

Adaptive Mobile Applications



Thomas Kunz

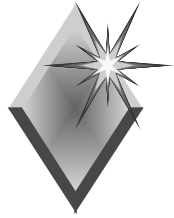
tkunz@uwaterloo.ca

<http://ccnga.uwaterloo.ca/~tkunz/>

University of Waterloo

August 28, 1997

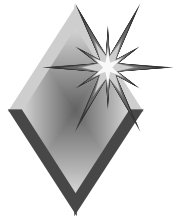




Presentation Outline

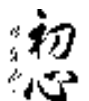
- Motivation
- QoS API
- Testbed
- Example 1: HiFi Sound over Wireless
- Example 2: Groupware Editor
- Example 3: M-Mail
- Conclusions

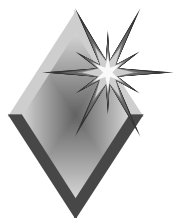




Motivation

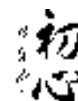
- ◆ Designing applications that perform well in a mobile environment difficult and different from “traditional” distributed application design:
 - ◆ Wireless communication is substantially different from wired communication:
 - ◆ **frequent spurious disconnections** (handoff, shadowed areas)
 - ◆ **lower bandwidth** (GSM, CDPD: less than 20 kbps)
 - ◆ **high bandwidth variability** (roaming outdoors vs. plugged into docking station)
 - ◆ heterogeneous networks (outdoors: cellular, indoors: infrared LAN)
 - ◆ security risks (everyone can listen in on wireless channel)

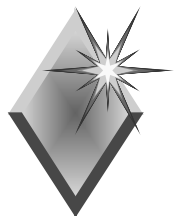




Motivation

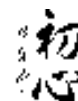
- ◆ Mobility:
 - ◆ address migration (mobile IP, location management)
 - ◆ location dependent information (where is closest printer?)
 - ◆ migrating locality (optimal connection changes over time)
- ◆ Portability (of computing device):
 - ◆ **limited power supply** (battery)
 - ◆ data security (someone steals your laptop)
 - ◆ small user interface (screen size of a PDA)
 - ◆ **limited storage capacity** (disks might be considered as a “power liability”)
- ◆ Solutions (or suggestions) exist to address all these issues, however, the solutions are often conflicting:
 - ◆ do not keep data on portable device to reduce security problem
 - ◆ do not send/receive data unless absolutely necessary to preserve valuable energy

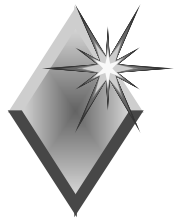




Motivation

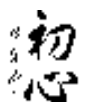
- ◆ Frequent idea: application partitioning (split client between mobile and base station)
 - ◆ reduce bandwidth requirements
 - ◆ overcome the physical limitations of the mobile computer
- ◆ How do you partition and when?
 - ◆ static partitioning is easier (many WWW browser prototypes)
 - ◆ however, computation environment can change radically (during execution or for different executions of same binary):
 - ◆ mobile moves to area with better networking infrastructure (maybe gets back into his docking station)
 - ◆ PCMCIA cards are plugged in and add more memory or devices
 - ◆ radio signal drops completely temporarily
- ◆ Growing interest in dynamic (run-time) partitioning

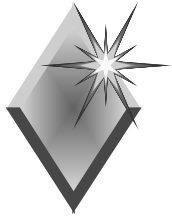




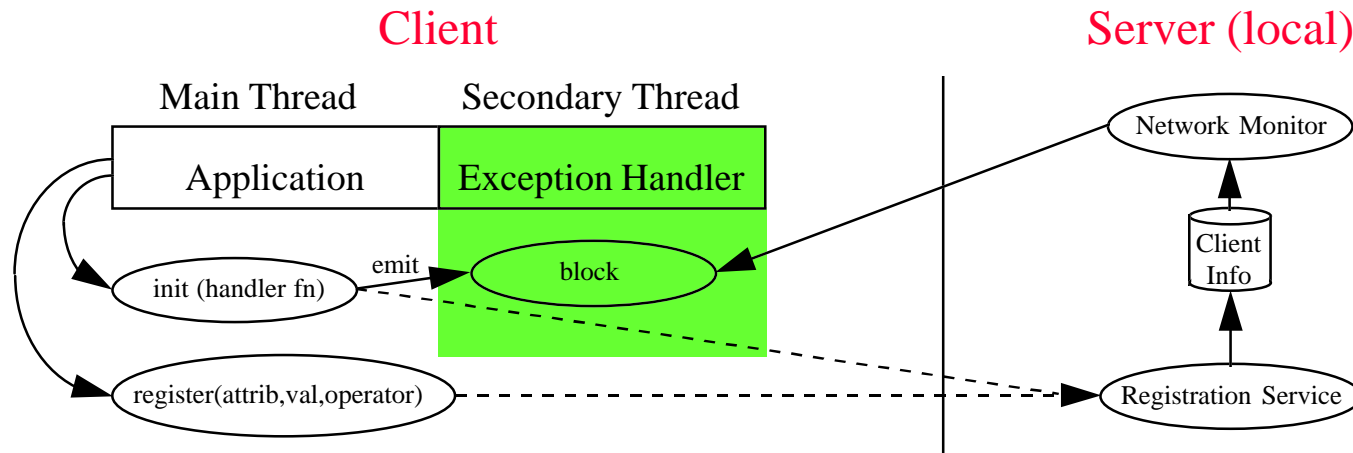
Motivation

- ◆ Dynamic Application Partitioning:
 - ◆ use same binary for different network scenarios
 - ◆ obtain information about environment during runtime:
QoS API
 - ◆ adapt to currently observed conditions dynamically:
adaptive mobile applications
 - ◆ test ideas with different applications and under different network scenarios
 - ◆ iterative, so we can refine what information is relevant in API
 - ◆ explore different ways to provide adaptivity
 - ◆ identify generic services that could be “factored out” and supported by runtime system



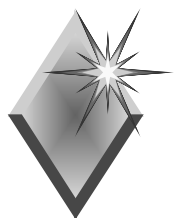


QoS API

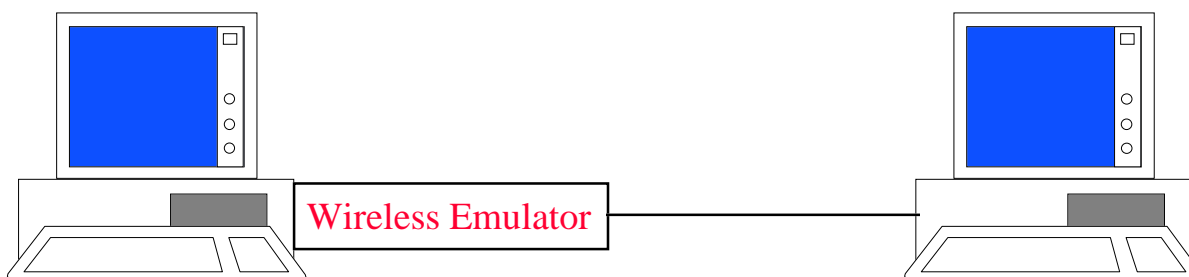


- ◆ how to notify application of changes: call-back functions
- ◆ what information do applications require?
 - ◆ all network parameters available via SNMP
 - ◆ reduced set of derived parameters: round-trip time to base station, CPU load averages, packets send/received and error rates for last 30 seconds, 1 minute, and 5 minutes
 - ◆ host characteristics (type and speed of CPU, list of devices)





Testbed

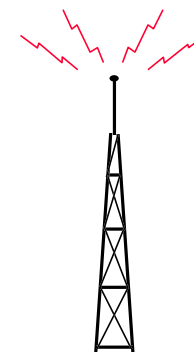


Two (or more) RS6k workstations, connected by 10 Mbps Ethernet and 100 Mbps FDDI
One workstation runs a modified Ethernet Device Driver, which slows down IP packets:

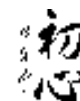
- 10 kbps (outdoor cellular)
- 100 kbps (future generation outdoor cellular)
- 1-2 Mbps (indoor wireless LAN)

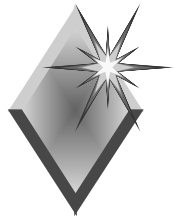


5 laptops with WaveLAN
PCMCIA cards, running Linux, Mobile IP



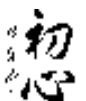
3 WavePOINT basestations,
connected to different IP
subnetworks in 2 buildings

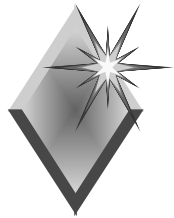




Testbed: The Wireless Emulator

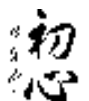
- ◆ Network device driver for AIX 4.2.0, can be disabled/enabled dynamically
- ◆ Reuse NDD (network device driver) for existing Ethernet card, replace ND (network demuxer) and NID (network interface driver) for existing device “en0”
- ◆ Supports dynamic changes of bandwidth and packet loss probability through user-level process
- ◆ Developed simulator to model different wireless MAC protocols and derive bandwidth and error distributions:
 - ◆ slotted Aloha
 - ◆ CDPD
 - ◆ WaveLAN (under development)

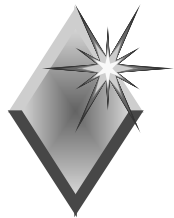




Example 1: HiFi Sound

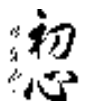
- ◆ High-quality sound files too large to download and play over wireless link in real-time
- ◆ Compression possible, but tradeoffs:
 - ◆ sound quality
 - ◆ overhead in compression/decompression
- ◆ Idea: dynamically adjust compression level to available bandwidth, taking capabilities of mobile host into account
- ◆ Information required from QoS API:
 - ◆ mobile host CPU and speed, maybe even load
 - ◆ link bandwidth, delays, error rates
- ◆ Adaptive application, but no dynamic partitioning

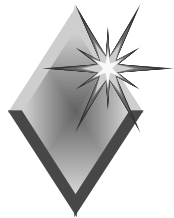




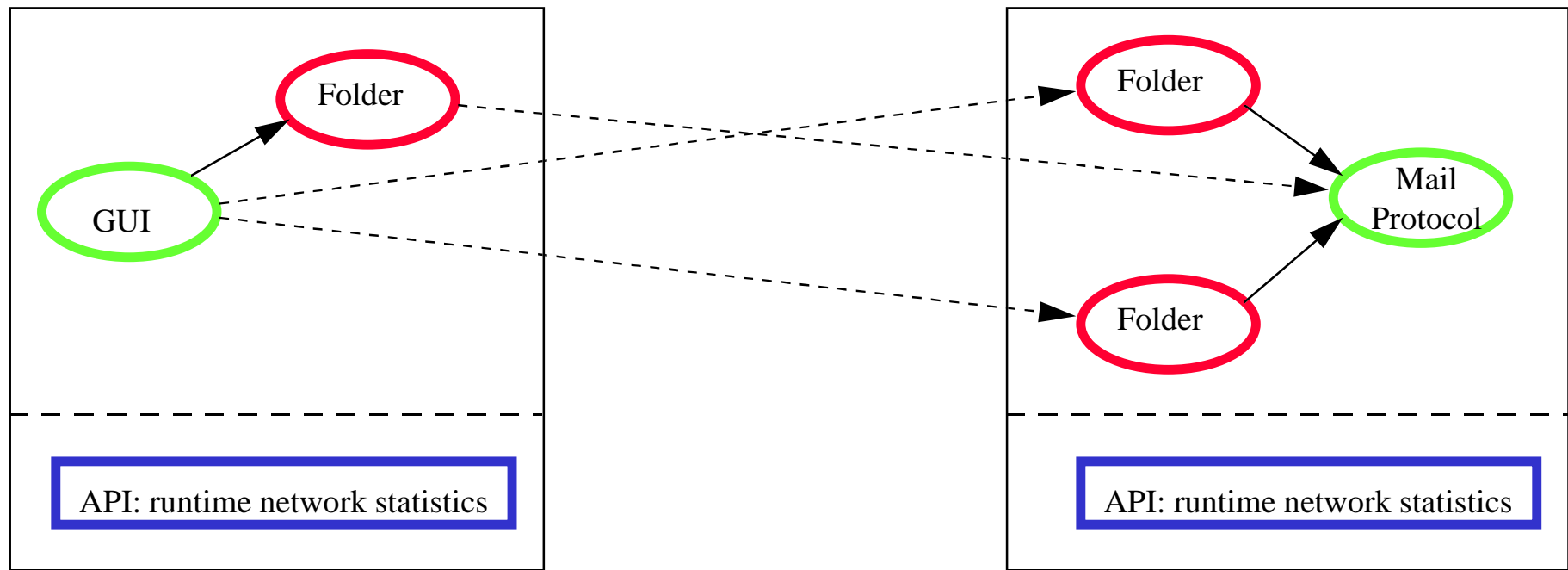
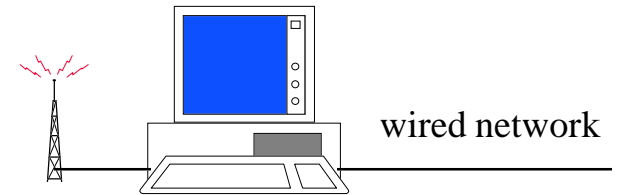
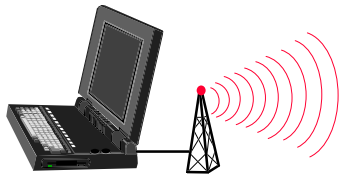
Example 2: Groupware Editor

- ◆ Idea: provide multiple levels of functionality per feature
 - ◆ full functionality: requires lots of resources
 - ◆ limited functionality: designed to reduce bandwidth (primarily)
- ◆ Monitor environment and select feature level:
 - ◆ updates per character or per block (word, paragraph, etc.)
 - ◆ download whole text or overview plus components user works on
 - ◆ locking granularity vs. level of concurrency
 - ◆ disable group awareness features (telepointers, annotations)
 - ◆ local backups in highly unstable environment (frequent disconnections)
- ◆ Information from QoS API: bandwidth, latency, error rates, disconnections, etc.
- ◆ Adaptive application, some dynamic rebalancing of functionality, though mostly static partitioning

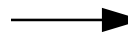




Example 3: M-Mail



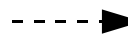
static object



local method invocation



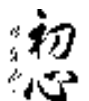
mobile object

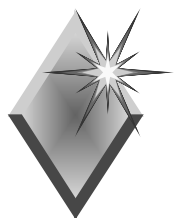


remote method invocation

QoS API information used:

- bandwidth, latency, error rates
- load on mobile host

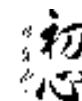


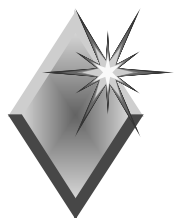


Example 3: M-Mail

Bandwidth	Mobile	Fixed	Dynamic
64 kbps	---	2583	2190
100 kbps	2018 [2002,2034]	637 [608,667]	669 [646,692]
1 Mbps	1346 [1280,1411]	577 [516,607]	532 [486,578]
10 Mbps	369 [353,386]	529 [525,533]	486 [479,492]

- ◆ All times in seconds, intervals are 90% confidence intervals
- ◆ Dynamic partitioning scheme fastest or close to fastest scheme
- ◆ Application Partitioning Scheme not yet optimized:
 - ◆ need to analyze in more detail what network parameters are useful
 - ◆ parameters picked on “educated guess” basis
 - ◆ detailed study of when objects get created where
 - ◆ support for object migration still missing
- ◆ Tests at speeds below 100 kbps failed due to problem with runtime system of underlying distributed object-oriented system





Conclusions

- ◆ Wireless communication and portable devices enable new visions of mobile computing: ubiquitous computing, nomadic computing,
- ◆ Existing applications cannot run unmodified over wireless links:
 - ◆ mismatch between characteristics of wired and wireless links
 - ◆ will not adapt to changes in environment
- ◆ We view wireless case as general case, wired scenario as a special case
- ◆ Demonstrated how to design and implement **adaptive** mobile applications, using information delivered by our **QoS API**
- ◆ Future work:
 - ◆ redefine and enhance information offered by API
 - ◆ identify and implement supplementary services: keeping TCP connections alive, prioritizing connections, etc. (in progress)
 - ◆ continue exploration of dynamic application partitioning of object-oriented applications, utilizing mobile agents technology

