

A General Purpose Proxy Filtering Mechanism Applied to the Mobile Environment

Authors: Bruce Zenel and Dan Duchamp

Presented by Eric Lin

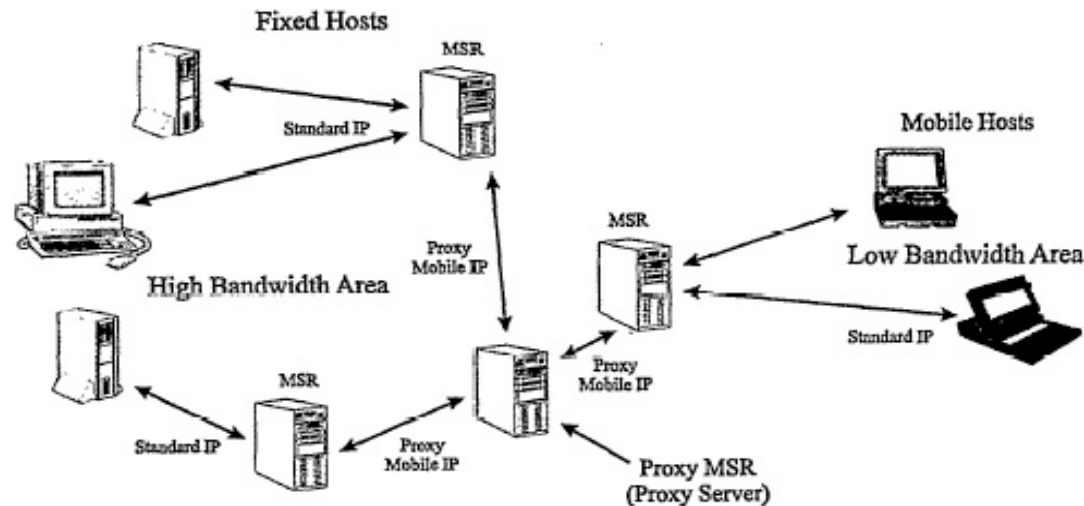


The Plan

- # Architecture: MICP and Proxy MSR
 - # Proxy Server
 - # Filters (a.k.a “applets”)
 - # Results from experiment
 - # Questions
-

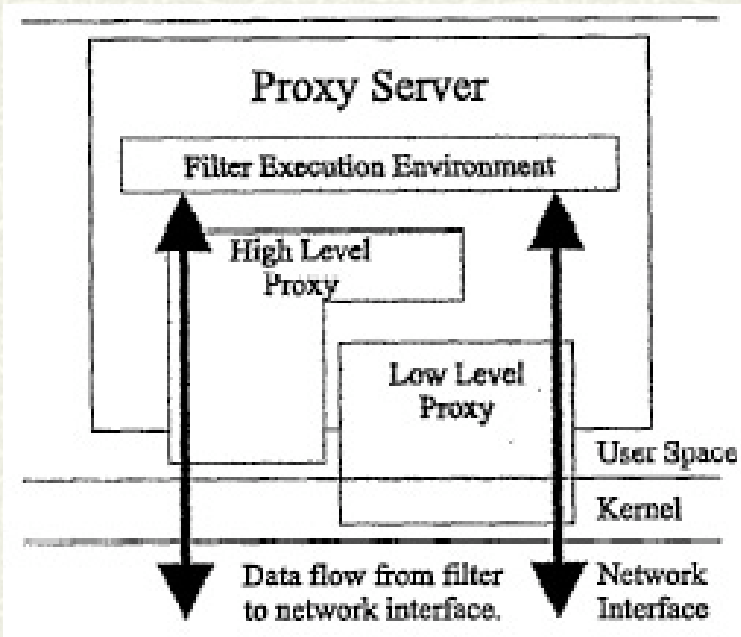
MICP and Proxy MSR

- Uses Columbia Mobile IP (MICP) – as in Bakre and Badrinath paper on I-TCP
 - MSRs route to a mobile host's local MSR
- Proxy MSR (PMSR)
 - Mobile host chooses a MSR to be its PMSR
 - Now, other MSRs route to the PMSR via Proxy MICP (PMICP)
 - PMSR can modify data before forwarding to local MSR (or if already there, to mobile host)



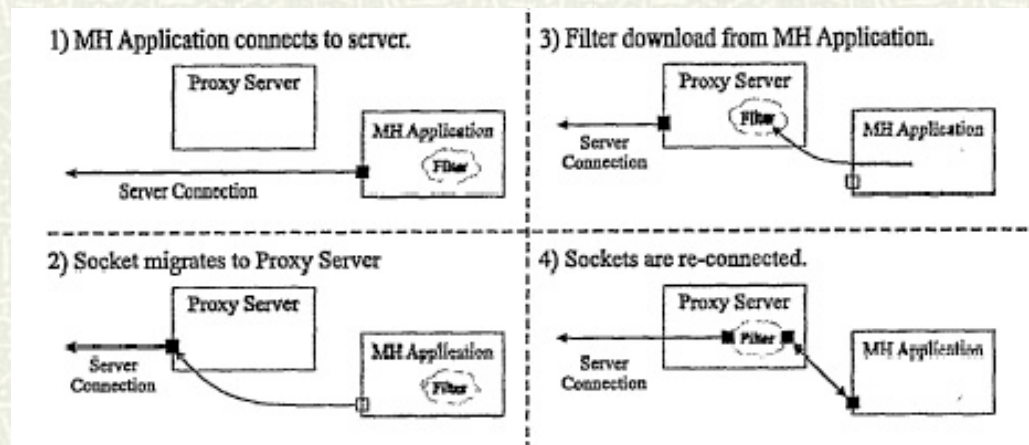
Proxy Server

- PMSR runs one or more Proxy Server (PS) processes
 - Each Proxy Server process – single execution environment for one or more general-purpose filter programs, or applets
 - Filters installed beforehand or dynamically
 - Fetched from mobile host (or nearby repository)



- 2 types of filtering:
 - High Level – for applications that use standard OS socket API
 - Low Level – requires kernel control, below socket layer

Filters



■ Filter insertion

- Uses techniques from Bakre and Badrinath's I-TCP paper: managing socket migration
- Mobile host requests new Proxy Server or just new filter in an existing Proxy Server
- PMSR kernel allows filters to bind to old socket from the mobile host
- Filters send data to mobile host on new connection

Filters (cont.)

■ Dynamic filter control

- Directly by application on mobile host
- Indirectly by Proxy Server
- Based on recent info, or events:
 - Current estimates of bandwidth
 - Network interface info (max. bandwidth)
 - Remaining battery power on MH
 - MH location (cell hand-offs)

■ Event Registry

- Relays events to mobile host or Proxy Server
-

Experiment

- Implemented on physical testbed
 - Tested 3 filters:
 - High Level – HTTP and NFS filters
 - Using 2 kinds of lossless compression
 - Low Level – TCP filter (Balakrishnan's snoop module)
 - Tested 3 interfaces:
 - 10Mbps Ethernet
 - 2Mbps WaveLAN
 - 33.3Kbps SLIP
-

Results

- Most results were not surprising:
 - Taking time to compress and decompress data increases overall latency on high bandwidth links (Ethernet)
 - Savings in total size of transmitted data lowered latency on slow links (SLIP)
 - TCP snoop filter works well with high packet loss
 - NFS filter performs well on WaveLAN, whereas HTTP does not
 - Fetching same text files: Why different?
 - Pipelines compression (per UDP packet) – decreases compression ratio
 - NFS uses a persistent filter (what does this mean exactly?)
 - HTTP filters inserted per transaction
-

Questions

- What exactly is filter overhead?
 - See latency of filter and baseline on SLIP compared to WaveLAN
 - 300 ms overhead on SLIP, but negligible on WaveLAN...
- Should we care about SLIP??
- Need for High Level Proxy?
 - How often do filter functions change per application? Per connection?
- Need for Low Level Proxy?
- Deployment issues?

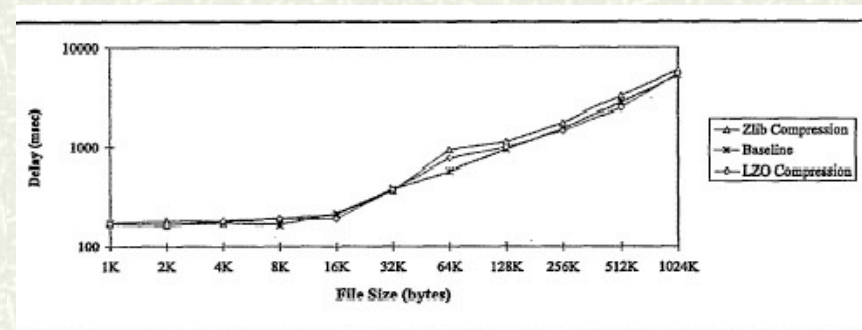


Figure 9: Wavelan testbed: Comparison of file transfer time using HTTP.

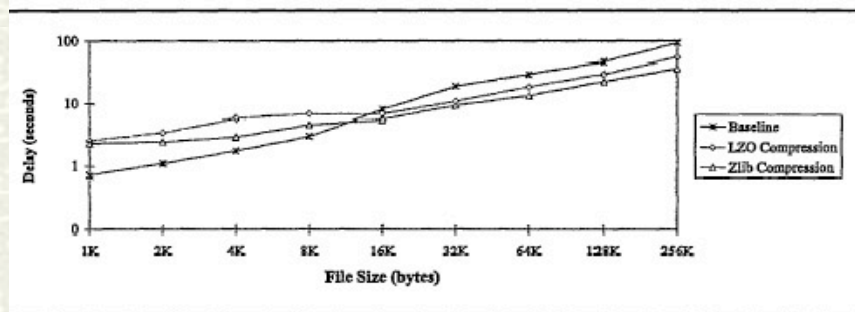


Figure 10: SLIP testbed: Comparison of file transfer time using HTTP.