

Mobile Computing

Manish Mahajan



Topics Covered

- System Support for Mobile, Adaptive Applications
- A Comparison of Mechanisms for Improving TCP Performance over Wireless Links



System Support for Mobile, Adaptive Applications

- Notion of Adaptability.
- Application adapts/changes the quality of data it delivers depending on the available resources (system, network, etc)
- E.g. PDA with video (limited), audio capabilities



Problem

- Anytime computing is exciting but taking advantage is difficult!
- Quality of network connectivity afforded to mobile users is turbulent; changes frequently, dramatically and without warning
- Applications must cope with this change



Reasons

- Wireless channel is subject to path loss, fading and interference which impact it's performance
- Different sets of overlapping technologies provide different tradeoff between bandwidth, coverage, cost, and reliability



How to deal...

- Take advantage of high-quality connectivity when available, but behave reasonably over networks with poor performance
 - E.g. Web caches tradeoff disk space with wide vagaries in network performance
- Such techniques don't hide orders-of-magnitude changes in performance very common in wireless
- Rather than relying on system, application must themselves adapt to prevailing network characteristics



Briefly...

- We look at Odyssey
 - A platform for adaptive mobile data access
- Odyssey Approach: Adjust quality of accessed data to match available resources
 - E.g. web browser asks for more compressed images; video player may reduce frame rate or frame quality
- To trade quality for performance, a notion of quality called 'fidelity' is introduced



Context for this work

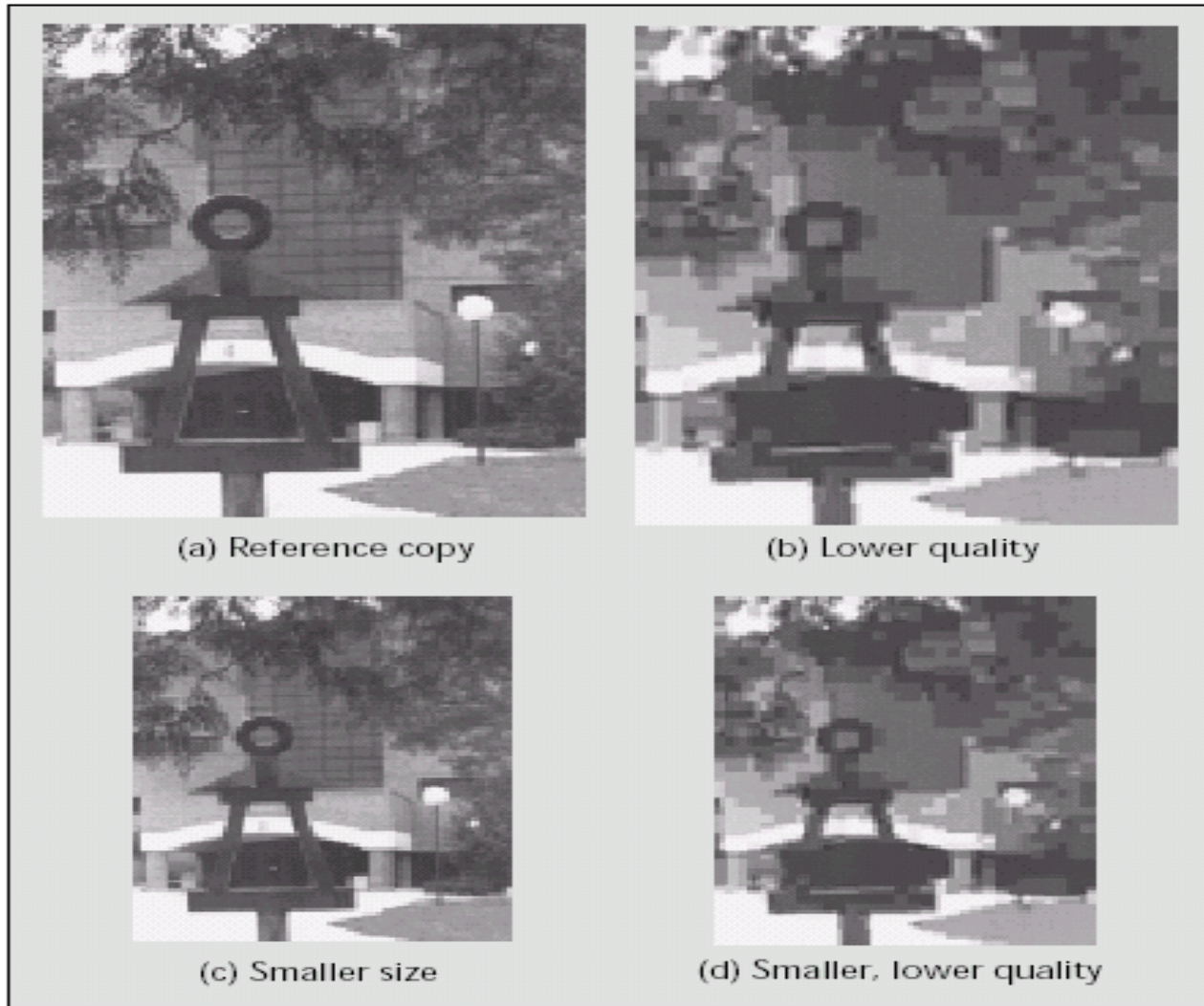
- Odyssey designed to support broad range of mobile information access applications
- These applications reside on a mobile client, but access data on remote server (need not be mobile)
- The client is best positioned to understand what resources are available to it and what it's current priorities are in using them
- Adaptive decisions are driven entirely by client
- Servers support by providing data at various qualities but don't actively participate in resource monitoring



Fidelity: A measure of data quality

- Degree to which a data item used by a mobile client matches a reference copy (a comparison standard)
- Has many dimensions
 - Consistency: shared by every data item regardless of type
 - E.g. Cached data updated by host with varying bandwidth
 - Knowledge of item's structure
 - E.g. video representation: reduced frame rate, quality of frame
- Fidelity is per type, NOT per application and is quantified on a single dimension/data type
 - E.g. fidelity metrics are assigned scalar value between 0.0 (no information content) and 1.0 (reference copy)

Fidelity dimensions for images





Various Collaborations

- Laissez-faire (do as you wish, gives full freedom)
 - Each application must infer the resource consumption of the others to make best possible adaptation decisions.
 - No common point of resource control
 - They don't have accurate knowledge of one another, and will tend to adapt at cross purposes
- Application-transparent adaptation
 - Operating system is wholly responsible for making adaptation decisions for applications
 - Competition for scarce resources is accounted for but individual applications cannot make own adaptive decisions
- Middle ground...

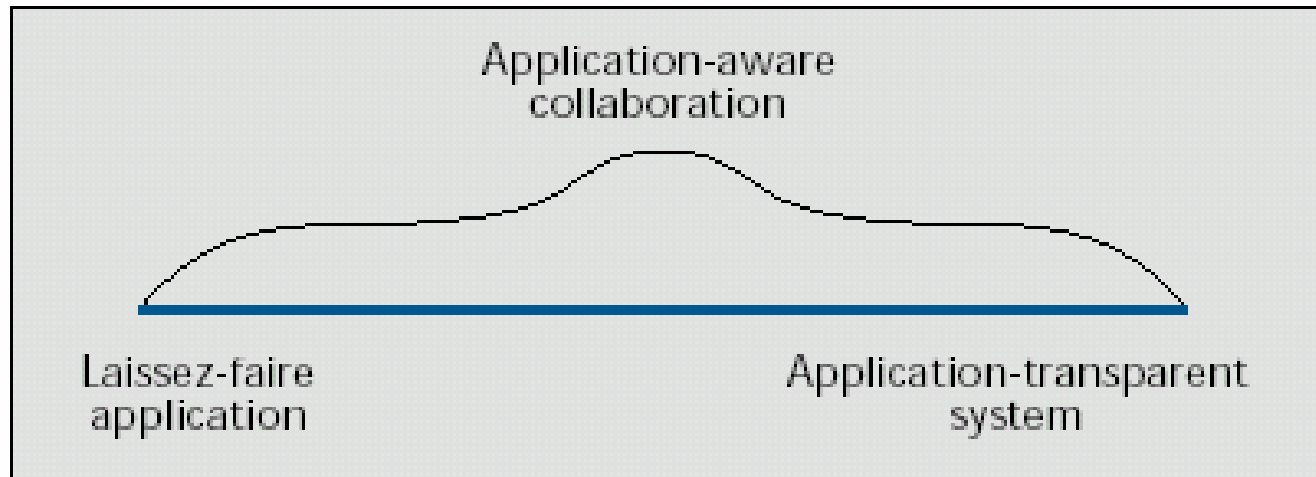


Application-Aware Adaptation

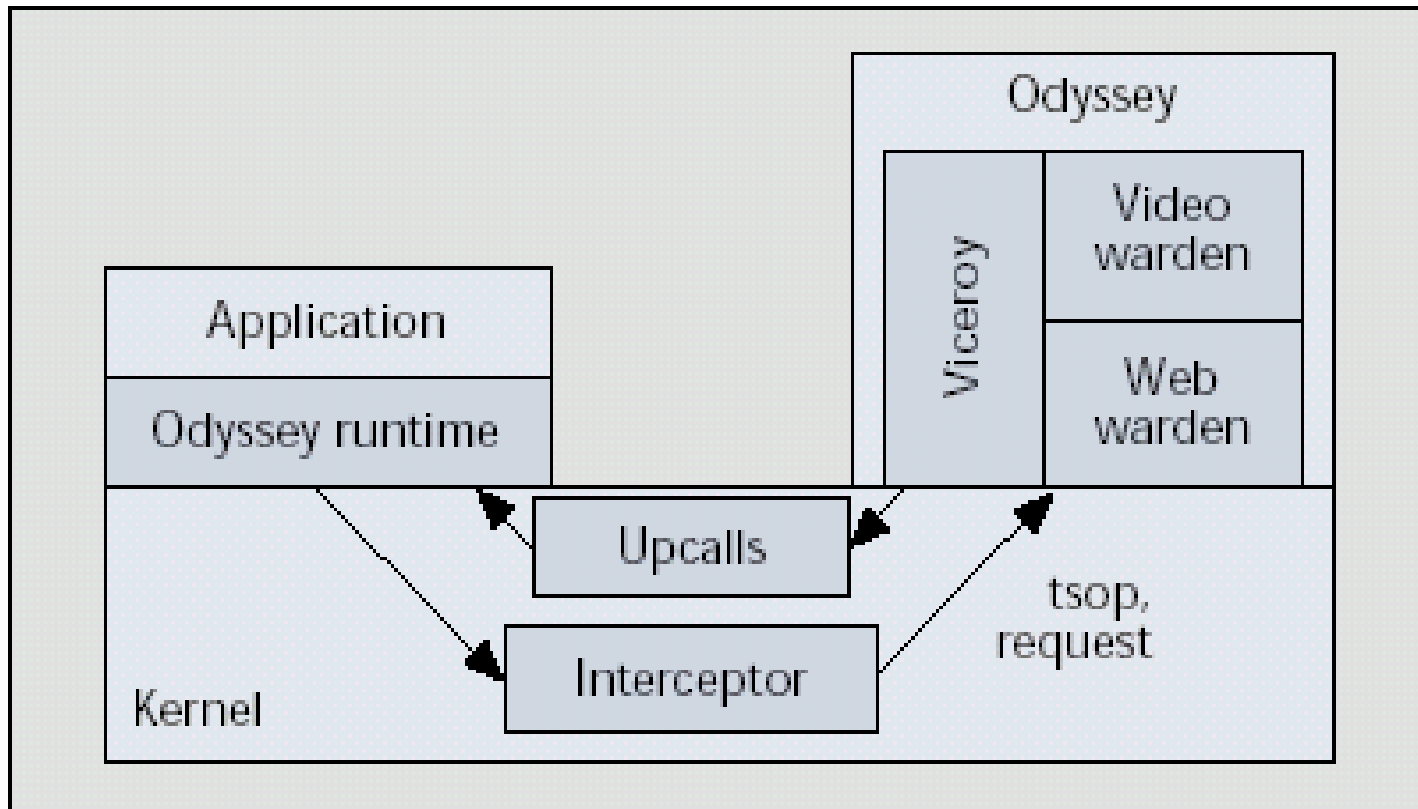
- Decisions must be jointly taken by both parties
- OS, as arbiter of shared resources can best determine resource availability and provides mechanisms for adaptation and common point of resource control
- Application can properly decide how to adapt to a given situation, and must be free to specify adaptive policy
- This approach describes a spectrum of approaches between laissez-faire and application-transparent adaptation



Spectrum of adaptation models



Odyssey Client Architecture





Client Architecture...

- Odyssey can be thought of as a set of OS extensions supporting adaptive applications
- Implemented as kernel component in user space for simplicity
- Odyssey proper has: Viceroy and Set of Wardens
- Viceroy is the single point of resource control
 - It monitors and notifies apps of significant changes in resource availability
- Wardens manage all communication between the client and various servers and offer a menu of fidelities from which apps can pick



Working of the System

- Application chooses fidelity and informs Odyssey of windows of tolerance (flexibility to change) through *resource request* system call
- This is passed on to Viceroy which notifies app of any estimate that lies outside of its declared window through an *upcall*
- *Upcall* carries new resource value so that app might properly react



Working of System...

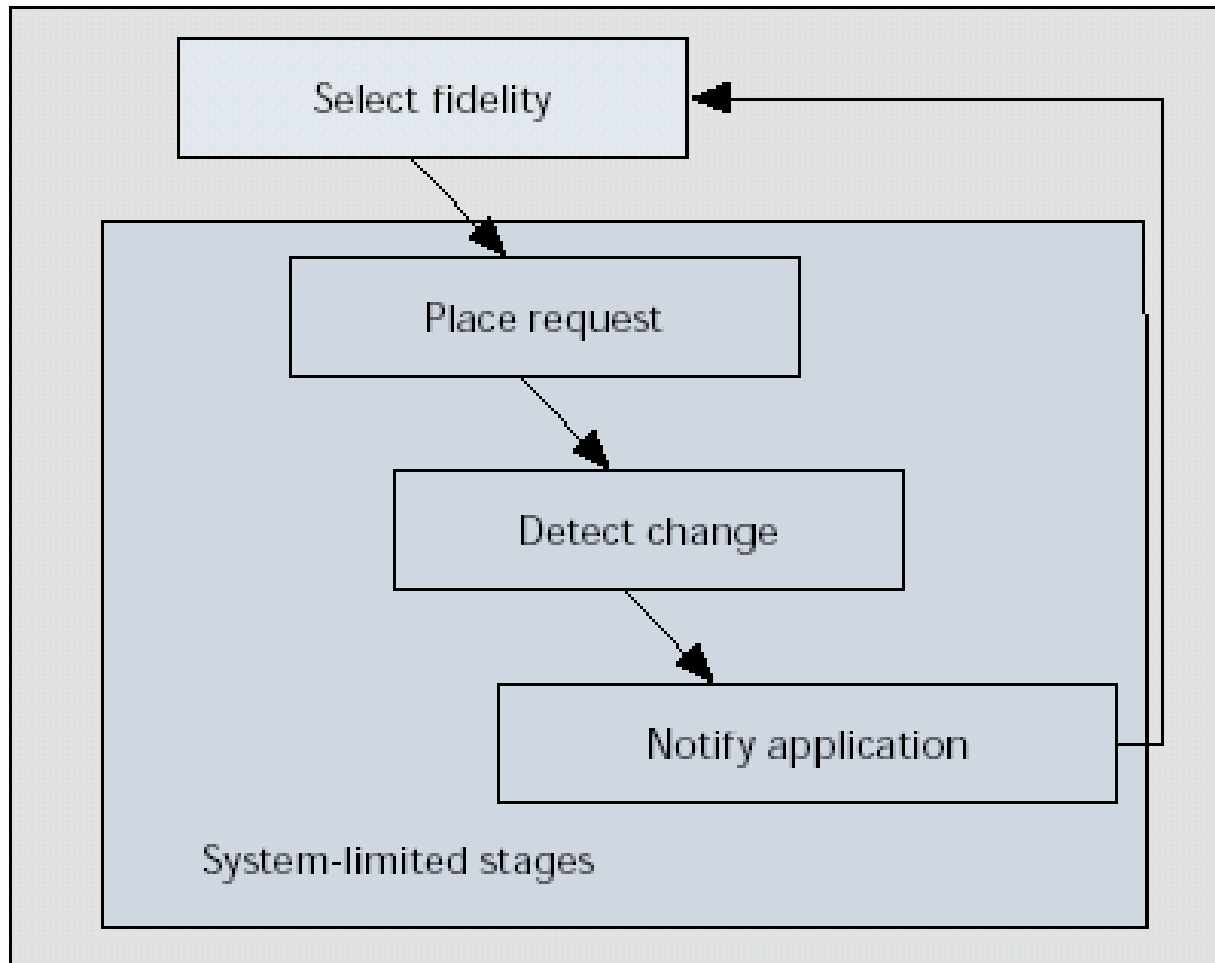
- Wardens incorporate type specific knowledge into resource usage decisions
- The type-specific operation (tsop) provides fidelity changing operations and enables the provision of type-specific access methods
 - E.g. video apps can speak in terms of frames of video, rather than range of bytes



Programming Model

- Adaptive decision loop (figure next slide)
- This control loop first selects a fidelity, then places resource requests with the viceroy as appropriate
- The viceroy monitors resource availability and when significant change is detected, informs each interested application
- The application then select a new fidelity, repeating the process

Adaptive decision loop





Agility: The limit of Adaptation

- Agility: System must be able to react to true changes in the environment quickly else it will be forced to lower its expectations or suffer unacceptable performance
- Agility of Odyssey is determined by adaptive decision loop (place request, detect change, notify app)
- Measuring cost of placing request and notifying apps is straightforward
- Hard part is to measure system's ability to detect changes in network performance



Measuring Agility

- Technique called *network trace modulation* allows live software to be run over a simulated networking environment in real time
- Environment can be generated *empirically* or *synthetically*
- These simulated techniques are useful as wireless networks exhibit performance that is both rapidly changing and impossible to duplicate precisely
- Empirical traces are generated by measuring wireless network performance from mobile client's viewpoint
- The observations are distilled to a replay trace which recreates the performance



Continued...

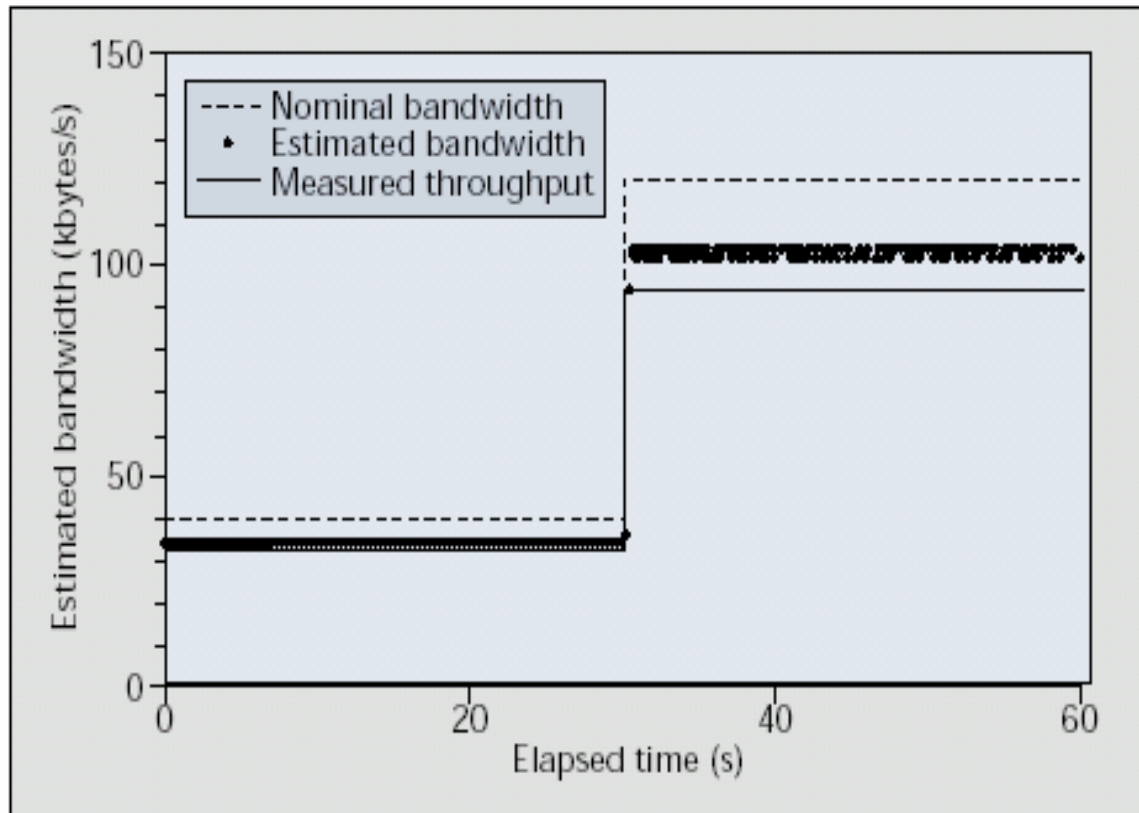
- Empirical trace falls short in assessing the agility with which one can detect changes in networking performance
- Synthetic traces provide instantaneous, ideal changes in network bandwidth
- Very similar to control-theoretic technique of impulse-response analysis
- It characterizes real time performance and simplifies understanding of what limits to agility are placed by the network estimator



Network Estimation in Odyssey

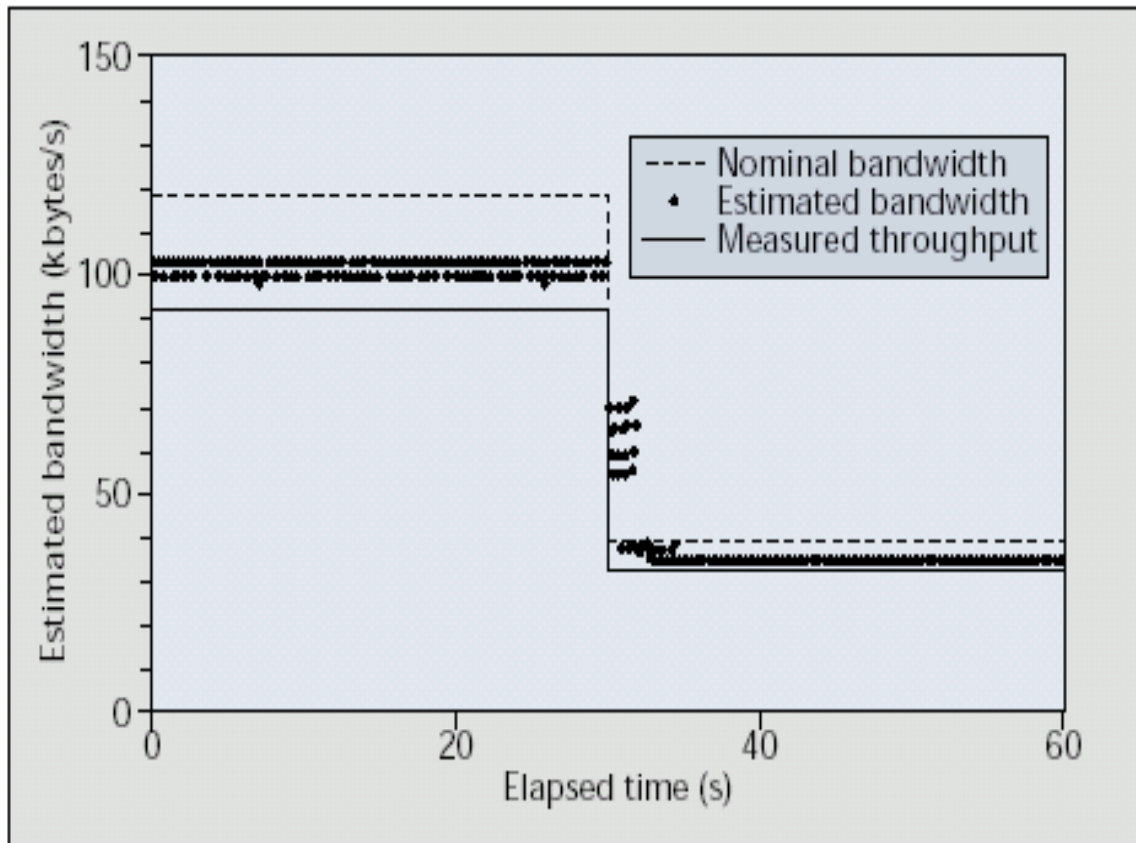
- Odyssey client estimates the quality of networks paths used by various apps
- It does a passive observation of the traffic generated by these apps all of which pass through the viceroy
- It estimates throughput by observing the time needed to complete each window of a bulk transfer
- It estimates bandwidth from throughput and latency
- It updates these parameters using the most recent observations to provide the best agility possible

Limit on Agility (detecting increased bandwidth)



- A perfectly agile Estimator would produce observations entirely within these two curves
- Increases in bandwidth are easy to detect; estimates reach new value within sampling period

Detecting reduced bandwidth



- Several estimates fall between Old and new values
- Large window problem when network bandwidth falls compounded by the protocols efforts to perform its own adaptation



Stability

- Agility defines the most turbulent environment in which a system can operate
- However the key is to balance agility with stability
 - E.g. video player with three fidelities: high, low, grayscale
 - Rapidly switching between first two is fine but not between last two
- Users are tolerant of frequent changes between fidelities with small perceptual differences, but are intolerant of frequent, perceptually large changes
- In Odyssey, stability is properly incorporated by individual apps, as their circumstances warrant but not in core (viceroy) as this would limit its applicability



Continued...

- Flexibility tuning: each proposed increase in flexibility might be met with some degree of skepticism, proportional to the perceptual distance between old and new fidelities
- How can it be done: The viceroy assists apps in the task of incorporating skepticism but doesn't provide stability
- While informing app of changes in resource availability, it could also include information about the expected variance in that estimate



Future work

- Currently extending the work on three fronts: notion of fidelity, supporting peer-to-peer, collaborative services, and improving the quality of information passed across the software/radio interface
- To bridge the gap between wireless devices and mobile software by exchanging the right information between software layer (producer of packets) and wireless network (forward and delivery as best)



Conclusion

- Structuring an adaptive, mobile system as a collaboration between the OS, which provides the mechanisms for adaptation, and the apps, which supply adaptive policy, has proven to be powerful
- Apps can take their own goals into account when choosing how to adapt to mobile environments, but can do so safely in the presence of other apps competing for the same set of scarce resources



A Comparison of Mechanisms for Improving TCP Performance over Wireless Links

- This paper compares several schemes designed to improve the performance of TCP in wireless and other lossy links
- Reason: TCP responds to all losses (losses due to bit errors and handoffs along with congestion losses) by invoking congestion control and avoidance algorithms
- This results in degraded end-to-end performance in wireless and lossy systems



The Various Schemes

- Schemes are classified into three broad categories
 - End-to-end protocols, where loss recovery is performed by sender
 - Link-layer protocols, that provide local reliability
 - Split-connection protocols, that break the end-to-end connection into two parts at the base station
- Results show that
 - Reliable link-layer protocol that is TCP-aware is good
 - Good performance possible without splitting e-2-e connect
 - Selective acks and explicit loss notifications also provide significant performance improvements



Introduction

- TCP (tuned for wired links and stationary hosts) assume congestion in the network to be the primary cause for packet losses and delays
- It does well by adapting to such delays and losses utilizing various schemes such as: dropping transmission window size, slow start, backing off retransmission timer
- These measures result in a reduction in the load on intermediate links thereby controlling congestion



Continued...

- In wireless, packets are lost for reasons other than congestion (i.e. high bit-error rates, intermittent connectivity due to handoffs)
- Here TCP's measures result in an unnecessary reduction in e-2-e throughput and in sub-optimal performance
- TCP performance in such networks suffers from significant throughput degradation and very high interactive delays



Continued...

- Various schemes proposed to alleviate these effects
 - Local retransmission
 - Split-TCP connections
 - Forward error correction, to improve e-2-e throughput
- This paper examines and compare the effectiveness of these schemes



Two approaches

- First approach hides any non-congestion-related losses from TCP sender and requires no change to sender
 - Intuition is that since the problem is local it should be solved locally and transport layer need not be concerned
 - Protocols that adopt this approach attempt to make the lossy link as a higher quality link with reduced effective bandwidth
 - As a result most losses seen by sender are caused by congestion



2nd Approach

- Second approach attempts to make the sender aware of the existence of wireless hops and realize that some packet losses are not due to congestion.
 - Sender can avoid invoking congestion control algorithms when non-congestion-related losses occur



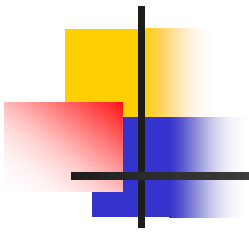
E-2-E protocols

- These attempt to make the TCP sender handle losses through the use of two techniques
 - Use selective ack (SACKs) to allow sender to recover from multiple packet losses in a window without resorting to a coarse timeout
 - Attempt to distinguish between congestion and other forms of losses using an explicit loss notification (ELN) mechanism



Split-connection Approach (other end of the spectrum)

- These completely hide the wireless link from the sender by terminating the TCP connection at the base station
- Use separate reliable connection between the base station and the destination host
 - The second connection can use techniques such as negative or sack rather than standard TCP to perform well over wireless link



Link-Layer Solution (solution in between the two approaches above)

- These protocols attempt to hide link related losses from the TCP sender by using local retransmission and perhaps forward error correction over the wireless link
- Local retransmission use the characteristics of wireless link to provide a significant increase in performance



Related Work

- Link Layer Protocols
 - Two main classes of techniques employed by these protocols are error correction, using FEC and retransmission of lost packets in response to automatic repeat request (ARQ) msgs
- Main advantage is that it fits naturally into the layered structure of network protocols
- The link layer operates independently of higher-layer protocols and does not maintain any per-connection state
- Main concern is the possibility of adverse effect on certain transport-layer protocols such as TCP

Continued...

■ Split Connection Protocols

- Split each TCP connection between a sender and receiver into two separate connections at the base station: one between sender and base station and other between base station and receiver
- Over the wireless hop, a specialized protocol tuned to the wireless environment may be used (e.g TCP or selective repeat protocol (SRP) on top of UDP)
- Disadvantage: TCP sender of wireless connection often times out, causing original sender to stall. Also every packet incurs overhead of going through TCP protocol processing twice at the base station
- E-2-e semantics are of TCP acks is violated since acks to host reach before pkts actually reach the receiver
- Maintains a significant amount of state at the base station/TCP connection, handoff procedures tend to be complicated and slow



Snoop Protocol

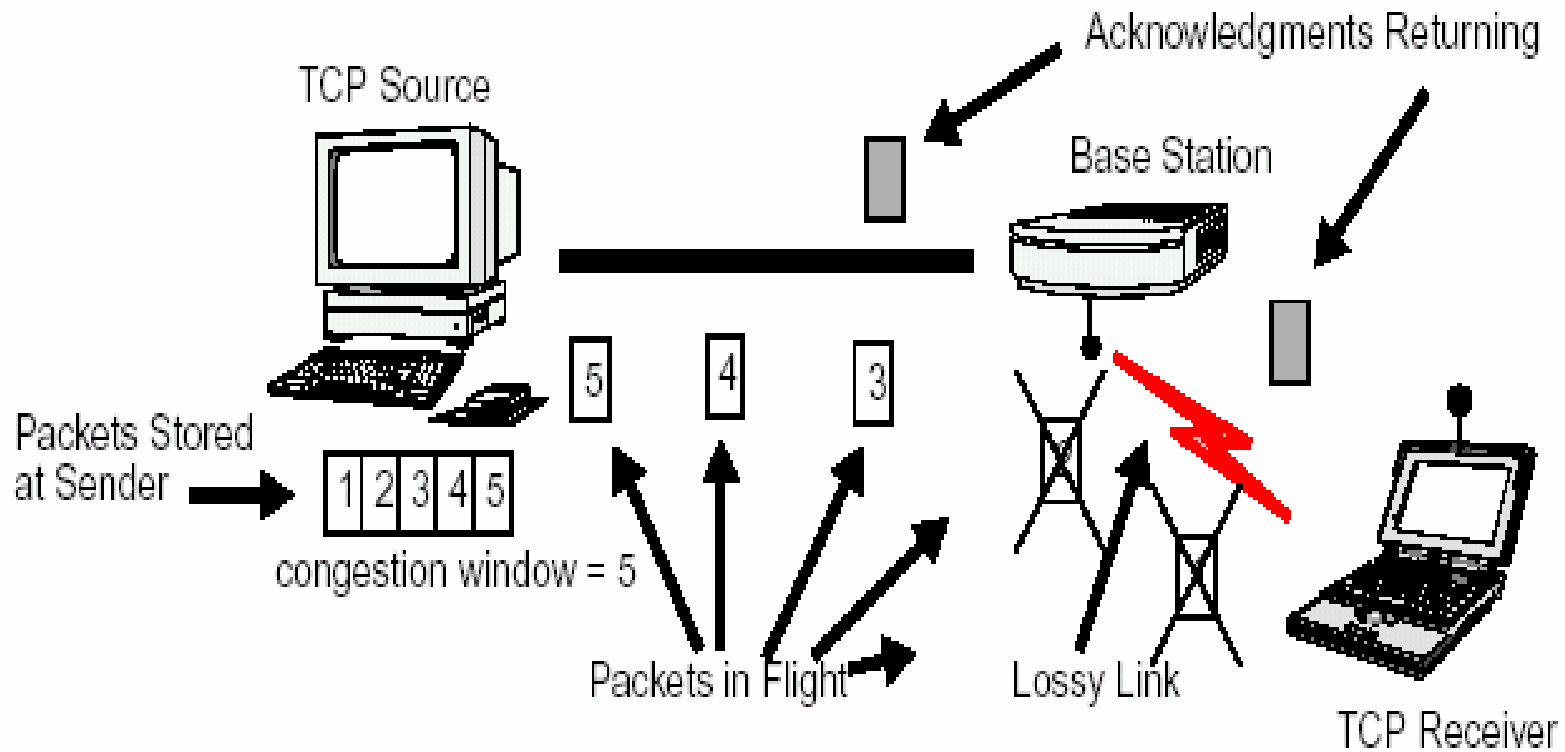
- Snoop protocol is a link-layer protocol that takes advantage of the knowledge of higher-layer transport protocol (TCP)
- Snoop agent monitors every packet that passes through TCP connection in both directions and maintains a cache of TCP segments sent across the link that have not yet been acknowledged by receiver
- Main advantage: Suppresses duplicate acks for TCP segments lost and retransmitted locally, thereby avoiding unnecessary fast retransmission and congestion control invocations by sender



Selective Acknowledgements

- Std TCP uses cumulative ack scheme, and does not provide the sender with sufficient information to recover quickly from multiple packet losses within a single transmission window
- SACK RFC proposes that each acknowledgement contain information about up to three non-contiguous blocks of data that have been received successfully by the receiver. Each block is described by starting and ending sequence number
- Alternate proposal: SMART uses acks that contain cumulative ack and sequence number of packet that caused receiver to generate the ack. Sender uses this information to create a bitmask of packets that have been delivered successfully the receiver

A typical loss situation





Summary of protocols studied in paper

Name	Category	Special Mechanisms
E2E	end-to-end	standard TCP-Reno
E2E-NEWRENO	end-to-end	TCP-NewReno
E2E-SMART	end-to-end	SMART-based selective acks
E2E-IETF-SACK	end-to-end	IETF selective acks
E2E-ELN	end-to-end	Explicit Loss Notification (ELN)
E2E-ELN-RXMT	end-to-end	ELN with retransmit on first dupack
LL	link-layer	none
LL-TCP-AWARE	link-layer	duplicate ack suppression
LL-SMART	link-layer	SMART-based selective acks
LL-SMART-TCP-AWARE	link-layer	SMART and duplicate ack suppression
SPLIT	split-connection	none
SPLIT-SMART	split-connection	SMART-based wireless connection



Briefly

- E2E-NEWRENO improves performance of TCP-Reno after multiple packet losses in a window by remaining in fast recovery mode
- E2E-SMART and E2E-IETF-SACK protocols add SMART-based and IETF selective acknowledgements respectively to the standard TCP Reno stack. This allows the sender to handle multiple losses within a window of outstanding data more efficiently.
- E2E-ELN protocol adds an Explicit loss notification option to TCP acks



Results: Link-Layer Protocols

- Link-layer retransmission scheme does not entirely avoid the adverse effects of TCP fast retransmission and the consequent performance degradation
- An enhanced link-layer scheme that uses knowledge of TCP semantics to prevent duplicate acks caused by wireless losses from reaching the sender and locally retransmits packets achieves significantly better performance



Result: End-to-End Protocols

- Adding ELN to TCP improves throughput significantly by successfully preventing unnecessary fluctuation in the transmission window.
- E2E protocol that has both ELN and SACKs will result in good performance, and is an area of current work



Result: Split-Connection Protocol

- In summary, while the split-connection approach results in good throughput if the wireless connection uses special mechanisms, the performance is worse than that of a well tuned, TCP-aware link-layer protocol
- Moreover the link-layer protocol preserves the end-to-end semantics of TCP acks
- This demonstrates that e-2-e connection need not be split at the base station in order to achieve good performance