Dynamic Adaptation in an Image Transcoding Proxy For Mobile Web Browsing

Pravin Bhagwat, Richard Han, Richard LaMaire, Todd Mummert, Veronique Perret, Jim Rubas
Mobile Networking Group

Chung-Sheng Li, Rakesh Mohan, John R. Smith
Image Information Systems Group

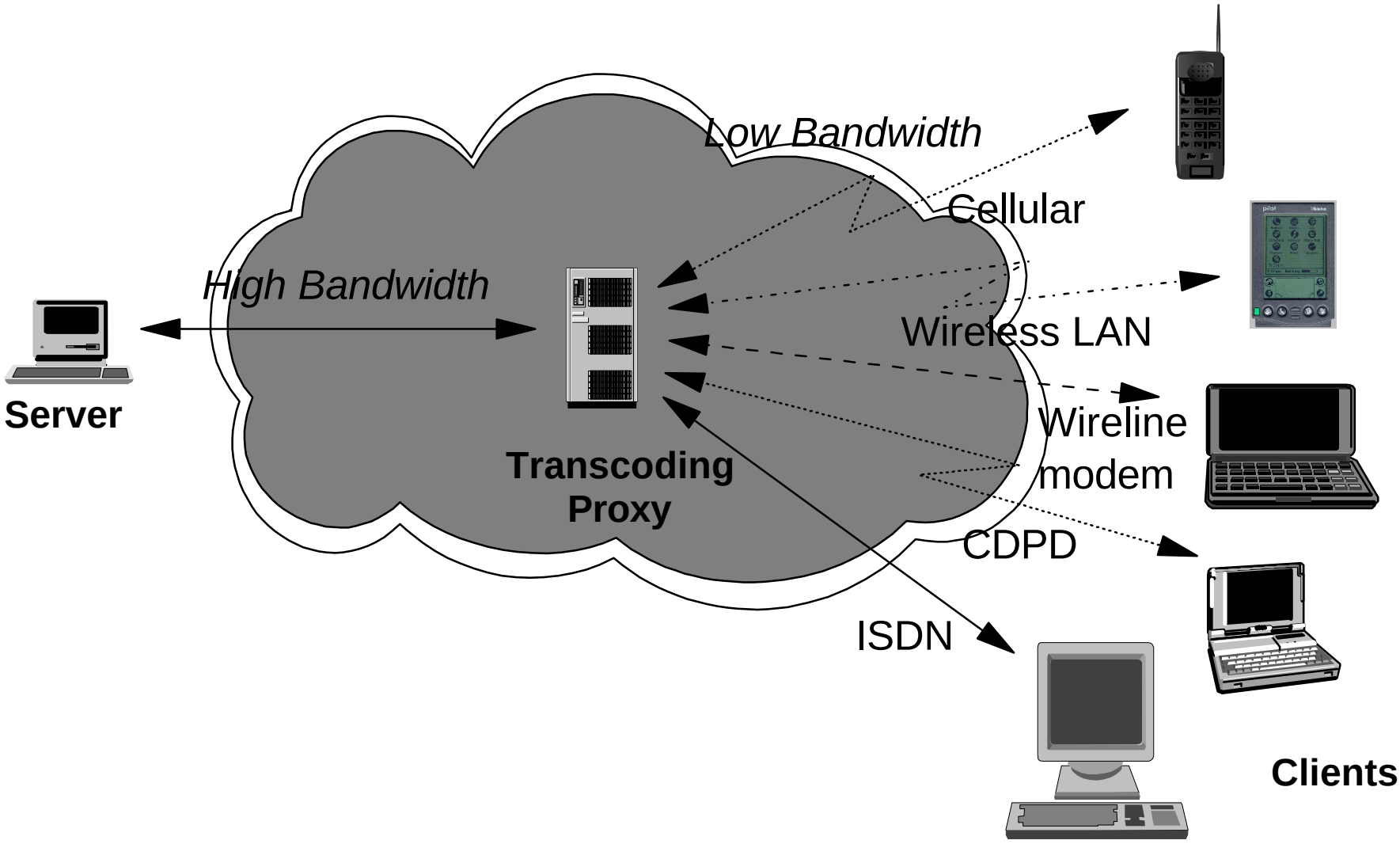
Contact: Rick Han (rhan@watson.ibm.com)

IBM T.J. Watson Research Center

Outline

- Architecture
- Analysis: when and when not to transcode
  - ► Case I: Store-and-forward
    - Image size prediction
    - Image delay prediction
    - Bandwidth estimation
  - ► Case II: Streaming
- Practical rules
  - ► GIF/JPEG
- Other transcoding proxies
- Summary

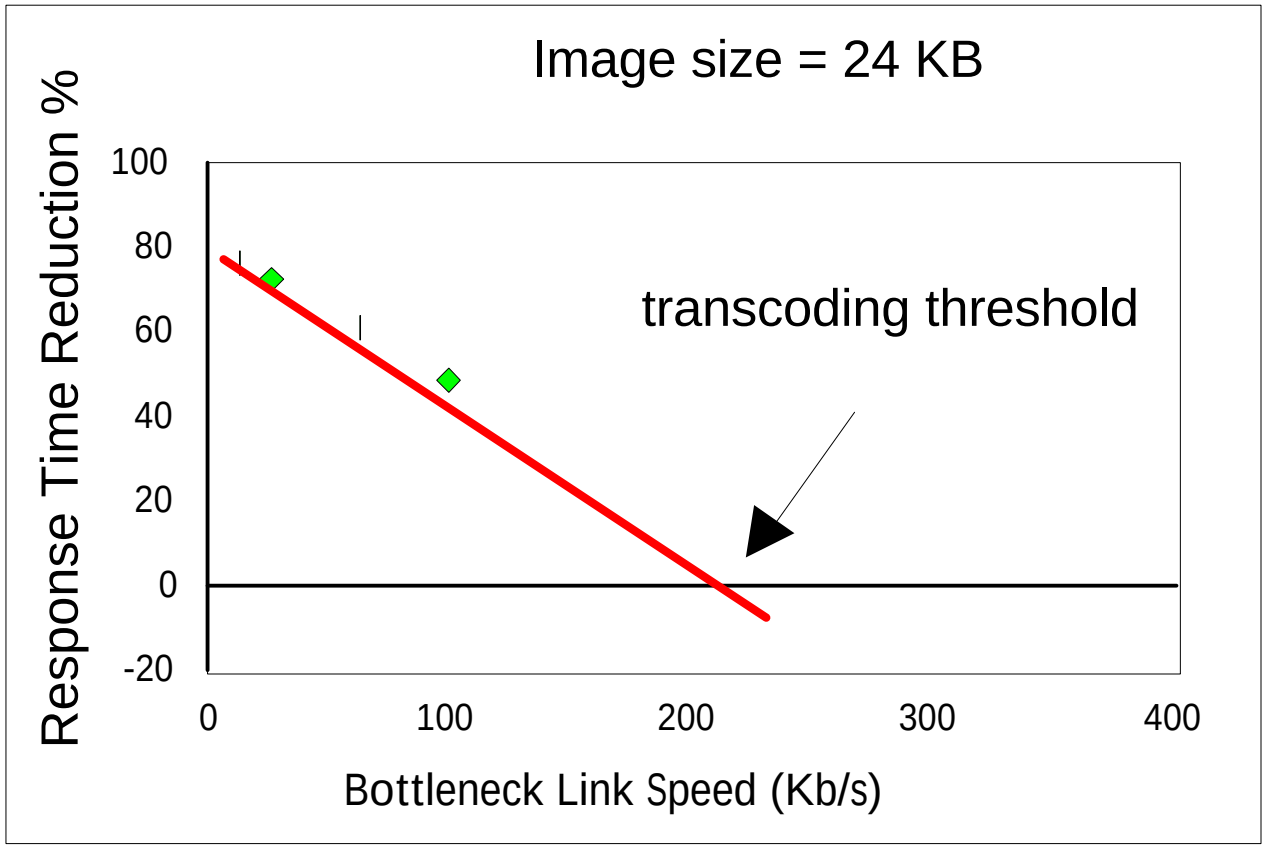
Transcoding Proxy Environment



Reasons to Transcode

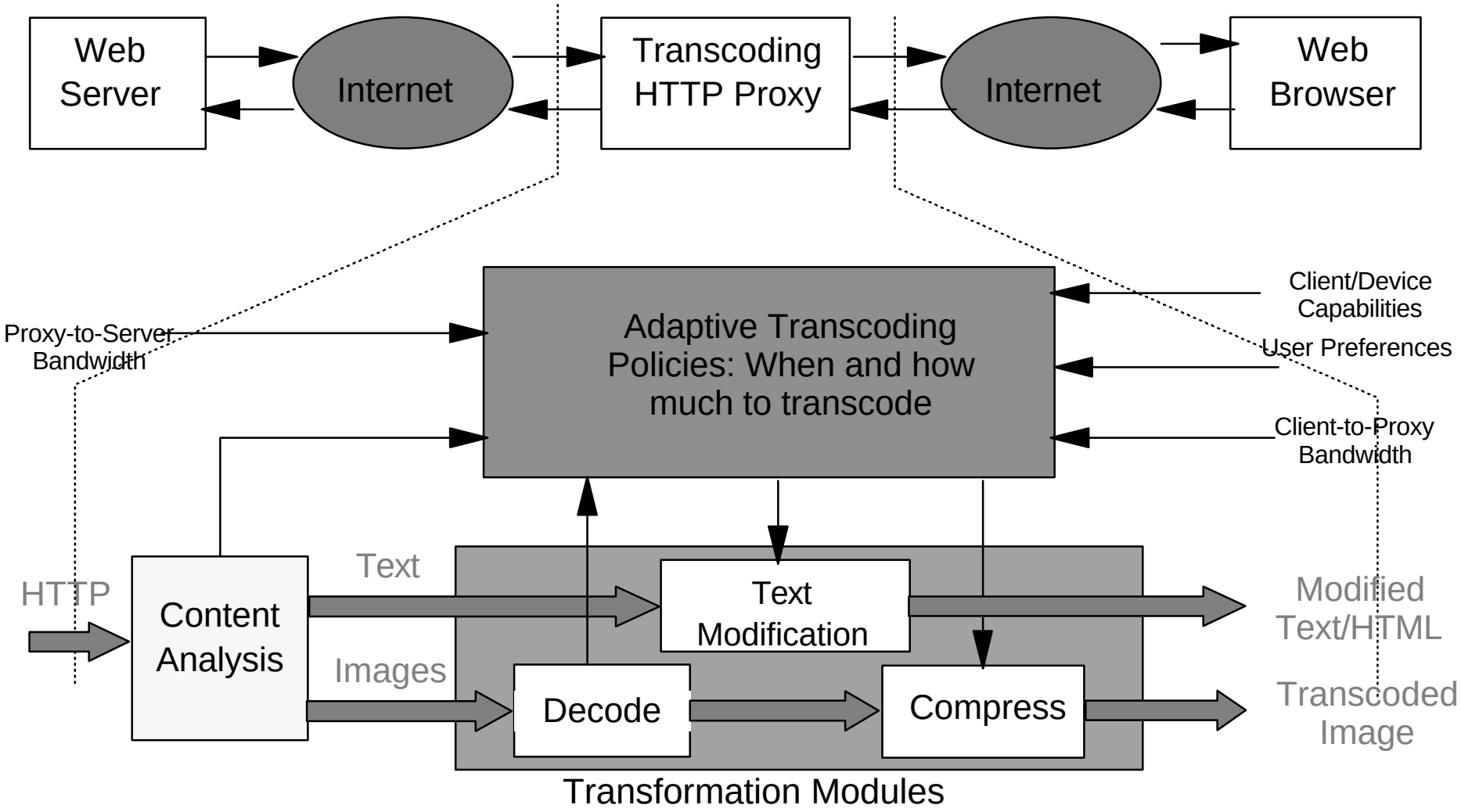
- A transcoder
  1. converts formats => tailoring of data to multiple devices (e.g. Palm PDA)

  2. permits compression
     - Reduced response time via compression over low-bandwidth links.
     - Cost reduction via compression over tariffed links.

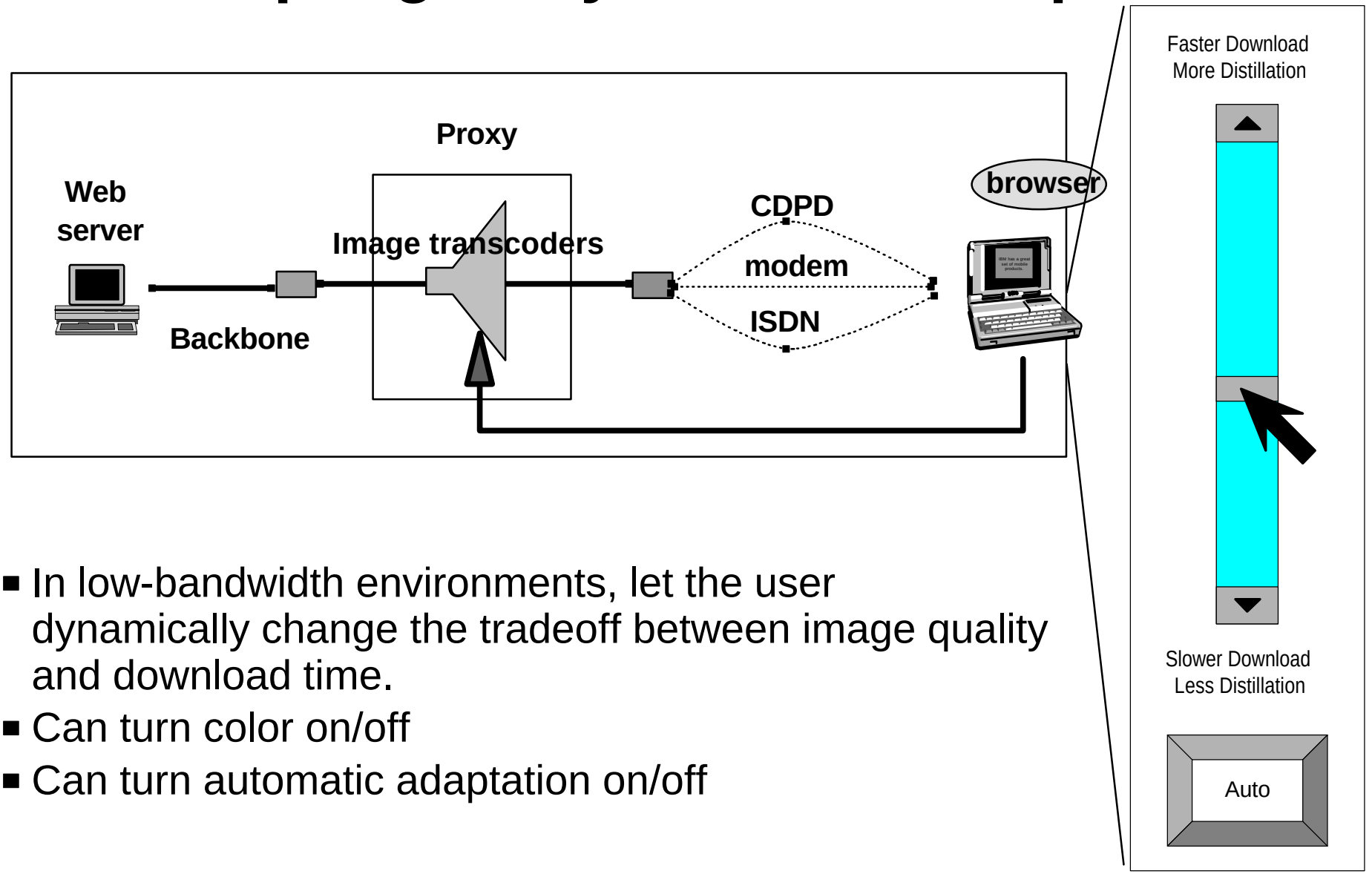


Example parameters: "AnyImage" library, 200 MHz Pentium Pro, Windows NT, JPEG quality = 5.

Transcoding Proxy Architecture

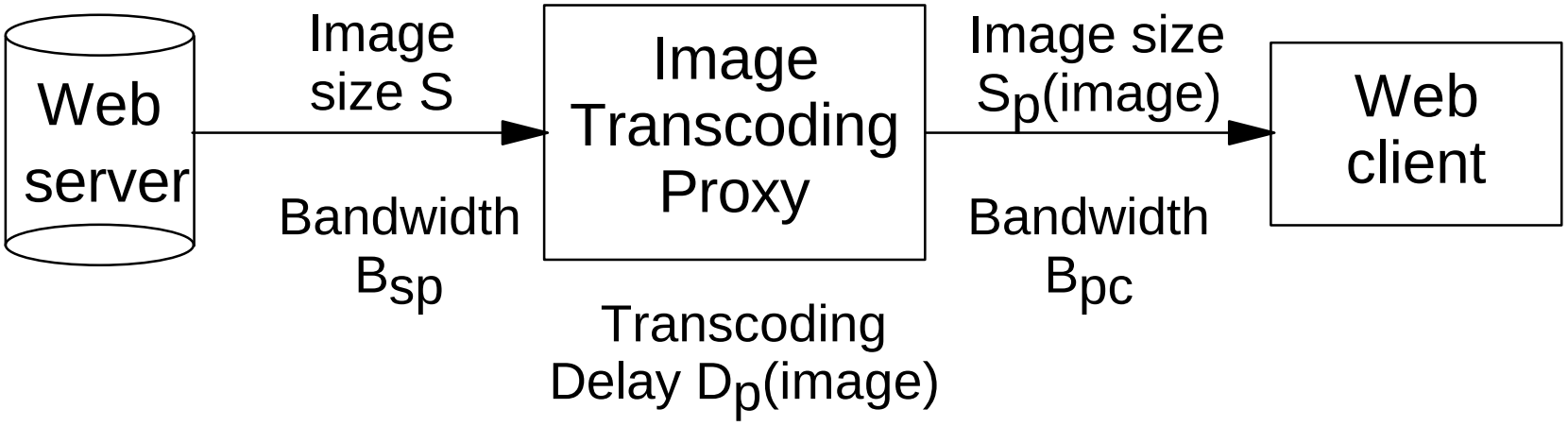


Adapting to Dynamic User Input



- In low-bandwidth environments, let the user dynamically change the tradeoff between image quality and download time.
- Can turn color on/off
- Can turn automatic adaptation on/off

Store-and-Forward Image Transcoding



Definition: A *store-and-forward* image transcoder must wait to transcode until the whole image is received and must wait to transmit until transcoding is completed.

$$t_{no\ proxy} = S/min(B_{pc}, B_{sp}) + D_{prop+queue}$$

$$t_{proxy} = S/B_{sp} + D_p(image) + S_p(image)/B_{pc} + D_{prop+queue}$$

S/F delay

transcoding delay

output transmission delay

To Transcode or Not To Transcode

Only transcode when $t_{proxy} < t_{no\ proxy}$ *:*

$$S/B_{sp} + D_p(image) + S_p(image)/B_{pc} < S/min(B_{pc}, B_{sp})$$

Case I: $B_{sp} < B_{pc}$ (server-proxy link is bottleneck)

=> Never transcode when Internet backbone is the bottleneck!

Case II: $B_{pc} < B_{sp}$ (proxy-client link is bottleneck)

=> Only transcode when

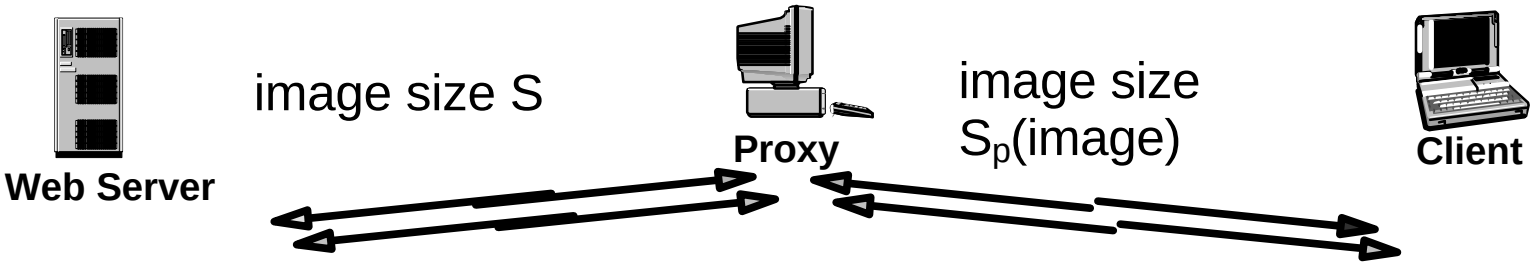
$$D_p(image) + S_p(image)/B_{pc} < S*(1/B_{sp} - 1/B_{pc})$$

Predicting the Output Image Size

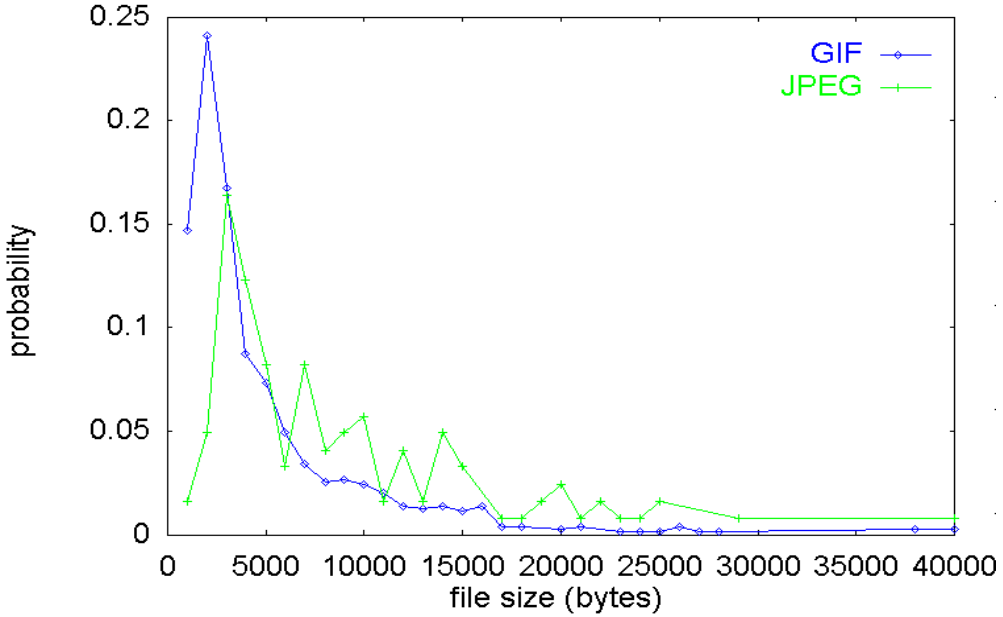
- S_p(image) depends upon image content, image dimensions, image input size, transcoding parameters, compression algorithm

- prediction occurs before transcoding => get info from
  - ► image headers
  - ► probabilistic distributions (image content)

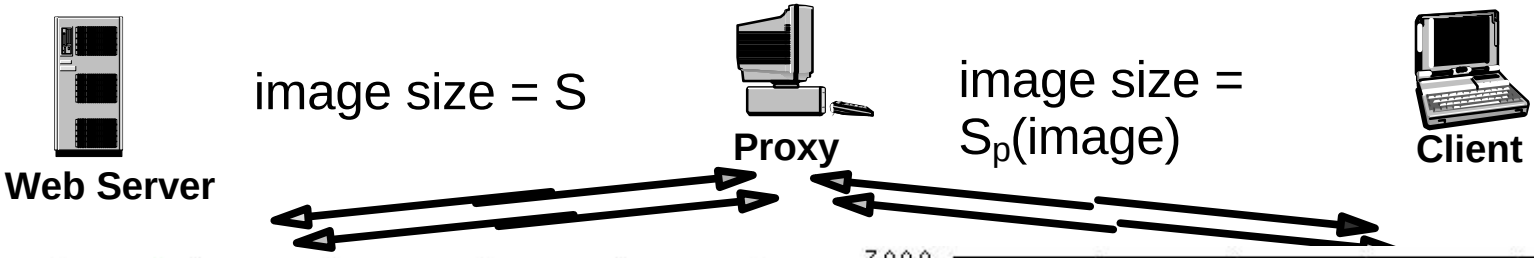
Experimental Description

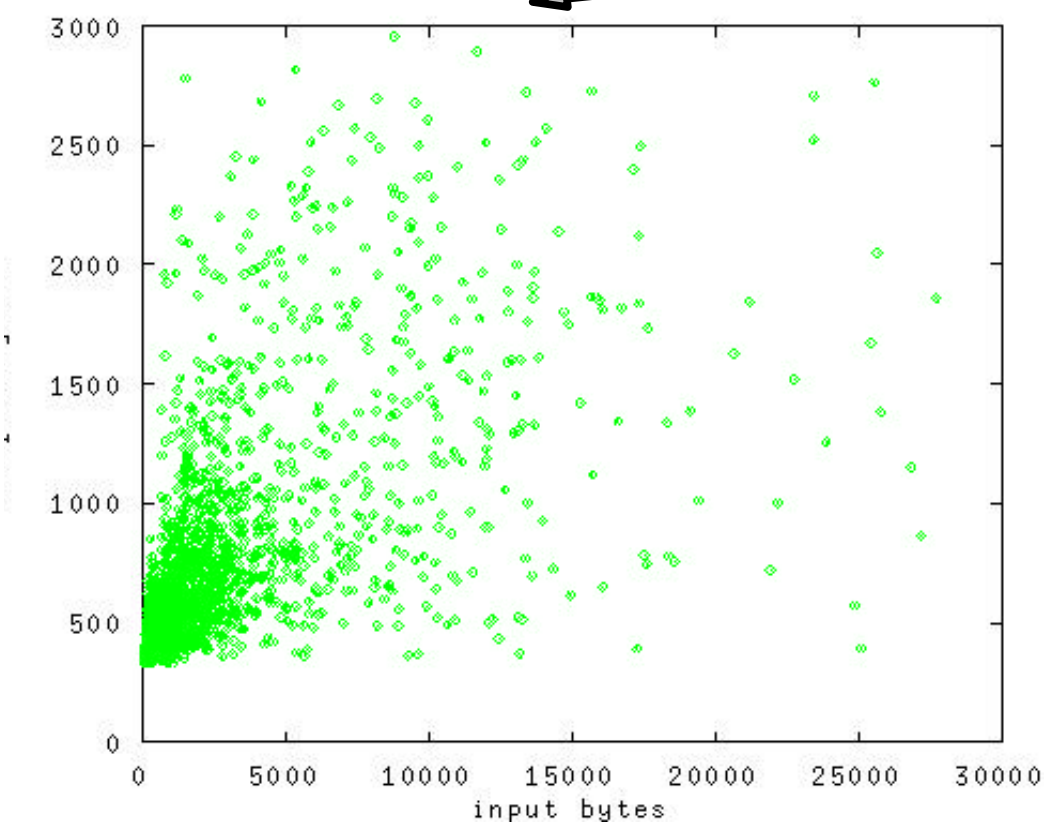


- Sample characteristics:
  - Obtained by visiting top 100 web sites.
  - 1074 GIFs (ave. of 2724 bytes), 123 JPEGs (ave. of 4697 bytes).
- Platform:
  - 200 MHz Pentium Pro proxy running Windows NT.
  - "Anymain" image library.

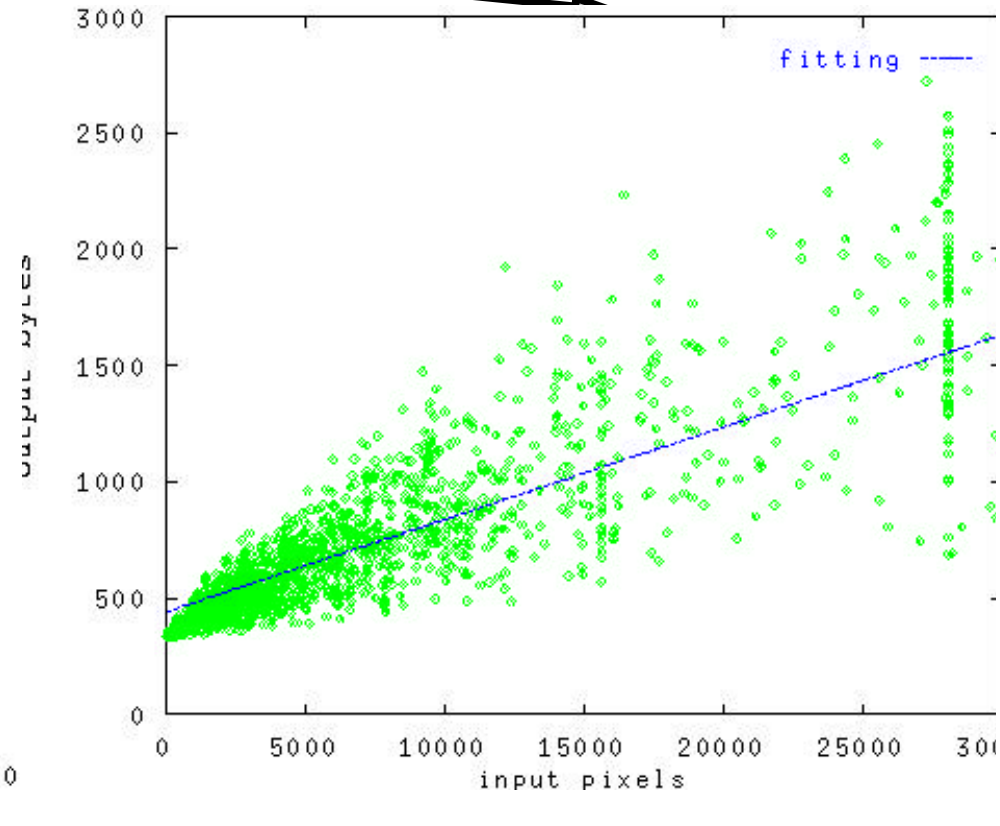


Predicting the Output Image Size



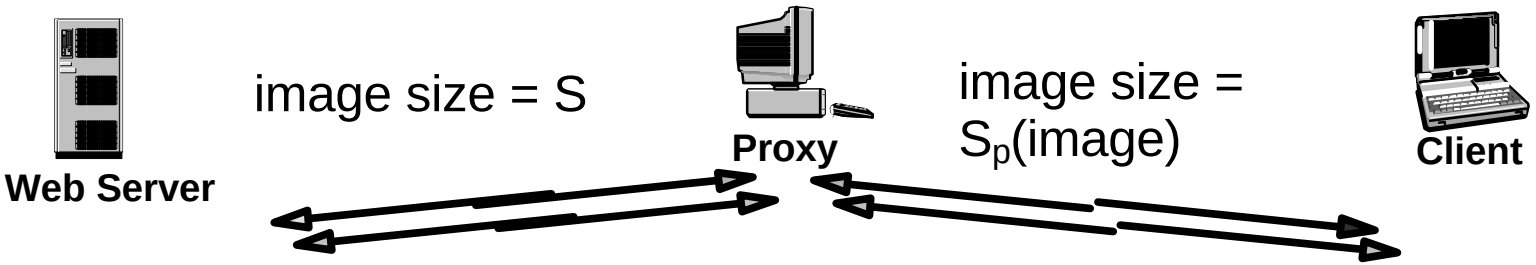


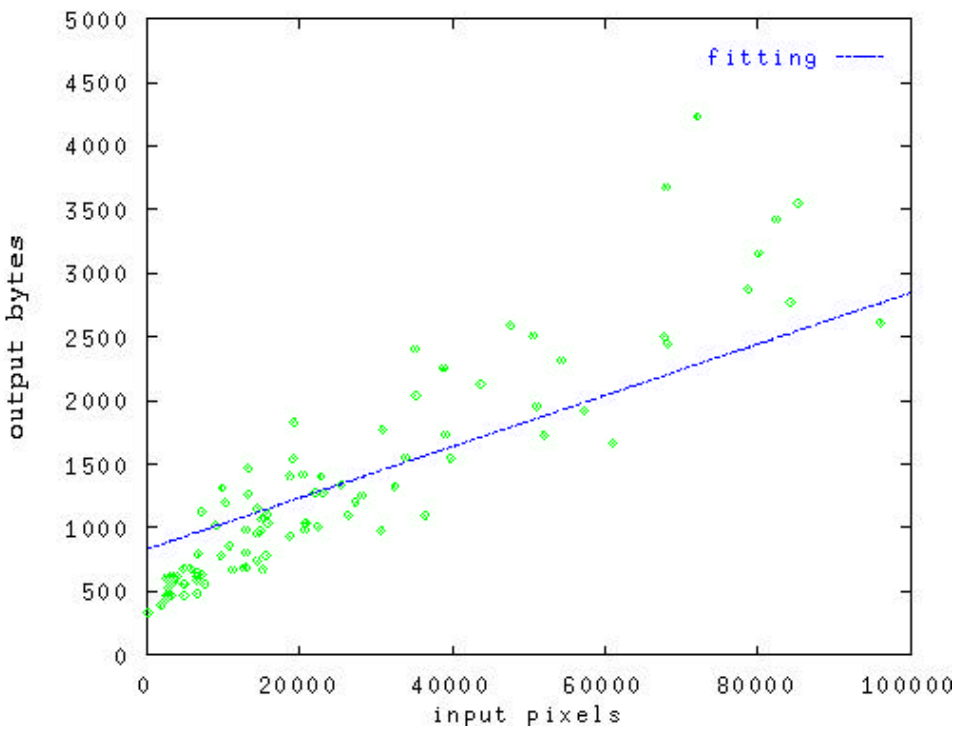
- S_p=f(input bytes), GIF-to-JPEG, q=5, ρ=0.54

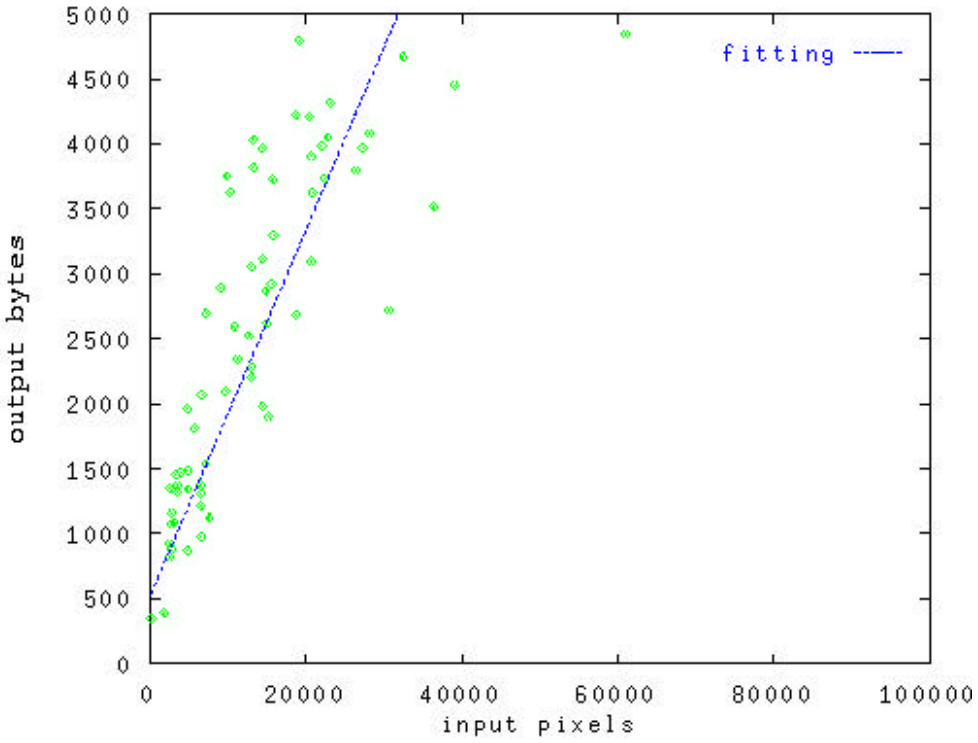


- S_p=f(input pixels), GIF-to-JPEG, q=5, ρ=0.90

Predicting the Output Image Size





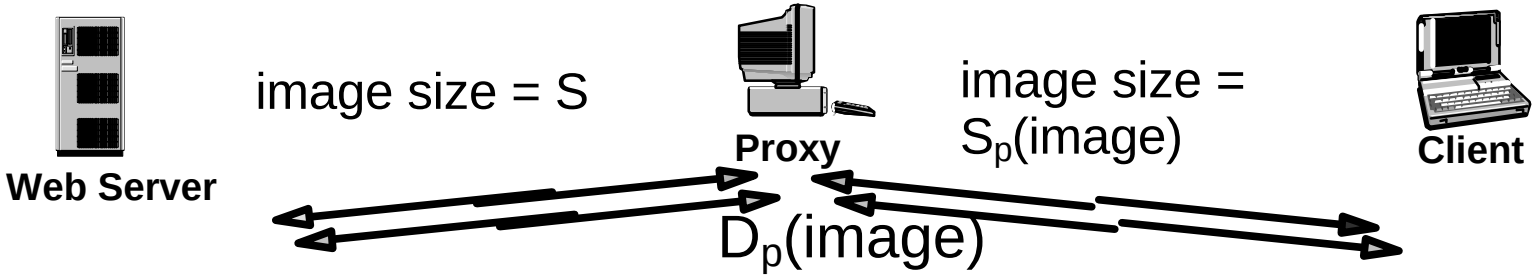


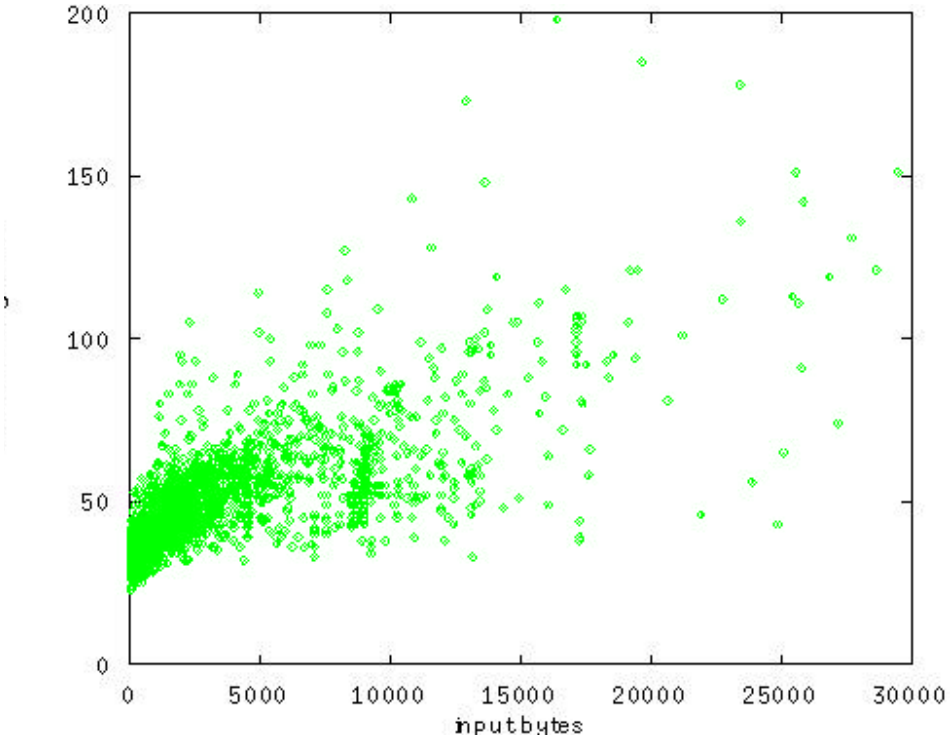
- S_p=f(input pixels), JPEG-to-JPEG, q=5, ρ=0.94
- S_p=f(input pixels), JPEG-to-JPEG, q=50, ρ=0.92

Predicting the Image Transcoding Delay

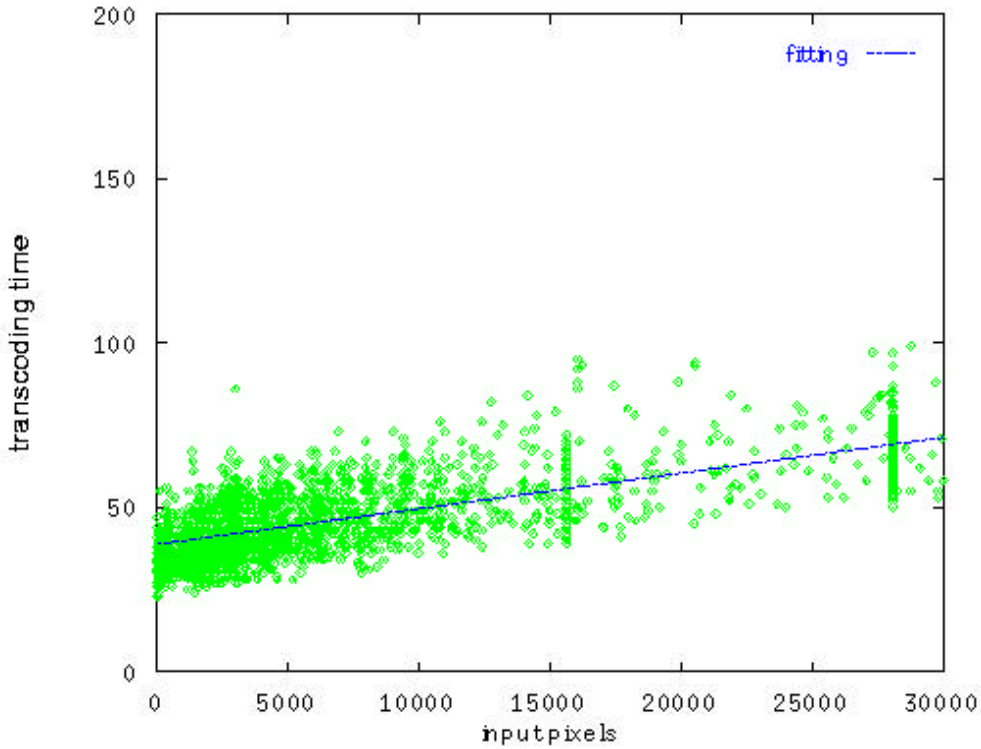
- $D_p(\text{image}) = D_{\text{image transformations}} + D_{CPU}$
- $D_{\text{image transformations}}$ depends on input size, image dimensions, image content, transcoding parameters, decompression & compression algorithms, implementation efficiency of image library
  - use image headers and statistical analysis to predict the image processing delay
- D_{CPU} depends on CPU bandwidth/other processes/threads

Predicting the Image Transcoding Delay



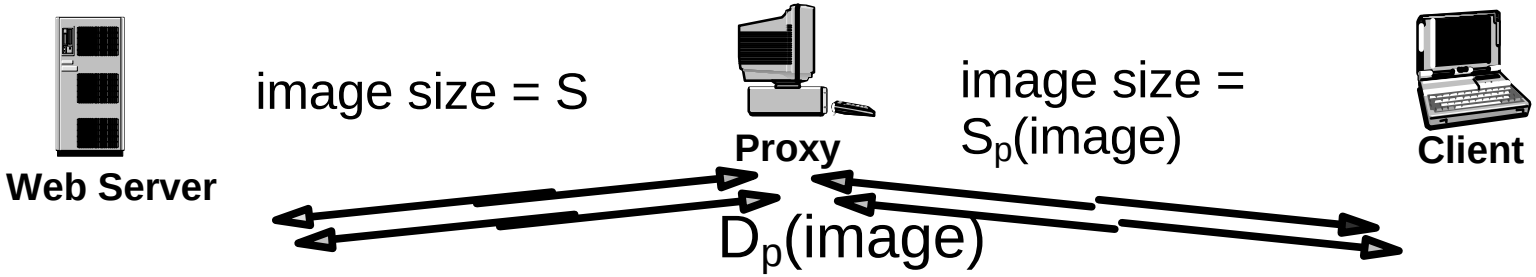


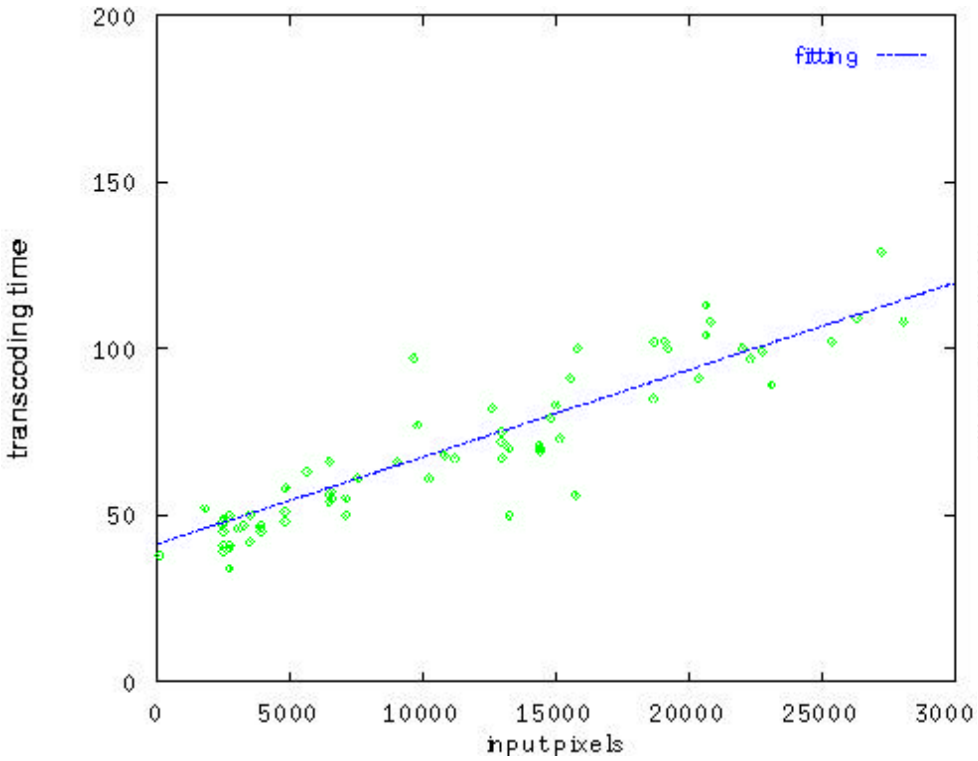
- D_p=f(input bytes), GIF-to-JPEG, q=5, ρ=0.65

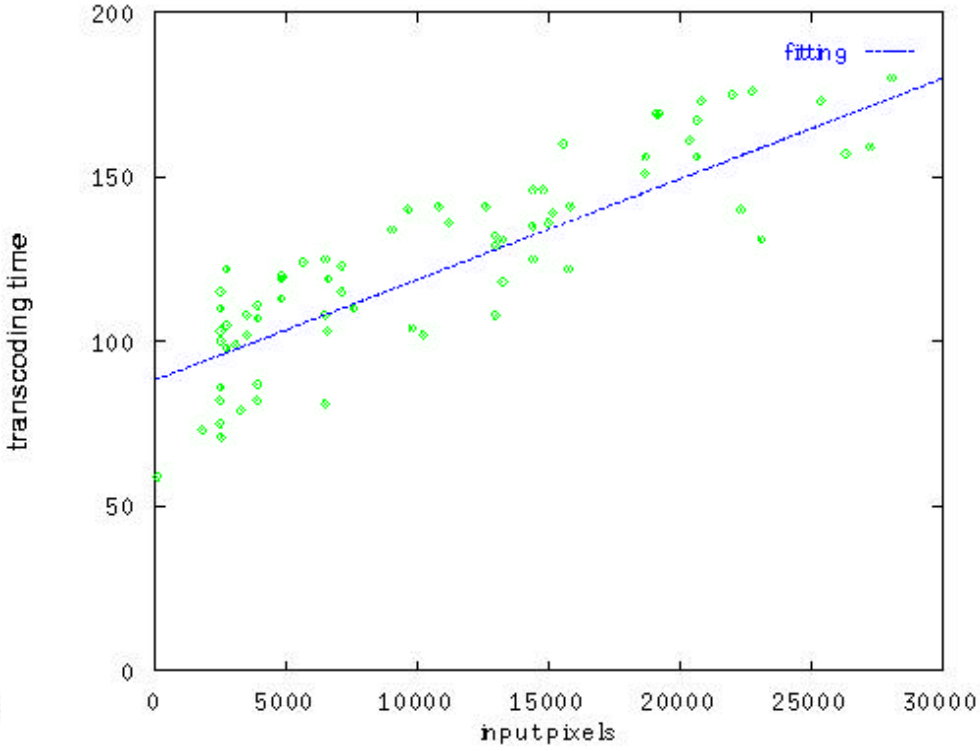


- D_p=f(input pixels), GIF-to-JPEG, q=5, ρ=0.82

Predicting the Image Transcoding Delay

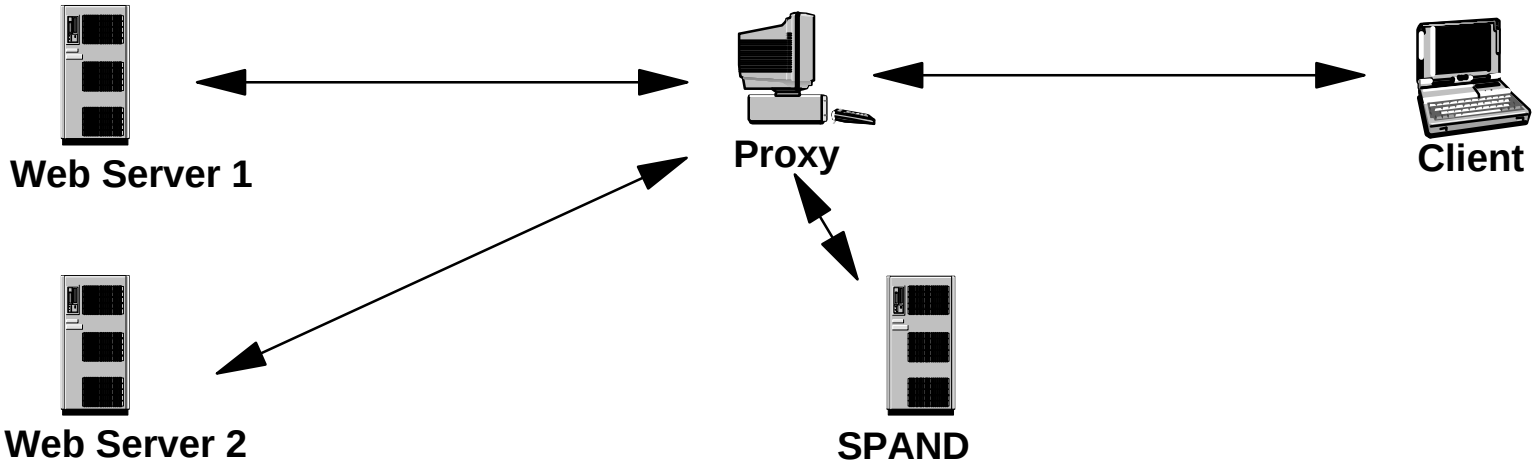






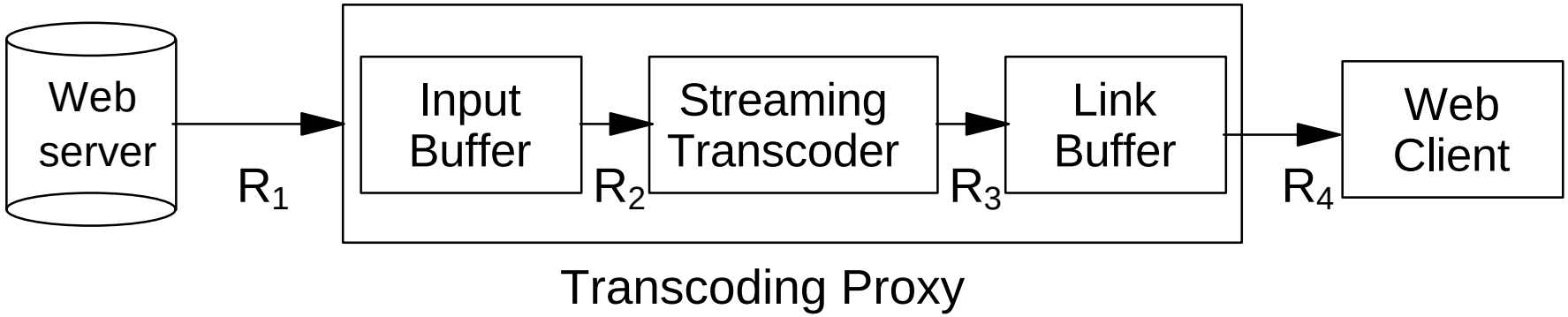
- D_p=f(input pixels), JPEG-to-JPEG, q=5, ρ=0.98
- D_p=f(input pixels), JPEG-to-JPEG, q=50, ρ=0.98

Bandwidth Estimation



- NetDyn: passive network monitoring
- Client-proxy link bandwidth estimation (long-lived)
- Proxy-server link bandwidth estimation (transient)
- Querying for SPAND-like statistics

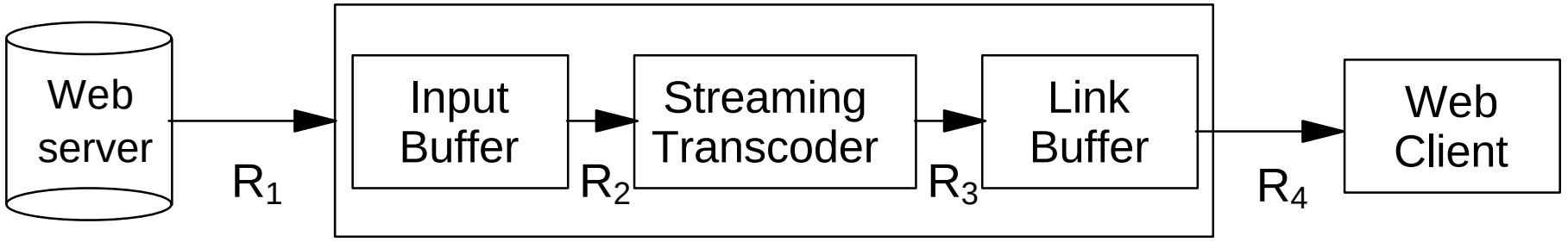
Streamed Image Transcoding



- Definition: A *streamed* image transcoder can begin writing out transcoded image data before having fully read in an image.

  R_1 = image rate into input buffer
  R_2 = image rate at which streaming transcoder empties input buffer
  R_3 = transcoded image rate into link buffer
  R_4 = image transmission rate out of link buffer

To Transcode or Not To Transcode Redux



- Condition I: Don't overflow the input buffer

  $R_2 > R_1$ => $S/D_p(\text{image}) > B_{sp}$

  => $D_p(\text{image}) < S/B_{sp}$

- Condition II: Don't overflow the output link's buffer

  $R_4 > R_3$, and image rate @ R_3 = image rate @ R_1

  => $B_{pc} > S_p(\text{image})/[S/B_{sp}]$

  => $\gamma > B_{sp}/B_{pc}$, compression ratio $\gamma = S/S_p(\text{image})$

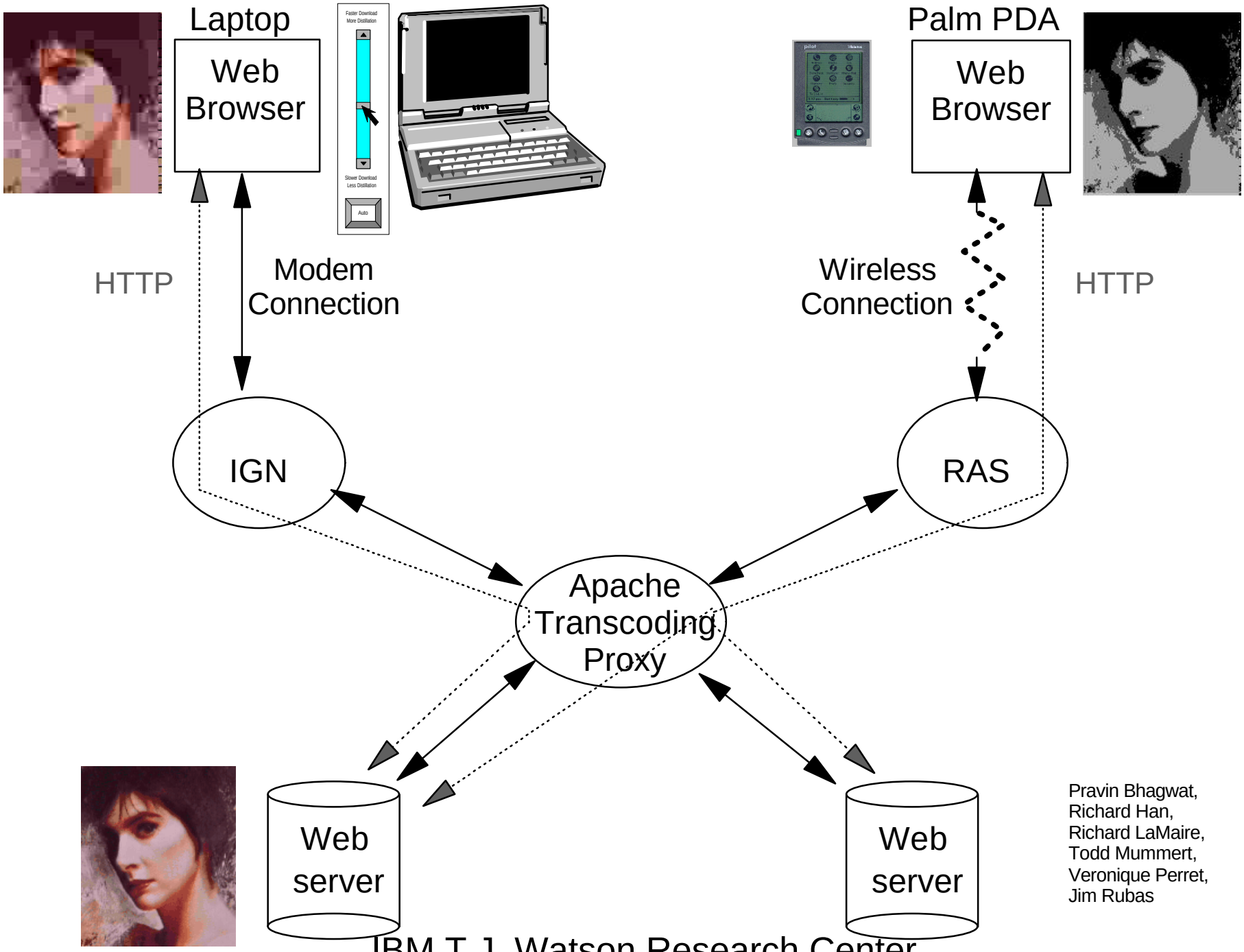
Streamed Image Transcoding (cont.)

- Only transcode when Conditions I and II hold:
  (I) $D_p(\text{image}) < S/B_{sp}$
  (II) $\gamma > B_{sp}/B_{pc}$

- If $B_{sp} < B_{pc}$ (server-proxy link is bottleneck)
  => Condition II always satisfied. Only have to test Condition I.

- If $B_{pc} < B_{sp}$ (proxy-client link is bottleneck)
  => Both Conditions I and II must be evaluated.

Reality: Transcoding Proxy Implementation



Pravin Bhagwat,
Richard Han,
Richard LaMaire,
Todd Mummert,
Veronique Perret,
Jim Rubas

Current Store-and-Forward Transcoding Policies

```
if to laptop
    if (input size>1000 bytes)
        if input is GIF
            if well-compressed GIF
                GIF->GIF as f(user pref.)
            else
                GIF->JPEG as f(user pref.)
        else
            JPEG->JPEG as f(user pref.)
    if (output > input size)
        send original image
    else
        send transcoded image
else /*to Palm*/
    GIF/JPEG->Palm as f(user pref.)
```

Properties

* adapts to user preferences and client device

* adapts to network bandwidth in a static sense

* currently does not predict image delay or output size

Why GIF Is Well-Suited to Compressing "Graphics"

- GIF based on LZW:
  - Build dictionary by scanning pixel rows
  - add a new word = matched word + 1 character/pixel
  - send offsets into dictionary

- When there are few colors, you get long runs of the same color/pattern, and GIF compresses well.

- When there are many colors, there are few long patterns, so GIF's don't compress as well.

Why JPEG Is Well-Suited to Compressing a Natural Image

- JPEG has 3 stages:

  DCT -> quantization -> lossless (RL + Huffman or arithmetic)

  DCT concentrates info in low frequency coefficients, so quantization tends to remove high-frequency coefficients

- Natural images often consist of mostly low-frequency/variation, or unimportant high-frequency background, so quantization doesn't hurt quality, and achieves high compression

- Graphical images have much high-frequency information, so JPEG hurts their reconstructed quality.

JPEG Image Quality Reduction Example

Original: 20,796 bytes

Quality factor = 5: 2525 bytes
12.1% of original

Quality factor = 35: 6,993 bytes
33.6% of original

Grayscale and
Quality factor = 5: 1,886 bytes
9.1% of original

Rationale for Transcoding Policies

```
if to laptop
    if (input size>1000 bytes)
        if input is GIF
            if well-compressed GIF
                GIF->GIF as f(user pref.)
            else
                GIF->JPEG as f(user pref.)
        else
            JPEG->JPEG as f(user pref.)
    if (output > input size)
        send original image
    else
        send transcoded image
else /*to Palm*/
    GIF/JPEG->Palm as f(user pref.)
```

- Static evaluation of store-and-forward inequality:

  $S/B_{sp} + D_p(image) + S_p(image)/B_{pc} < S/min(B_{pc}, B_{sp})$

- fix B_{sp}=1 Mb/s, B_{pc}=50 kb/s, D_p=40 ms+Δ, S_p=αS
  => S>(263+Δ')/(1-α)
  => ~800 byte lower limit
  (α=0.5)

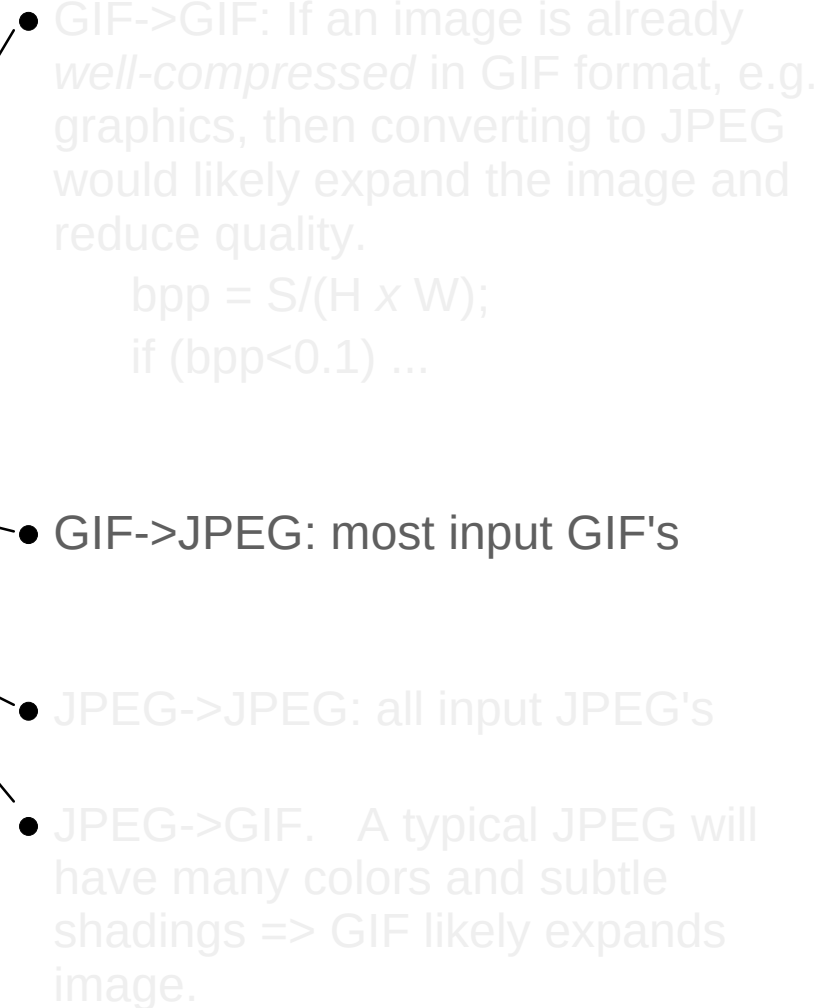
Rationale for Transcoding Policies

```
if to laptop
    if (input size>1000 bytes)
        if input is GIF
            if well-compressed GIF
                GIF->GIF as f(user pref.)
            else
                GIF->JPEG as f(user pref.)
        else
            JPEG->JPEG as f(user pref.)
    if (output > input size)
        send original image
    else
        send transcoded image
else /*to Palm*/
    GIF/JPEG->Palm as f(user pref.)
```

- GIF->GIF: If an image is already *well-compressed* in GIF format, e.g. graphics, then converting to JPEG would likely expand the image and reduce quality.
  bpp = S/(H *x* W);
  if (bpp<0.1) ...
- GIF->JPEG: most input GIF's
- JPEG->JPEG: all input JPEG's
- JPEG->GIF. A typical JPEG will have many colors and subtle shadings => GIF likely expands image.

Rationale for Transcoding Policies

```
if to laptop
    if (input size>1000 bytes)
        if input is GIF
            if well-compressed GIF
                GIF->GIF as f(user pref.)
            else
                GIF->JPEG as f(user pref.)
        else
            JPEG->JPEG as f(user pref.)
    if (output > input size)
        send original image
    else
        send transcoded image
else /*to Palm*/
    GIF/JPEG->Palm as f(user pref.)
```



Rationale for Transcoding Policies

```
if to laptop
    if (input size>1000 bytes)
        if input is GIF
            if well-compressed GIF
                GIF->GIF as f(user pref.)
            else
                GIF->JPEG as f(user pref.)
        else
            JPEG->JPEG as f(user pref.)
    if (output > input size)
        send original image
    else
        send transcoded image
else /*to Palm*/
    GIF/JPEG->Palm as f(user pref.)
```

- GIF->GIF: If an image is already *well-compressed* in GIF format, e.g. graphics, then converting to JPEG would likely expand the image and reduce quality.
  bpp = *S*/(H *x* W);
  if (bpp<0.1) ...
- GIF->JPEG: most input GIF's
- JPEG->JPEG: all input JPEG's
- JPEG->GIF. A typical JPEG will have many colors and subtle shadings => GIF likely expands image unless we can find fast colormap reduction.

Other Transcoding Proxies

- UCB/ProxiNet - real-time transcoding proxy for small clients (Pythia supported PC's/laptops)
- Spyglass' Prism - real-time transcoding proxy for small clients (text and images)
- AvantGo 2.0 - hot-synch proxy for small devices, possibly off-line transcoding
- Intel's QuickWeb - defunct, real-time transcoding proxy for PC's/laptops, light transcoding
- 3COM's "Web clipping" proxy for Palm VII (no images?)

Other Research Issues

- Integrating caching proxies with transcoding
- Split browser paradigm
- Scalability/Load-Balancing
- Security

Summary

- Derived theoretical conditions determining when proxies should adaptively transcode images
  - Store-and-forward
  - Streaming
- Presented a set of practical transcoding policies that are statically adaptive
- Paper: "*Dynamic Adaptation In An Image Transcoding Proxy For Mobile Web Browsing*", December 1998, *IEEE Personal Communications Magazine*. Contact: rhan@watson.ibm.com.