example: 「高(kuchi)」 and 「髙(hashigo)」
  - same code point is used upon input
    - may output either character shape (?)



髙 (hashigo)

高 (kuchi)

25/66

髙 (hashigo)

高 (kuchi)

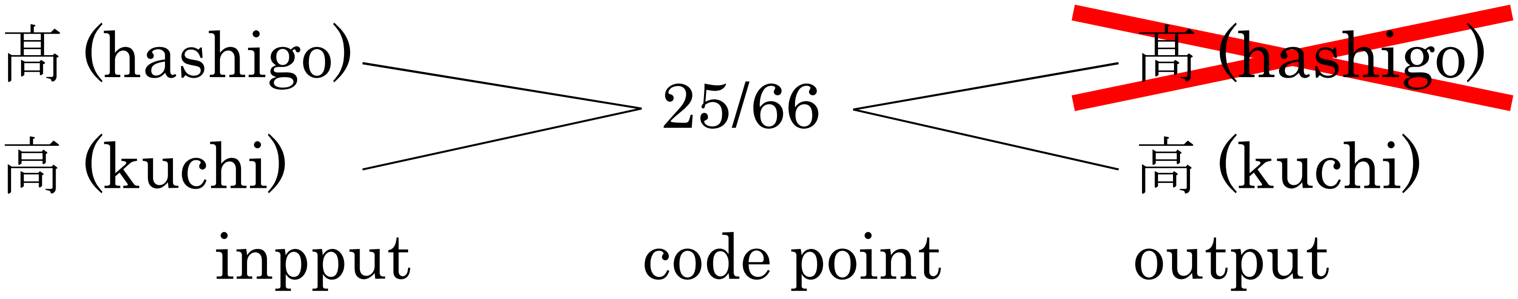
input

code point

output

What is Unification?

- same code point is used upon input
  - may be OK
- may output either character shape ???
  - in practice, no implementation output **髙** (hashigo)
    - but aren’t input/output symmetric?



髙 (hashigo)

高 (kuchi)

25/66

~~髙 (hashigo)~~

高 (kuchi)

inppput

code point

output

Definition of Unification in JIS

- “treatment of shape variations of kanji” in 78JIS
  - character shape presented at a code point allows for certain variations and should be considered to be a representative
- in 97JIS, like Unicode
  - do not distinguish multiple character shapes and assign the same code point
  - at each code point, character shapes unified to the point are not distinguished

The Problem of Unification

- used as a reason not to distinguish CJK kanji by ISO 10646 (Unicode)
  - can output any of CJK kanji
    - or, can not output distinguised CJK kanji
- can not add existing (unified to existing code point) kanji
  - 「髙 (hashigo)」was not added by extension of JIS kanji for classes 3/4

Unicode

- standard developed in US to encode all the characters in the world with 16bits
  - impossible and unnecessary as ISO 2022 exists
    - no state maintenance of ISO 2022 necessary?
    - 16 bit space is too small
      - even some European characters are represented by base characters combined with diacritical characters (already stateful)
      - CJKT characters (each >50,000 characters) are unified
    - as other characters from the world is collected, the space overflowed

ISO 10646

- was standard developed in ISO to encode all the characters in the world with 31bits
  - simple encoding for all the characters in the world without unification
- was overridden by Unicode with CJKT unification to be useless within international context

What is Character Code?

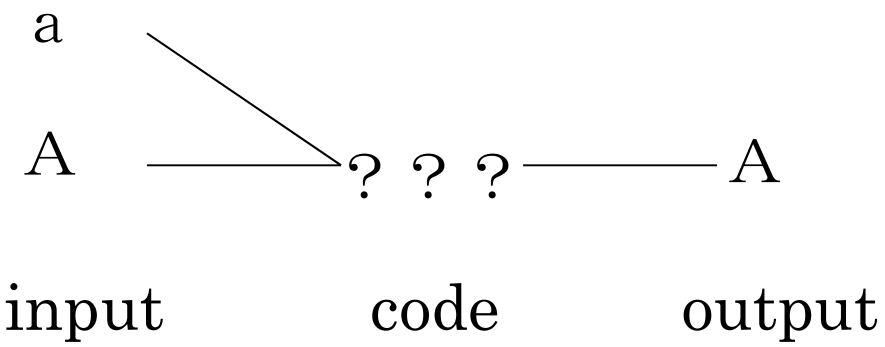
- rule to correspond string and sequence of numbers
  - definition to broad to be useless
  - character input/output possible as images without coding
- finite state rule to correspond string and sequence of numbers
  - without finite stateness, search is practically impossible
  - plain text should be finite state, structure text may have more complex state
- what is character encoding?

Characters and Unification (1)

- unification is a concept first appeared in JIS kanji code?
  - because JIS kanji has very large number of characters!!
    - wrong!
    - unification occurs even with Latin character codes

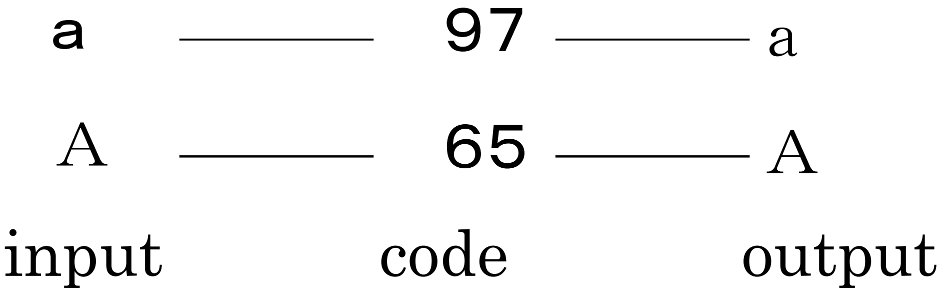
Latin Character Code and Unification

- when a byte was 6 bit
  - – only capital Latin letters encoded?
  - – how can small letters in text encoded?
    - upon input, coded as capital letter
    - upon output, printed as capital letter



Breaking Unification of Latin Character Code

- as a byte becomes 7 bit (ASCII)
- small/capital Latin characters are separately coded



a — 97 — a

A — 65 — A

input code output

  - migrate to mixed small/capital letter environment
  - files created in 6bit/byte era is used as is
  - JIS was wrong not to add 「髙 (hashigo)」

Are Latin Capital/Small Letters Same Character?

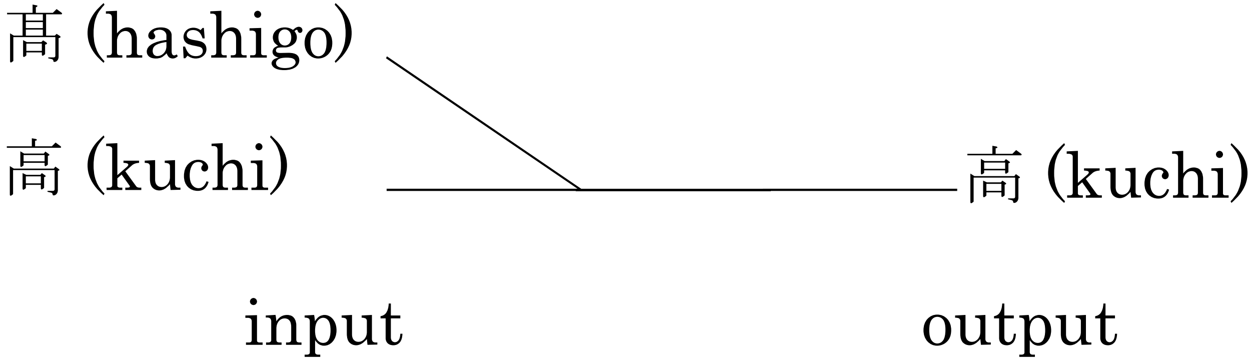
- not a problem in 6bit/byte era
- in early era of UNIX, use of small letters strongly promoted
  - on capital letters only output devices, “A” was output as “＼A” “a” as “A”
  - by default, search commands (grep) distinguish capital/small letters
    - UNIX users tend to think capital/small letters different
- recent OSes do not distinguish them, by default
  - seemingly, common sense of natives?

Characters and Unification (2)

- unification is a concept first appeared in character code?
  - unification already occurs with type setting and type writing!

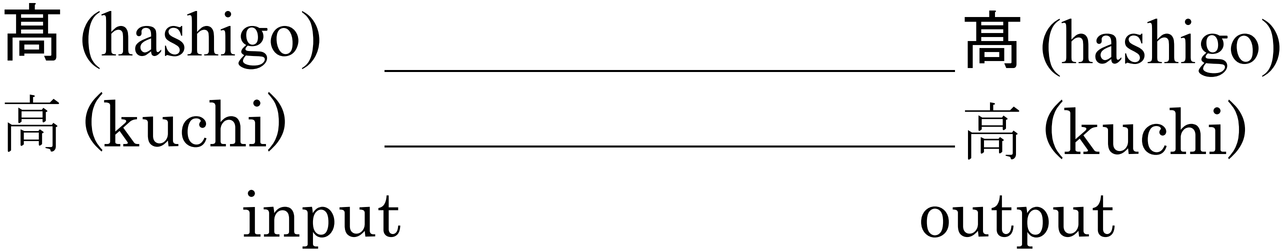
Type Setting and Unification

- when printing was by combining types
  - with type set without「髙 (hashigo)」type
    - upon input, type set as「高 (kuchi)」
    - upon output, printed as「高 (kuchi)」



Breaking Unification with Types

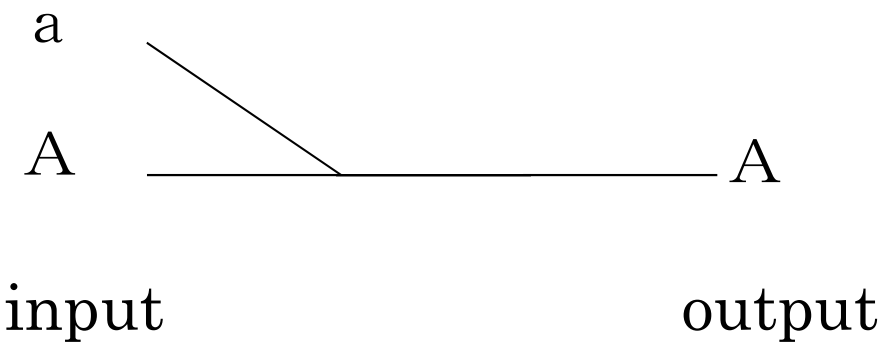
- how「髙 (hashigo)」 is type set, if type of「髙 (hashigo)」is added?
    - upon input, type set as「髙 (hashigo) 」
    - upon output, printed as「髙 (hashigo) 」



- existing printed materials are used as is

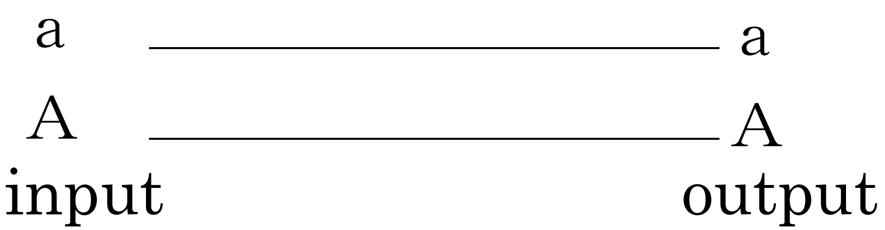
Typewriters and Unification

- by cheap (toy) typewriters
  - can type capital characters only
  - how can small letters in text treated?
    - upon input, typed as capital letter
    - upon output, printed as capital letter



Breaking Unification with Typewriters

- with full fledged type writers
  - Latin capital/small letters may be typed
    - upon input, typed as small letter
    - upon output, printed as small letter



a

A

input

a

A

output

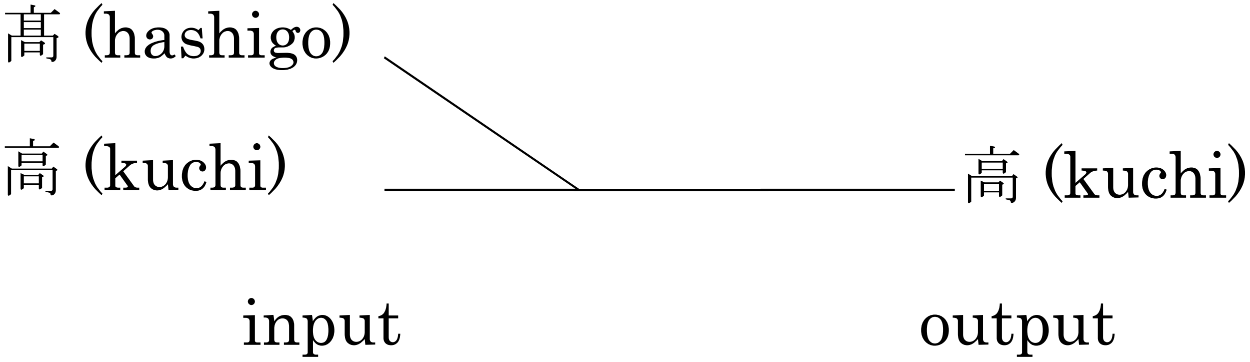
- printed materials by cheap (toy) typewriters are used as is

Characters and Unification (3)

- unification is a concept first appeared in type setting?
  - unification already occurs with hand written characters

Hand Written Characters and Unification

- a person who think「髙 (hashigo)」and「高 (kuchi)」are the same character?
    - upon input, recognize as「高 (kuchi)」
    - upon output, hand write as「高 (kuchi)」



髙 (hashigo)

高 (kuchi)

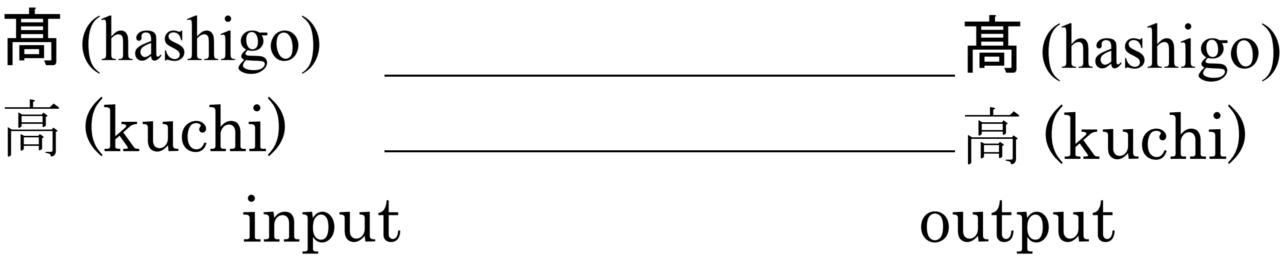
高 (kuchi)

input

output

Breaking Unification with Hand Writing

- after the person recognizes「髙 (hashigo)」 is different from「高 (kuchi) 」
    - upon input, recognize as「髙 (hashigo) 」
    - upon output, hand write as「髙 (hashigo) 」



髙 (hashigo) —— 髙 (hashigo)

高 (kuchi) —— 高 (kuchi)

input output

- - exsiting hand written materials are used as is

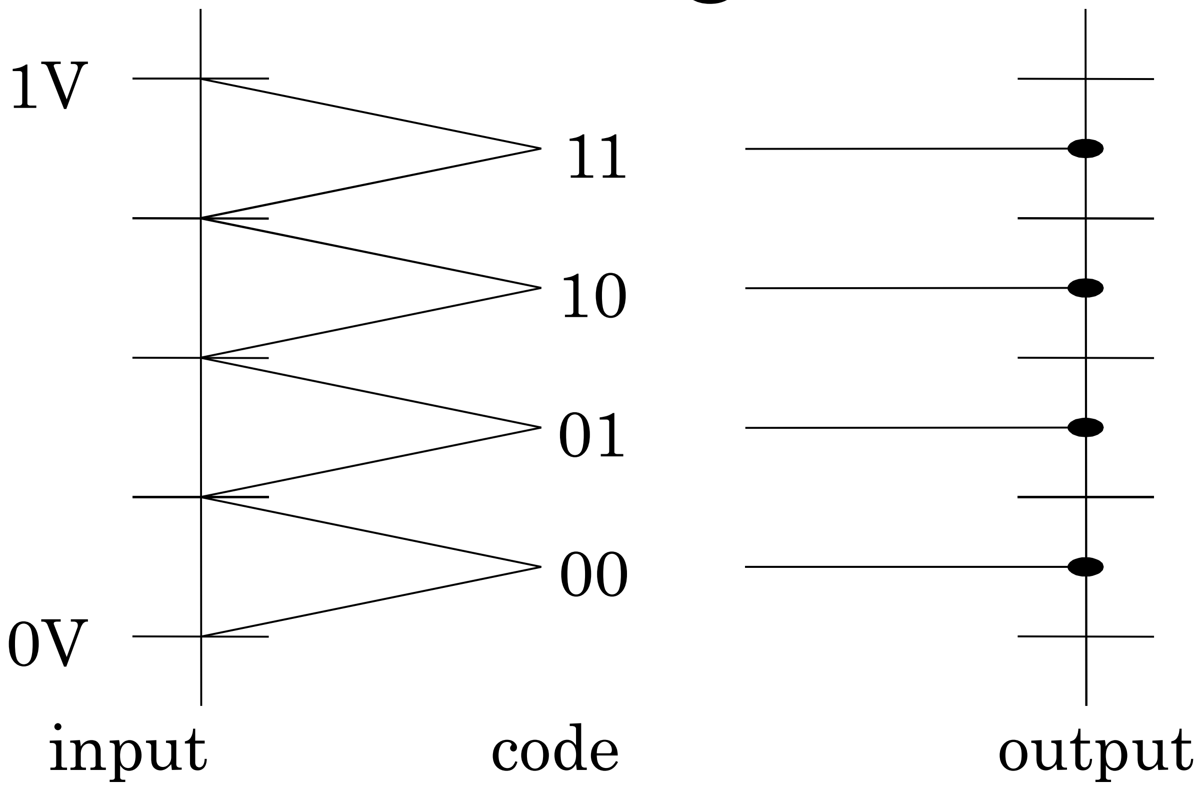
Characters and Unification (4)

- unification is a concept implied by characters
- then, what are characters
  - what is the difference to images
    - images are analog
    - characters are abstract concept and is digital!
      - isn’t unification a concept implied by digitization (ignoring small differences)?
      - compare with AD/DA conversion of voltages
        - » 0~1V, 2bit, linear

Digital and Analog

- digital ignores small differences
  - can remove noise
- language (incl. spoken one) is digital
  - voice and song are digital
- character is digital
  - can represent very subtle feelings with 17 characters
  - calligraphy is analog
- to what extent, small differences should be ignored? (how many bits should be used?)

Ideal AD/DA Conversion of Voltages



In Ideal AD/DA Conversion

- 0~1V is equally divided by 4 to be range of each code
- 0.0625V, 0.125V and 0.1875V are
  - encoded as representative voltage (0.125V)
    - 0.0625V is not error
      - 1.5V may be treated as error
  - decoded as representative voltage (0.125V)
    - never output as 0.0625V
      - to minimize average error

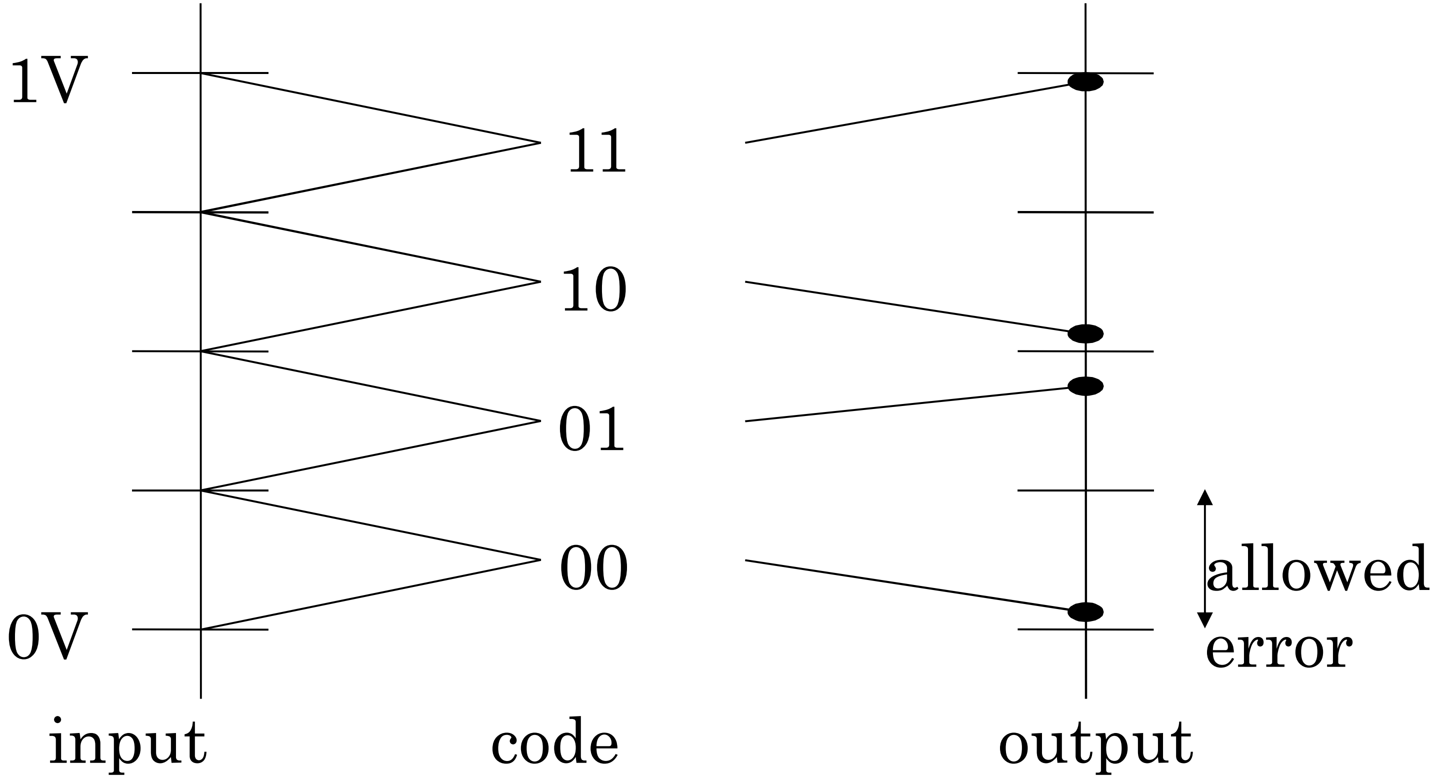
Character Input/Output as Ideal AD/DA Conversion

- character shapes belonging to each code point is specified by standards
- with JIS, both「髙 (hashigo)」and「高 (kuchi)」are
  - encoded as representative character (「高 (kuchi)」)
    - 「髙 (hashigo)」is not a error
      - ununifiable characters may be treated as error
    - encoded as representative character (「高 (kuchi)」)
    - never output as「髙 (hashigo)」
      - 「高 (kuchi)」minimize error on expected output

What is Unification?

- quantization error by digitaization!
  - occurs only at input
  - may not output all the shapes unified in a code point
    - character output as ideal DA conversion
      - only representative character shape may be output
    - similar to unification with Latin character code, type setting, type writing and hand writing
    - character output as practical DA conversion?
      - involves output error, cofused with unification at output

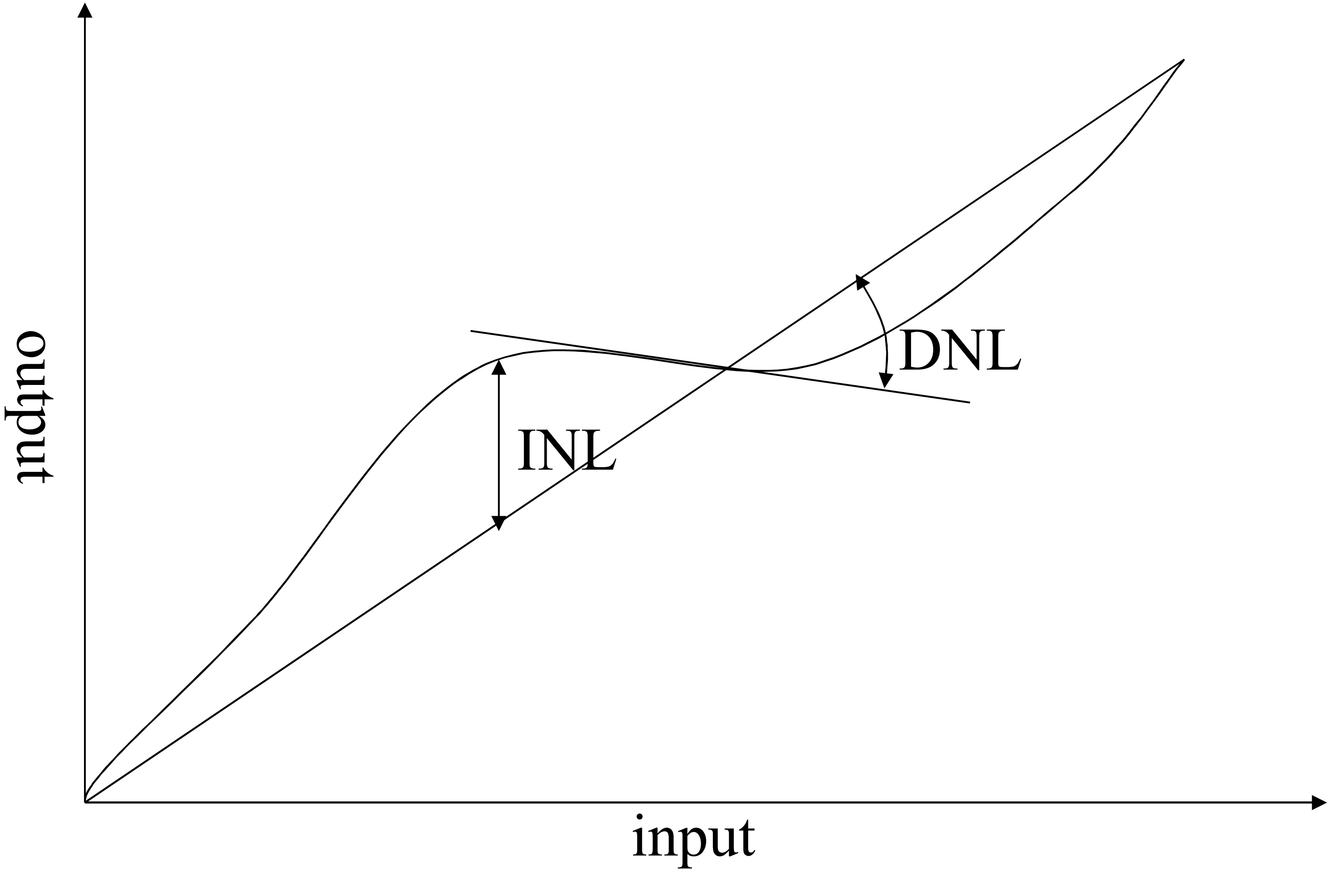
Practical AD/DA Conversion of Voltages



In Practical AD/DA Conversion

- 0.0625V, 0.125V and 0.1875V are
  - encoded as representative voltage (0.125V) but decoded with allowed error from the representative voltage
    - typical allowed error is ±1/2LSB
      - monotonity is assured
    - other allowed error is possible and is actuall used
      - e. g., ±1/4LSB, typically when the number of bits is small,
      - e. g., INL ±6LSB and DNL ±2LSB, typically when the number of bits is large,

INL(Integral Non-Linearity) and DNL(Differential Non-Linearity)

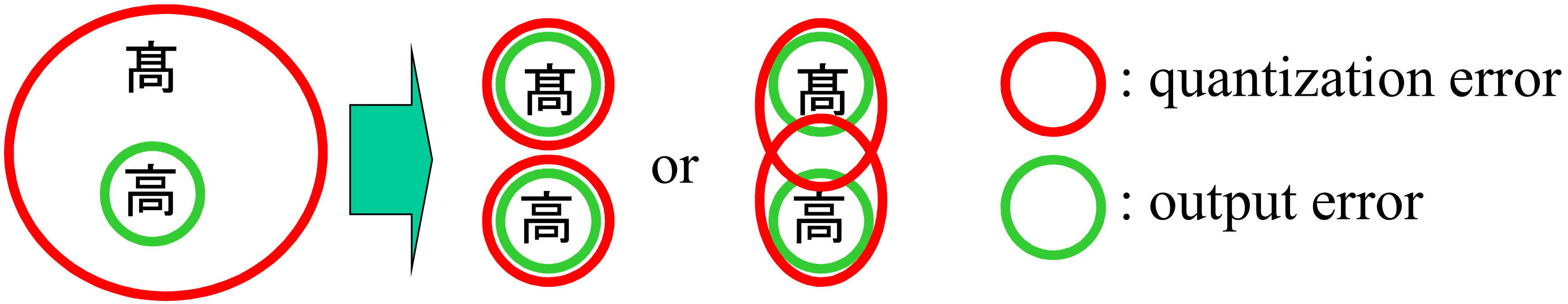


Character Output as Practical DA Conversion

- both「髙 (hashigo)」and「高 (kuchi)」are
  - encoded as representative character (「高 (kuchi)」) but output with allowed output error from the representative character
    - allowed output error specified by JIS X 0208:1997
      - "must be distinguishable from other characters"
        - » corresponds to allowed error of ±1/2LSB
    - other error grade must be allowed
      - as an industrial standard for poor output device
      - when the number of character is large, similar characters may be output with the same shape (corresponds to large DNL)

Increasing # of bits of AD/DA Conversion and # of Characters

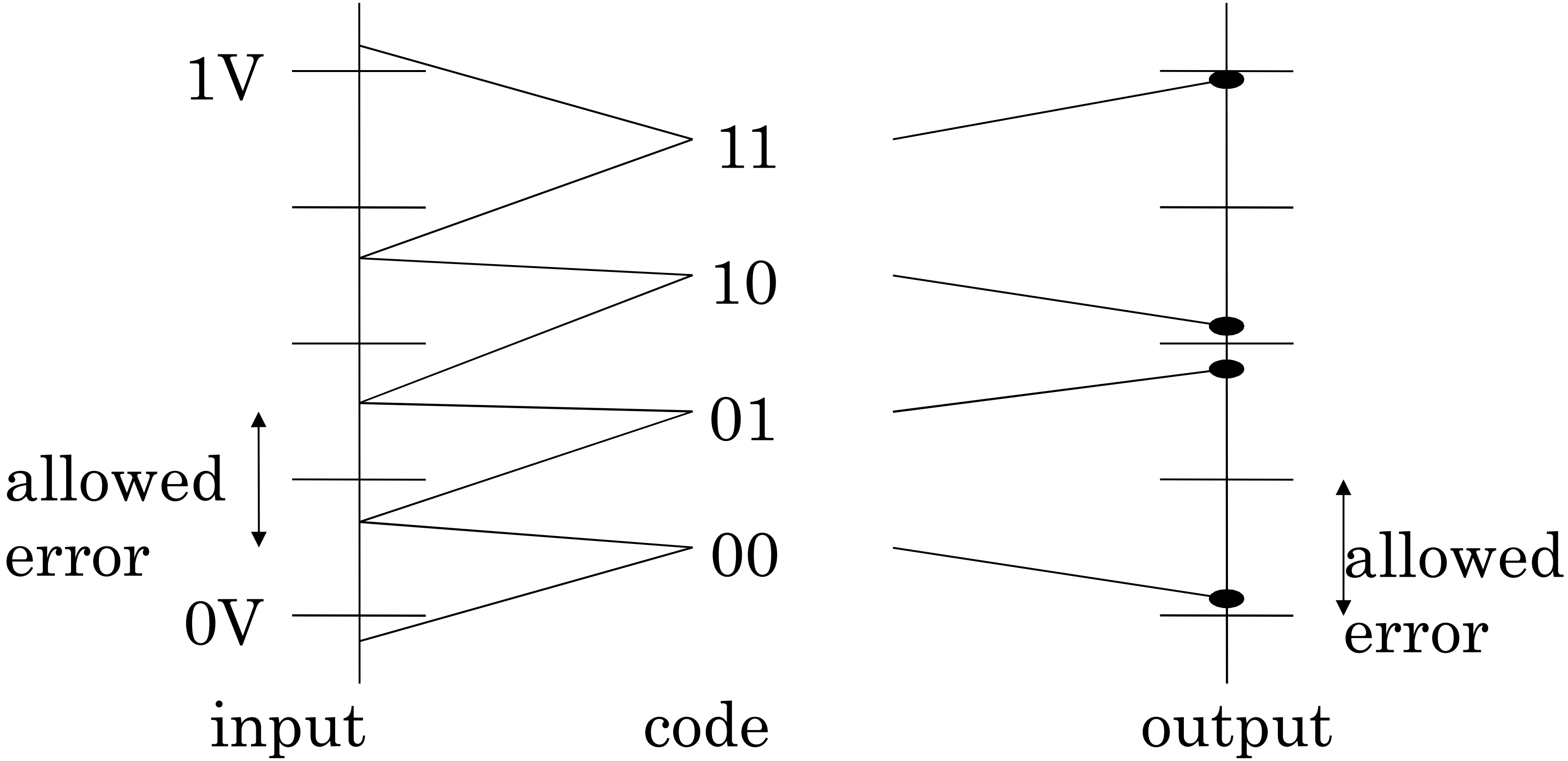
- may extend voltage range (0~1V⇒0~2V)
  - addition of totally new characters
- may subdivide voltage range
  - separate existing unified characters
    - separate「髙 (hashigo)」from「高 (kuchi)」



🟥 : quantization error

🟩 : output error

Practical AD/DA Conversion of Voltages



In Practical AD/DA Conversion

- 0.0625V, 0.125V and 0.1875V are
  - encoded as representative voltage (0.125V) but decoded with allowed error from the representative voltage
    - typical allowed error is ±1/2LSB
      - monotonity is assured
    - other allowed error is possible and is actuall used
      - e. g., ±1/4LSB, typically when the number of bits is small,
      - e. g., INL ±6LSB and DNL ±2LSB, typically when the number of bits is large,

In Practical AD/DA Conversion

- error is inevitable both to input and output
  - industrial standard must tolerate error
    - input error (noise) may make same voltage to be different code
  - practical equipments has error tolerance
    - same set of representative voltages may have multiple grades of error tolerance
  - error is accumulated with repeated input/outout

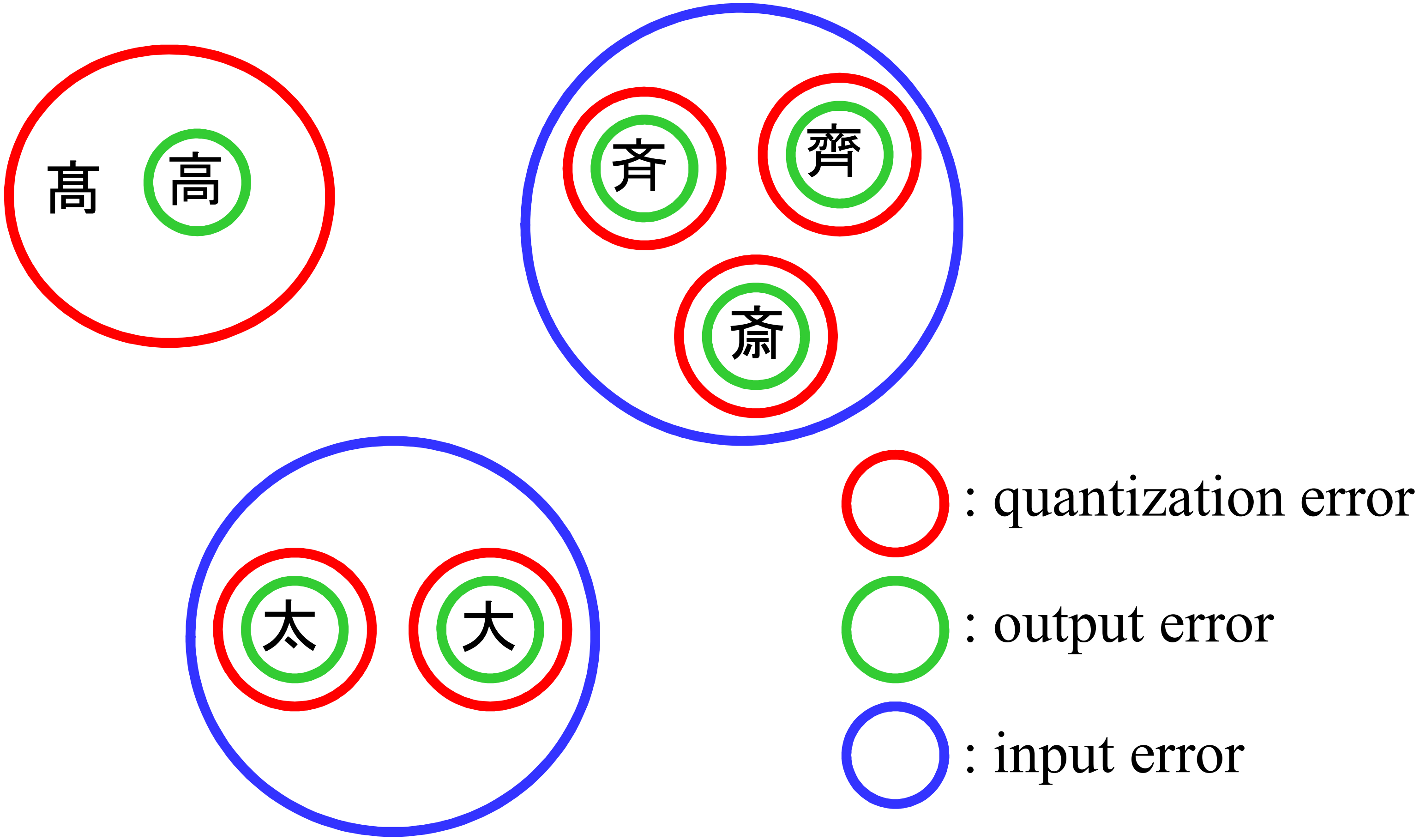
Character Input/Output as Practical AD/DA Conversion

- error is inevitable both to input and output
  - industrial standard must tolerate error
    - input error (noise) may make same image to be different code
  - practical equipments has error tolerance
    - same set of representative shapes may have multiple grades of error tolerance
  - error is accumulated with repeated input/outout
    - similar to error accumulation by repeated copying of a book by hand writing

Noise

- should be inevitable to character input/output
- thermal noise
  - wrong type setting, wrong kana kanji conversion
  - reduced by careful input (lower temprature)
- shot noise
  - output error by using small number of dots
  - reduced by increasing the number of dots (increase current)

Realistic Use of Kanji



○ : quantization error

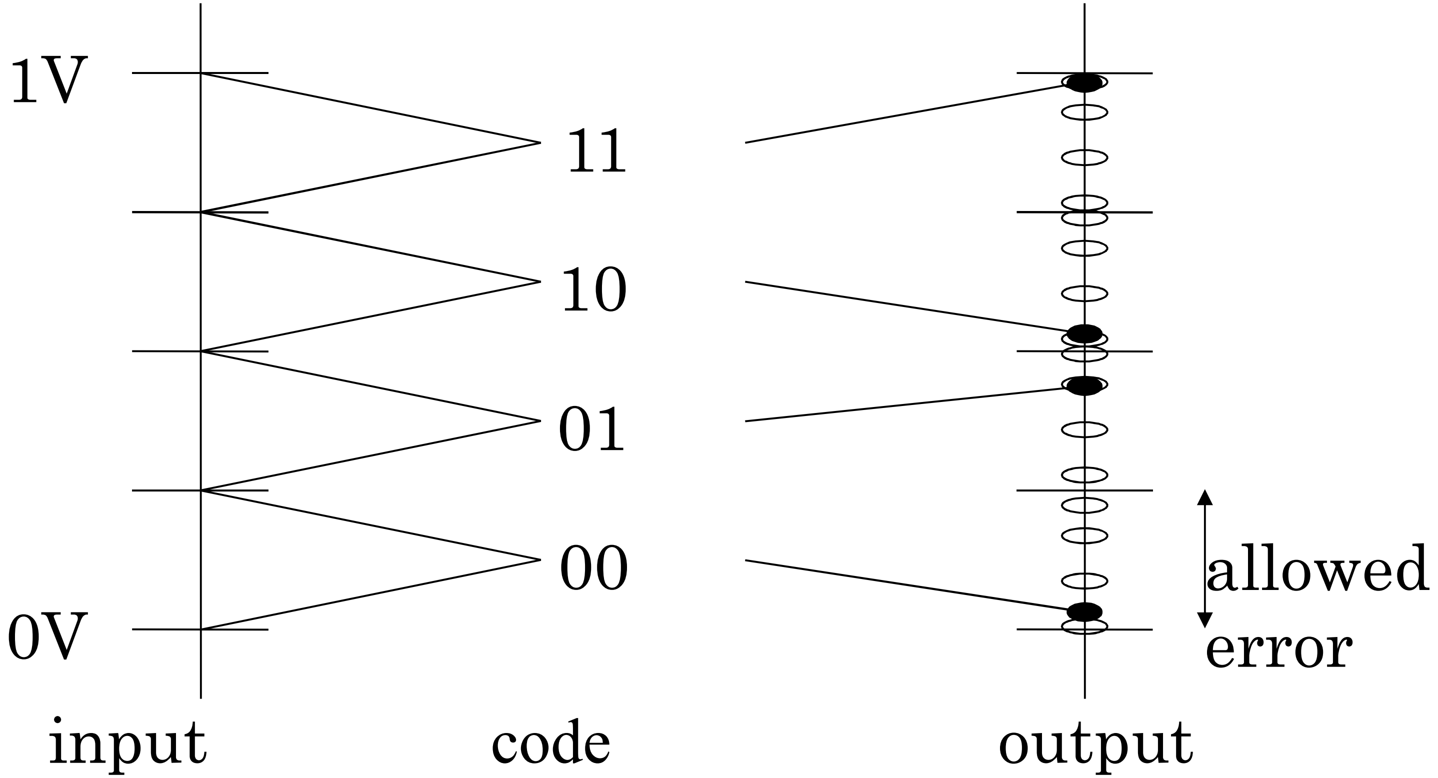
○ : output error

○ : input error

Quantization Error and Input/Output Error

- quantization error much smaller than input/output error is not very meaningful
- modern display has large # of pixels
  - output error is small
  - character set with large # of characters meaningful

Multiple Representative Voltages Causes Large Output Error



⬭ : representative voltage

Unicode?

- actively allow all the unified character shapes to be used as representative characters
  - causes large output error
  - a lot larger than capabilities of typical modern output devices
    - not usable without CJK localization
    - not internationalized at all

What’s Wrong with the Current JIS Kanji Standards

- code points specify multiple representative characters and no input error specified
  - a code point should specify only representative character shapes
    - range of unification may depend on common sense
      - e.g. may encode「髙 (hashigo)」into code point of「高 (kuchi)」
    - end users should have their own specifications
- output error allowance of ±1/2LSB only is wrong
  - good output devices should output exact shapes of representative character shapes
    - less capable devices may output different characters with same shape

How to Standardize Representative Character Shapes

- unlike standards for light speed etc.
  - characters are human
    - “1 foot is length of a foot of the king” is OK
    - 常用漢字表 (table of common use kanji) and 康熙字典 (Kangxi dictionary) are the standards
      - other official character shapes may exist
  - may vary in relatively short term
    - upon variation, should representative character shape change or new characters should be added?
      - should existing electric text remain as is

Unification and Search

- unification simplify search?
  - 「高」and「髙」are the same character
- ambiguous search is necessary, anyway
  - 「国」and「國」are different characters
  - 「竜」and「龍」are different characters
  - no different from unification of “A” and “a”
  - 「太田」and「大田」may also match

How to Specify Range of Unification?

- unification range
  - varies person by person, purpose by purpose
- character code has its own ranges of unification
  - like voice codec has its own number of bits of each sample
  - should be judged by common sense
    - often, only shapes of representative characters are specified
- “universal” character code must have the narrowest range of unification
  - to be compatible with other character codes

Presenting CJ Mail under Unicode (UTF-8) Environment

- C mail with GB code (charset=GB2312) and J mail with JIS code (charset=ISO-2022-JP) are properly presented
- kanji with Unicode (charset=UTF-8) is improperly presented
  - when C mail arrives with Unicode
    - JIS kanji is presented with JIS font
    - other kanji is presented with GB font

Treatment of Space by TEX and HTML

- words are separated by one or more space characters or line change in Latin script
  - – people recognize it as a single space
- TEX ignores space and line change characters between kanji characters
- HTML recognizes them as a single space
  - – displayed as “空白 文字”
  - – because of CJKTV (Vietnam) unification
    - Vietnamese script separate words by space

Other Problems of Unicode

- support nested bi-directionality
- not 16 bit character code at all
  - as long as 31 bit
    - unification of kanji not necessary
- optional variation selector is introduced
  - to choose proper character shape character by character
- YEN SIGN problem
  - presented differently in Japan and Korea

Language Tag (rfc1766)

- put standard name to languages
  - extension of ISO 639 ("JA" for Japanese)
- server provide information in language desired by clients
  - by Content-Language header of MIME
- may be used for CJK dis-unification?
  - confuse language and script!!!
    - done so knowingly

Scripts to Represent Japanese

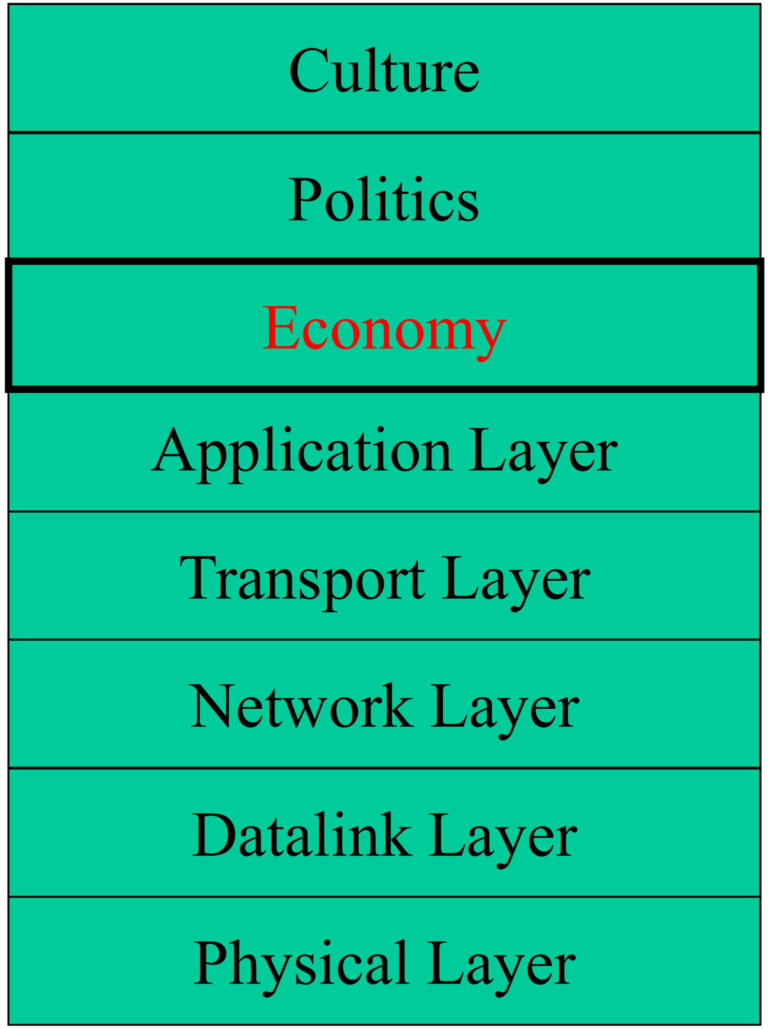
- kana (hiragana, katakana, manyogana)
- mixed kanji kana
- romaji (Hepburn, Monbusho, etc.)
  - “masataka” in French should be “massataka”
- and phonetic representations in various local script systems such as Hangul

“Internationalized” Domain Name

- characters usable in domain (host) names
  - 0-9, A-Z (a-z), “-”
- “internationalized” domain name
  - can use kanji etc. as domain name
- technically, not difficult
  - DNS is 8 bit transparent (though case insensitive)
  - may be encoded to ASCII characters
- used almost not at all

Background of “Internationalized” Domain Name

- domain name (=trade mark) registration is profitable
  - 1 domain under “.com” was $35/year
- new registration may decrease
- TLDs other than “.com” increasing
  - biz, info, museum, name, ...
- domain name registries and registarars want more domain names registered



Culture

Politics

Economy

Application Layer

Transport Layer

Network Layer

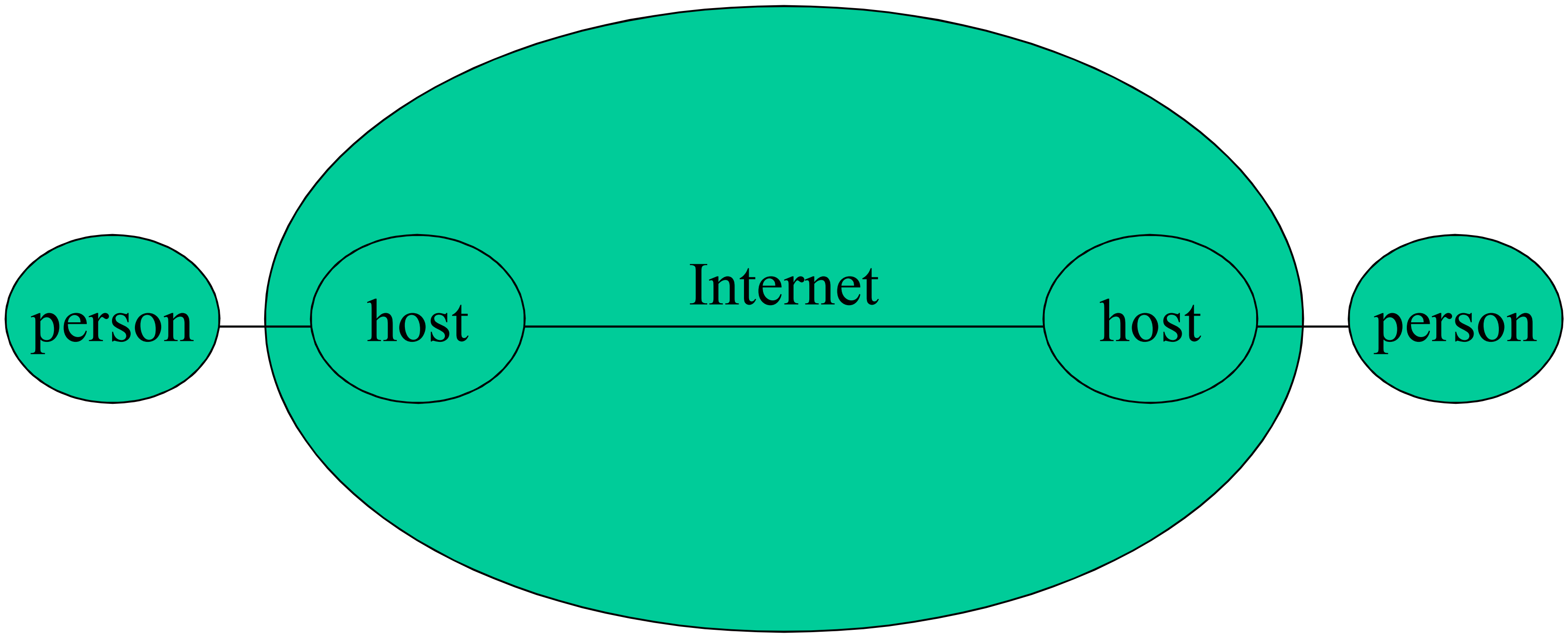
Datalink Layer

Physical Layer

Layering Structure over the Internet!

Internet and Internationalization (I18N)

- Internet
  - connects hosts around the world
- should all the hosts be internationalized?
  - maybe
- Internet
  - connects people around the world
- should all the people be internationalized?
  - maybe, but, ...



end to end principle beyond hosts

Internationally Recognizable Characters

- digits, Latin characters and some symbols
- kanji domain name outside kanji using society
  - can not be recognized
    - even simple identification is hard (「大」、「太」、「犬」)
- on passport and international airline ticket
  - names are represented in latin characters
- the current domain names are international domain name
  - kanji domain name is localized domain name

Various Problems of Kanji Domain Names

- similar names
  - 「国」and「國」,「竜」and「龍」,「高」and「髙」,「－ (hyphen-minus)」と「ー (long vowel) 」
    - identification different culture by culture
- 「漢字.JP」and「漢字．日本」are unnatural
  - if「漢字株式会社」can be automatically converted to「漢字．会社.JP」
    - not a domain name, anymore

Name Spaces other than that of DNS

- though there are a lot of proposals
  - selling names is so profittable
- DNS is enough if names are globally unique
- if dupulication is allowed
  - not different from search engine
  - search engines can search amoung similar names
  - can increase search priority (SEO) by paying to search engine providers

E-mails and rfc822

- RFC822: STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES
- specify format of e-mails
- mail consists of header and body
  - header ends with blank line and body follows

Structure of Header

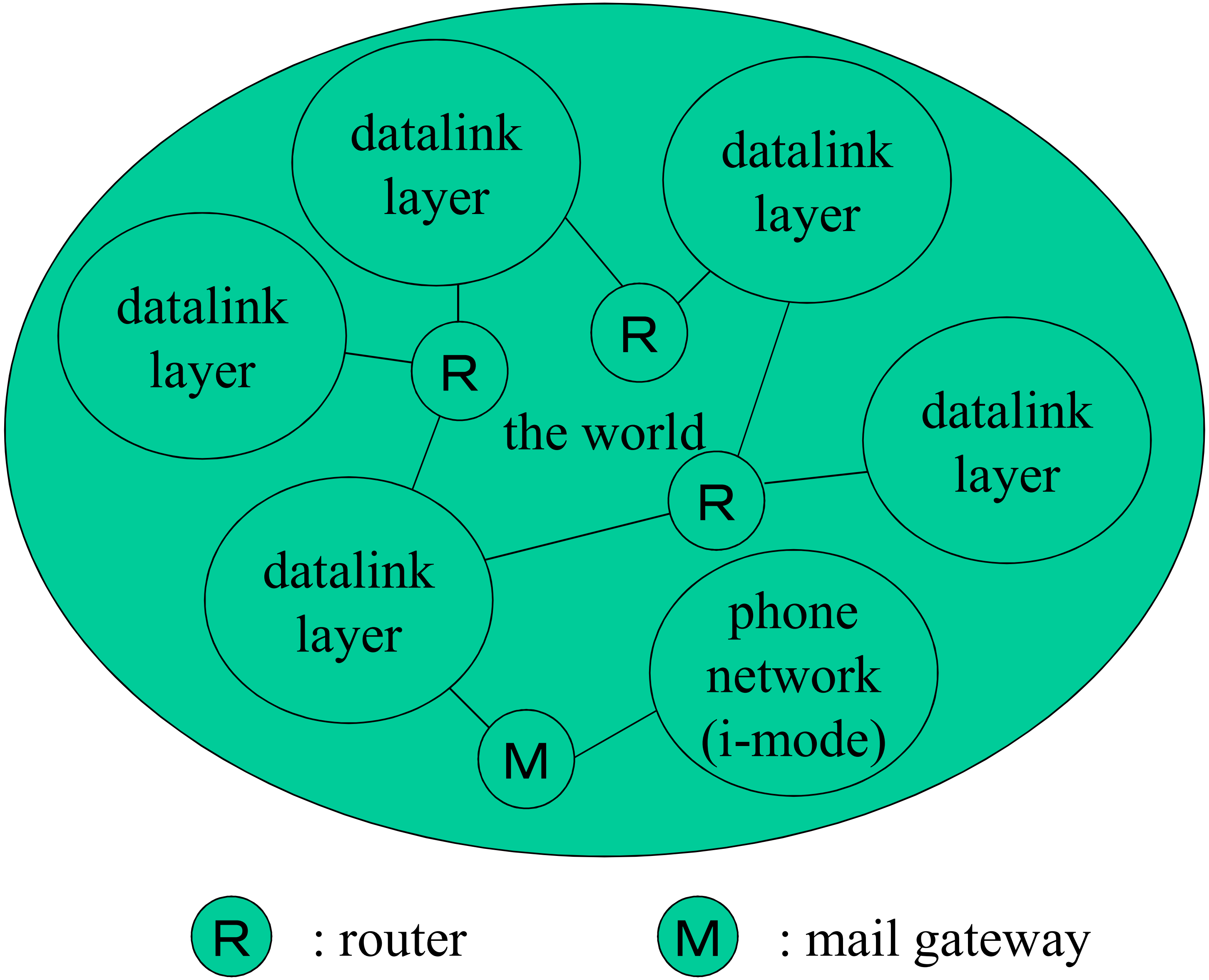
- consists of fields
- fields start with field name terminated by “:”
- line starting with space characters is continuation from previous line
- field content depends on field name
  - mail address (To:, From:, Cc:, etc.)
  - text, “;” and date (Received: etc.)
  - plain text (Subject: etc.)

Examples of Header Field

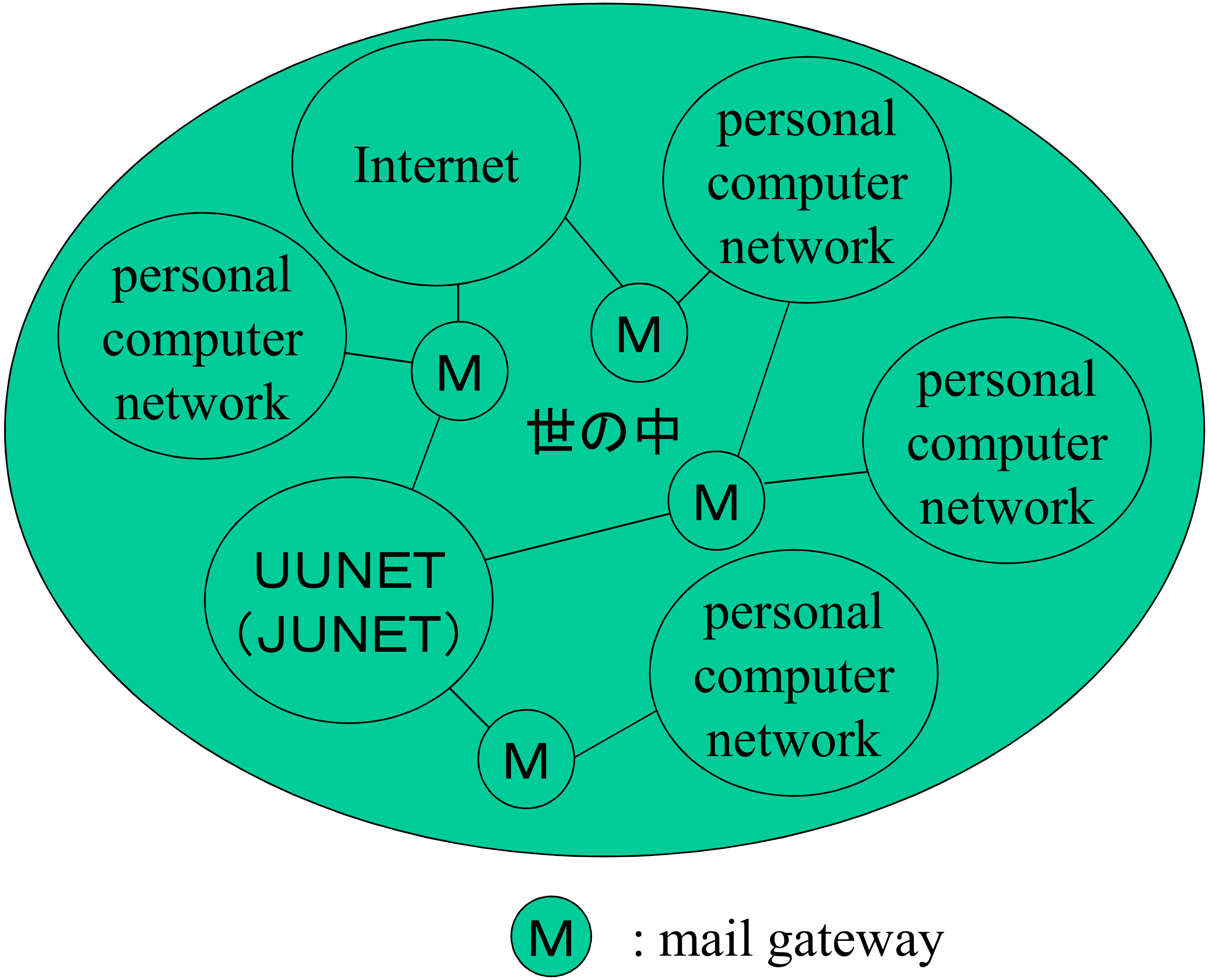
- To:, Cc:, Bcc:
  - destination
- From:
  - address of author
- Sender:
  - address of sender (was often a secretariat)
- Receiver:
  - history of relay

SMTP (rfc821) and e-mails

- Simple Mail Transfer Protocol
  - the protocol to exchange e-mails over the Internet
- use TCP port# 25



e-mail environment involving the Intenet and phone network



Internet

personal computer network

personal computer network

M

M

世の中

M

personal computer network

UUNET （JUNET）

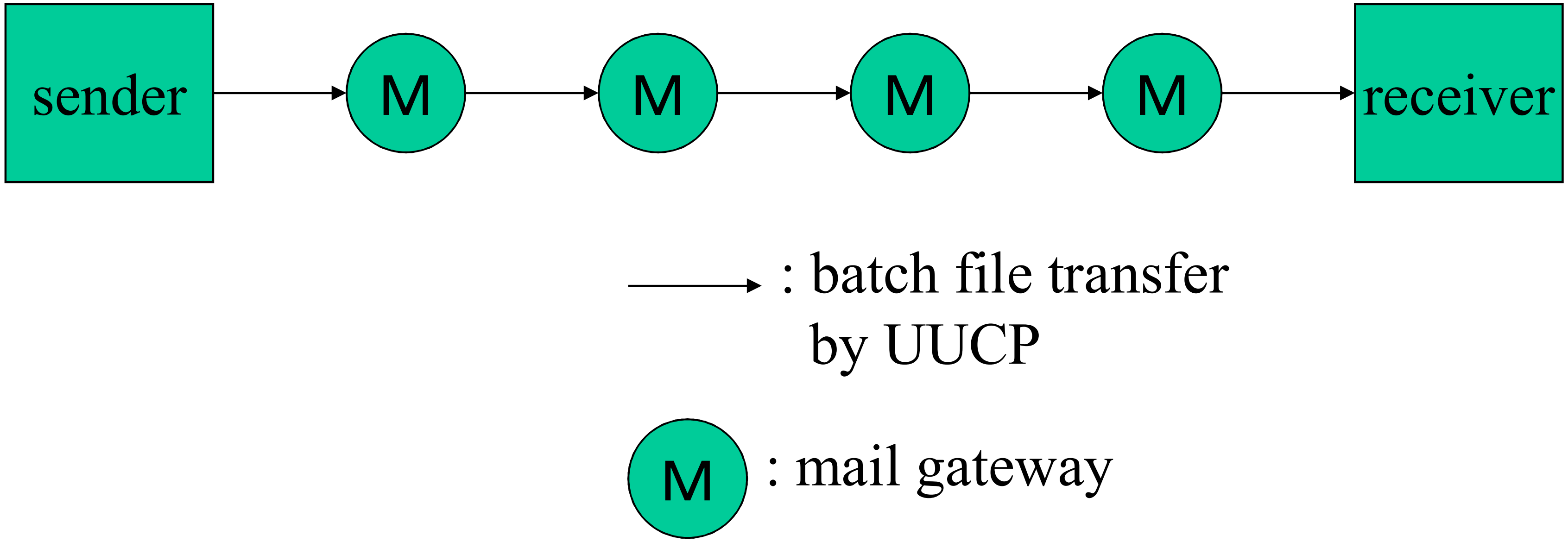
M

personal computer network

M : mail gateway

e-mail environment in the past

Mail Transfer by UUCP



sender → M → M → M → M → receiver

→ : batch file transfer by UUCP

M : mail gateway

End to End Principle and E-mails

- e-mails were used in networks other than the Internet
  - must transfer e-mails outside of the Internet
    - E2E principle not applicable
- e-mail was the most important application to the Internet
  - reliability is important
    - reliability by E2E principle

Mail Relays

- e-mails are relayed over various networks
- destination of SMTP may not be the final destination
  - mail servers temporarily accepts mails
- e-mails may not be read in real time
  - reciepient person may be temporarily absent
- store and forward is OK

DNS and E-mails (rfc974)

- e-mail addresses of the Internet is DNS (domain name) based
- MX RR of a domain specify (multiple) mail servers for the domain
  - MX RR also specify mail server priority
    - if servers with high priority is down, servers with lower priority can receive mails

e-mail and E2E Multihoming

- e-mail (SMTP+DNS (rfc974) supports E2E multihoming at application layer
  - if a mail server have multiple addresses
    - all the addresses are tried
  - it is of course as e-mail was the most important application of the Internet
- DNS also support E2E multihoming
  - all the addresses of NSes are tried

TCP and Command

- commands and replies represented in ASCII strings are exchanged over TCP
  - reply often begins with 3 digits followed by a space and text explaining reply in English
- line is terminated by CR and LF
- data may be sent over the same TCP connection (SMTP) or other TCP connection (FTP)
  - separator for data is necessary for sending over the same TCP

Command and Reply of SMTP

- command
  - HELO, MAIL, RCPT, DATA, SEND, SOML, SAML, RSET, VRFY, EXPN, HELP, NOOP, QUIT
- reply
  - 3 digits (xyz) + message

Commands of SMTP (1)

- HELO
  - initial greetings (notify host name)
- MAIL
  - start of command sequence of a mail
- RCPT
  - specify destination of a mail
- DATA
  - body of mail follows (terminated by “.”)

Commands of SMTP (2)

- SEND, SOML, SAML
  - directly notify user currently logged in
- RSET
  - reset
- VRFY, EXPN
  - verify/expand an address
- HELP, NOOP, QUIT
  - help, no operation, quit

Meaning of 3 Digit Reply Code (1)

- 1yz Positive Preliminary reply
  - not used by SMTP
- 2yz Positive Completion reply
- 3yz Positive Intermediate reply
- 4yz Transient Negative Completion reply
- 5yz Permanent Negative Completion reply

Meaning of 3 Digit Reply Code (2)

- x0z Syntax
- x1z Information
- x2z Connections
- x5z Mail system
- “z” gives a finer gradation of meaning in each of the function

Example of Command/Reply Sequence (1)

```
R: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready
S: HELO USC-ISIF.ARPA
R: 250 BBN-UNIX.ARPA

S: MAIL FROM:<Smith@USC-ISIF.ARPA>
R: 250 OK

S: RCPT TO:<Jones@BBN-UNIX.ARPA>
R: 250 OK

S: RCPT TO:<Green@BBN-UNIX.ARPA>
R: 550 No such user here
```

Example of Command/Reply Sequence (2)

```
S: RCPT TO:<Brown@BBN-UNIX.ARPA>
R: 250 OK

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: .
R: 250 OK

S: QUIT
R: 221 BBN-UNIX.ARPA Service closing transmission channel
```

POP and IMAP

- Post Office Protocol (rfc1939)
- Internet Message Access Protocol (rfc2060)
- protocol to receive mails from (final) mail server
  - POP/IMAP clients are not persisitently connected to the Internet

MIME (Multipurpose Internet Mail Extensions, rfc2045~2049)

- extension (complication) to rfc822
  - body in non-ASCII characters
  - body other than text
  - multiple bodies (multipart)
  - header with non-ASCII characters
- widely deployed, though unnecessary

Non-ASCII characters

- tagging by “charset” in “Content-type:”
  - “charset=ISO-2022-JP
  - can not mix multiple charsets in body
    - possible with “multipart/mixed”?
      - depends on implimentations
- not necessary as ISO 2022 is enough
  - was already actually so in Japan when MIME was developed

8bit Transparency

- special string in header
  - =?<u>CHARSET</u>?[BQ]?<u>TEXT</u>?=
- treatment in body is specified
  - Content-Transfer-Encoding header
- Quated Printable Encoding
  - if mostly ASCII
- Base 65 Encoding
  - represent 2*8 bits by three ASCII characters (+, /, 0-9, A-Z, a-z)

8 bit Transparency was not Necessary

- 7 bit is enough for ISO 2022 text
  - was already actually so in Japan when MIME was developed
- binaries was encoded with UUENCODE
  - in EBCDIC environment
    - transparent to BASE 64 characters
    - characters used by UUENCODE may be modified?

Body other than Text

- tagging by Content-type: header
  - text, image, audio, video, application, (multipart), message
    - finer tagging by subtype (e. g. “text/plain”)
- in practice
  - only “application/octet-stream” is used
  - file name extension (e. g. “.jpg”) specify type
    - UUENCODE is enough

ESMTP (rfc1651)

- Extended SMTP
- developed with MIME
- various negotiations possible
  - primarily for 8bit transparency (rfc1652)

ISO-2022-KR Charset (rfc1557)

- 7bit character code to encode hangul and kanji by Korean character set
- G0 for ASCII, G1 for KS C 5601
  - switched by SI/SO
- in each line containing SI, escape sequence to specify KS C 5601 to G1 is given
- the same rfc also specify EUC-KR charset

ISO-2022-JP-2 Charset (rfc1554)

- extension to ISO-20220JP
  - KS C 5601 (Korean), GB2312 (China), ISO 8859/1 (Western Europe), ISO 8859/7 (Greek) are added
- 94 character set is G0, 96 character set is used as G2 (SS2(ESC+“N”))
- purely 7 bit, though byte value of 127 is used

Wrap Up

- unification is quantization error
  - UNICODE confuses quantization and output error and is unusable for I18N
- argument for I18N is full of misdirections
  - internationalized domain name, language tag
- e-mail format is specified by rfc822
  - MIME extension was not necessary
- e-mail transport is by rfc821
  - ESMTP extension was not necessary

Confusions in Proper Interpretations on RFC821 (SMTP) for Domain Names

太田昌孝

東京工業大学情報理工学研究科

mohta@necom830.hpcl.titech.ac.jp

An e-mail from Tony Finch (March 2014)

https://www.ietf.org/mail-archive/web/dnsop/current/msg11925.html

- CNAME pointing at MX is a different problem, which does not work consistently in practice. The requirement in RFC 1123 is a restatement of RFC 821 section 3.7 (last paragraph) and page 30 (penultimte paragraph).

Specification of RFC821 (no alias is allowed in domain name)

- Whenever domain names are used in SMTP only the official names are used, the use of nicknames or aliases is not allowed.
- Hosts are generally known by names which are translated to addresses in each host. Note that the name elements of domains are the official names -- no use of nicknames or aliases is allowed.

RFC1123 to Clarify(?) RFC821

The domain names that a Sender-SMTP sends in MAIL and RCPT commands MUST have been "canonicalized," i.e., they must be fully-qualified principal names or domain literals, not nicknames or domain abbreviations. A canonicalized name either identifies a host directly or is an MX name; it cannot be a CNAME.

The sender-SMTP MUST ensure that the <domain> parameter in a HELO command is a valid principal host domain name for the client host.

Actual Requirement (RFC6409)

- Nonetheless, unconditionally resolving aliases could be harmful. For example, if www.example.net and ftp.example.net are both aliases for mail.example.net, rewriting them could lose useful information.

```
www.example.net  CNAME mail.example.net
ftp.example.net      CNAME mail.example.net
mail.example.net    MX 0 mx.example.net
```

Why Aliasing Harmful?

- can cause loop with old fragile implementations

```
cname.example.com CNAME mx.example.com
 mail.example.com MX 0 cname.example.com
                  MX 1 other.example.com
                  MX 2 mx.example.com
```

- alias is used at the right side of MX
- how about left side?
  - not harmful
  - why forbidden by rfc821 and 1123?

History of Domain Name and Host Name

- was sharing a file “hosts.txt” maintained by ISI
  - – to translate hostname and IP address
- as the Internet grows, DNS was introduced as loosely couped distributed DB
  - – basic specification is by RFC1034,1035
    - has additional functionality in addition to hostname and IP address translations
      - – there are various attempts to specify mail servers of a mail domain (MB, MD, MF, MG, MINFO, MR, MX)
      - – initially, only translation between hostname and IP address
        - » no mail domain exist

RFC881 first enables translation from mail domain to mail server

- The domain server design also provides for mapping mailbox addresses to the host name of the mail server for that mailbox. This feature allows mailboxes to be related to an organization rather than to a specific host.
- no similar specification in RFC819 (specified at the same time as RFC821)
  - mail domain and host name was not distinguished
  - mail domain name is the host name of mail server?
    - still interpreted so, if MX is not specified to a doain (rfc974)

Original Intention of RFC821

```
cname.example.com CNAME mail.example.com
mail.example.com    A 192.0.2.1
```

is prohibited

- specification of RFC821 is fine
- as MX was introduced (after rfc821), aliases only at the right side of MX need to be prohibited
- interpretation of rfc821 by rfc1123 is wrong

RFC1123 to Clarify(?) RFC821

~~The domain names that a Sender-SMTP sends in MAIL and RCPT commands MUST have been "canonicalized," i.e., they must be fully-qualified principal names or domain literals, not nicknames or domain abbreviations. A canonicalized name either identifies a host directly or is an MX name; it cannot be a CNAME.~~

The sender-SMTP MUST ensure that the <domain> parameter in a HELO command is a valid principal host domain name for the client host.

Conclusions

- irrational specification of rfc821 and rfc1123 is studied archeologically
- specification of rfc821 does not assume mail-only domain name, as MX was not invented, and use mail domain name as mail server host name
- rfc1123 (issued 1 year 11 months after rfc1123 specifying MMX) misinterpreted rfc821
  - DNS was already so common
    - wrongly thought MX was available when rfc821 was issued
  - as a result of rapid development/spreading of the Internet
    - there may exist similar misinterpretations/confusions