

Energy Management in Wireless Embedded Systems

Primary Reference:

Davide Bertozzi†, Anand Raghunathan‡, Luca Benini† and Srivaths Ravi‡
“Transport Protocol Optimization For Energy Efficient Wireless Embedded Systems”, Design, Automation, and Test in Europe (DATE'03)

† University of Bologna, DEIS, Viale Risorgimento, 2 Bologna

‡ C & C Research Labs, NEC USA, Princeton, NJ 08540

‡ Alphon Corp., Eatontown, NJ 07724

Student Lecture EECS 800: Survivable Networks

Ashwin Kumar Chimata

November 8th 2005

email: ashwinc@ittc.ku.edu

Department of Electrical Engineering and Computer Science
University of Kansas

Outline

- Introduction
- Related work
- Implementation of Low-power TCP buffering
 - TCP receive buffer full
 - TCP receive buffer empty
- Experimental results
- Conclusions
- References

Introduction

- Power consumption in wireless devices directly determine the survivability of the network they participate in.
- Power is used up largely in the network interface card's operation.
- This work is based on identifying reasonable intervals of idle time in the network interface cards which can be exploited to shut it down.
- TCP run-time parameters are used for identification of these idle times.

Related Work

- Improving the energy efficiency of low power wireless embedded systems has been attempted in almost all layers of the protocol stack.
- Adjusting the coding and modulation schemes and controlling transmission powers have been employed at the *physical layer*.
- A major portion of power optimized protocols have been suggested/implemented at the *MAC layer*.
- Though the MAC layer protocols have direct access to channel variations, transitions to sleep mode and active mode require periodic confirmations.

Related Work

- Application specific information is used to either identify or predict data bursts at the *Application layer*, which in turn can be used to conserve energy by turning off the interface card.
- Methods to reduce power, by making the sender to send data in a predictable manner have also been investigated.
- In this paper the transport layer is used to identify power conservation intervals.

Outline

- Introduction
- Related work
- **Implementation of Low-power TCP buffering**
 - TCP receive buffer full
 - TCP receive buffer empty
- Experimental results
- Conclusions
- References

Implementation of Low-power TCP buffering

- In this paper the receiver side of the TCP connection is used for identifying plausible idle times, during which the network interface card can be shut down
- The TCP buffer full and buffer empty states are exploited for finding coarse granular idle times
- Finding a reasonably long idle interval is important because the network interface card takes some time to re-associate with the LAN.

A. TCP receive buffer full

- TCP prevents buffer overflowing by piggybacking the "*receive window*" along with the ACK packets.
- The sender also estimates the amount of data that can be on the fly in the network without causing congestion.
- The sender estimates the number of bytes to be sent by using the following equation:

$$txbyte = \min (rxwnd, cwnd)$$

where rxwnd = receive window

cwnd = contention window

A. TCP receive buffer full : Vanilla behavior

- A FTP download to the FLASH memory of the iPAQ handheld device is considered here.
- FLASH memory write operation is very time consuming, mainly due to the cell erasure overhead
- The Linux Garbage collector which is in charge of this task consumes 70% of the FTP transfer execution time.
- Thus, this is a scenario where buffer is usually filled up soon as data processing rates are large

A. TCP receive buffer full : Vanilla behavior

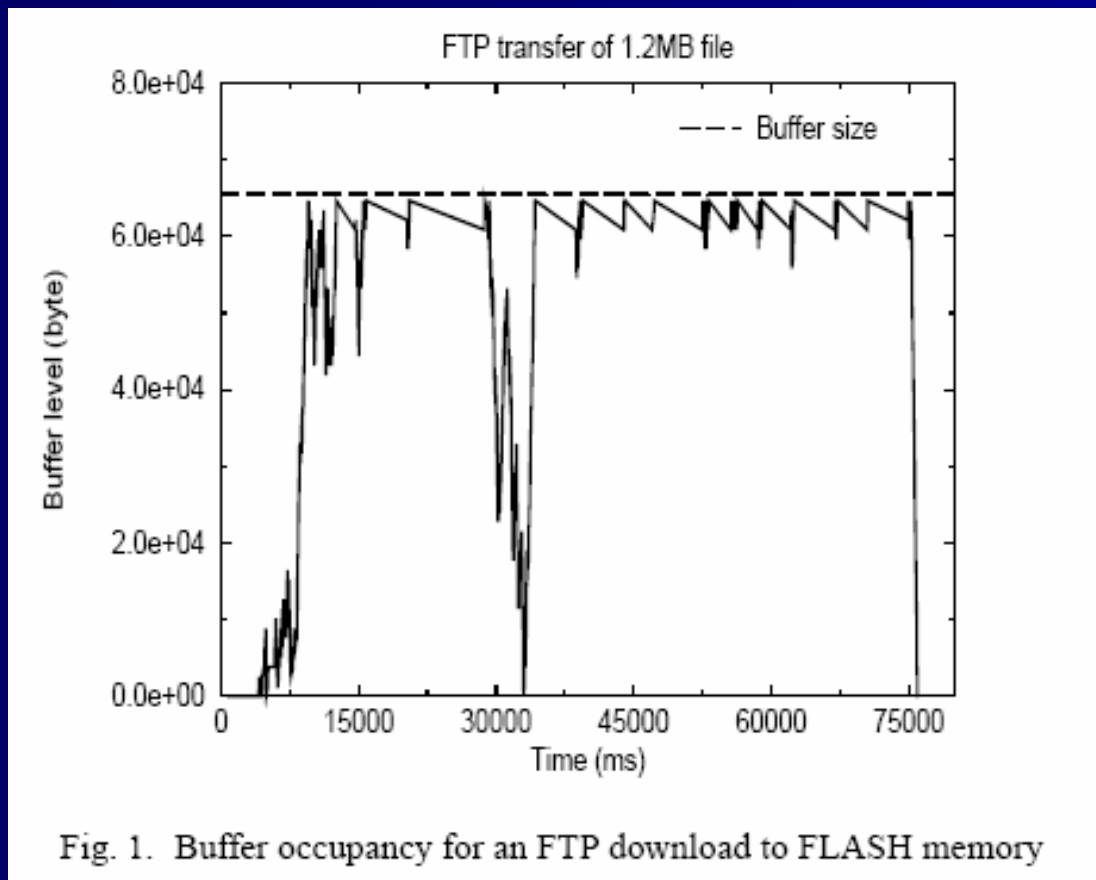


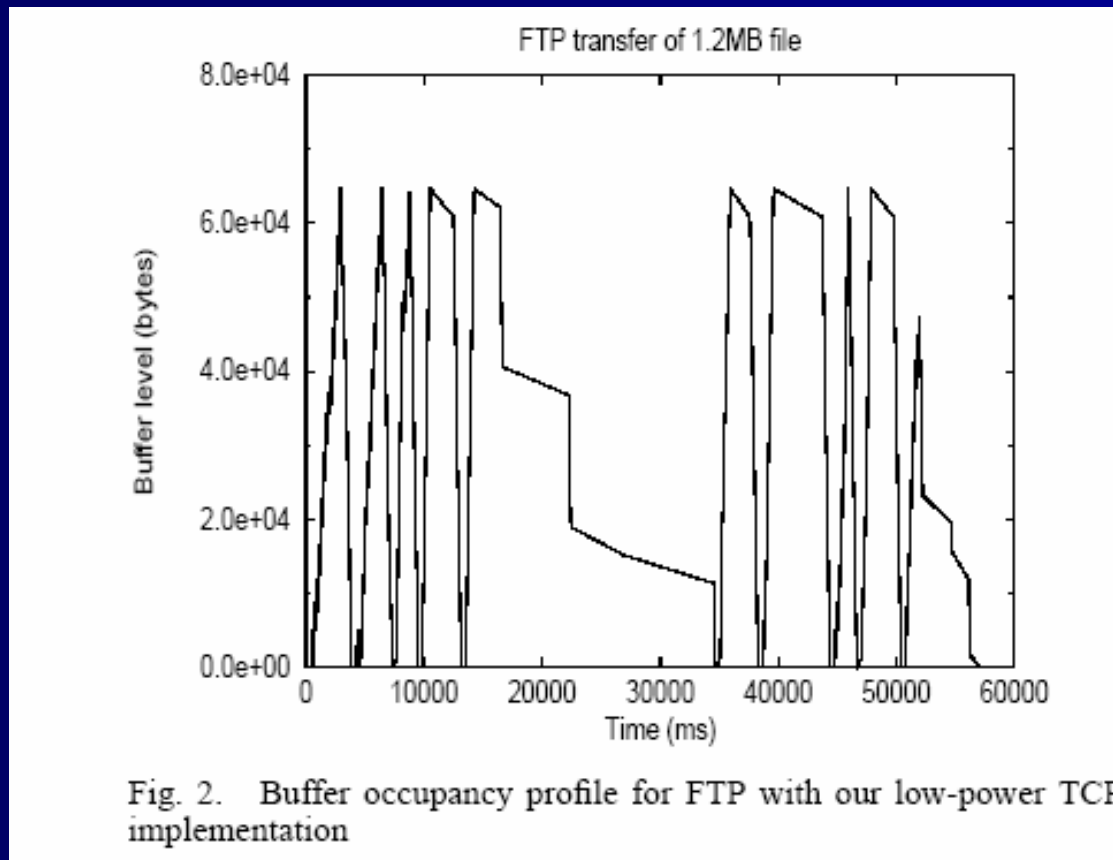
Fig. 1. Buffer occupancy for an FTP download to FLASH memory

Buffer occupancy profile during a FTP download to a iPAQ handheld device running "linux familiar distribution" (LinuxTCP)

A. TCP receive buffer full : Exploited flavor

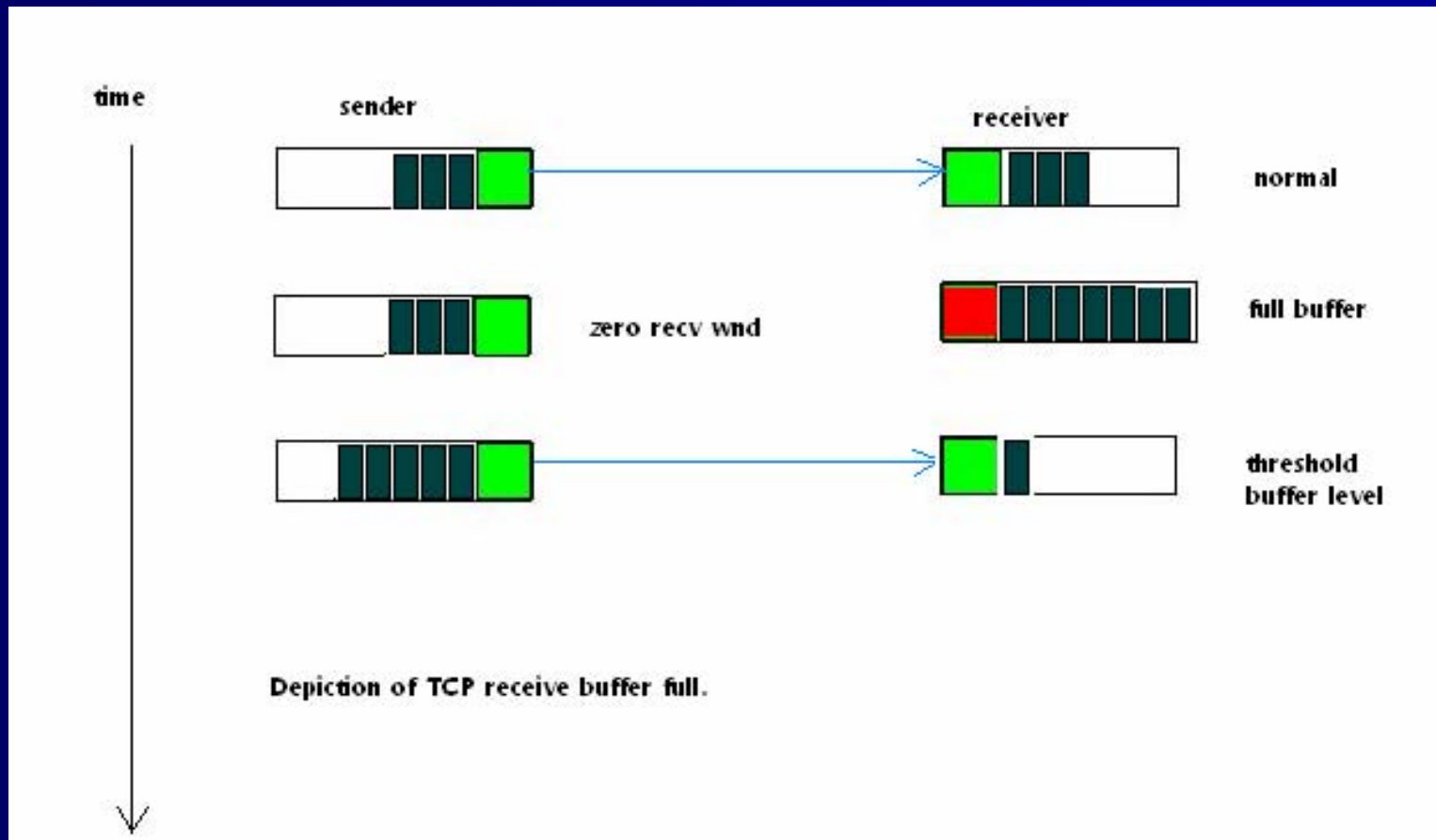
- The inherent client driven flow control mechanism is extended and exploited in this implementation.
- The interface card is completely shutdown between the zero and non-zero TCP receive window advertisements, as we know that the sender will not be transmitting data at this time.
- Further, the buffer occupancy threshold is advertised as a tunable parameter, with the default Linux TCP's as the upper bound
- The new buffer occupancy graph is as follows

A. TCP receive buffer full : Exploited Flavor



Buffer Occupancy Profile for the FTP Transfer when the buffer occupancy advertisements are tuned in accordance with the low power TCP (lpTCP).

A.TCP receive buffer full : Exploited Flavor



B. TCP receive buffer empty

- Idleness due to conditions where there are no open sockets or due to temporary inactivity of sockets can also be utilized for power conservation
- These conditions are typically identified with an empty buffer.
- The status of TCP sockets are monitored to predict traffic flow and to turn the network interface card on/off appropriately.
- Idle times which have predictably longer intervals are preferred over than those induced by ad-hoc channel variations.

B. TCP receive buffer empty

Typical HTTP session's Buffer occupancy profile

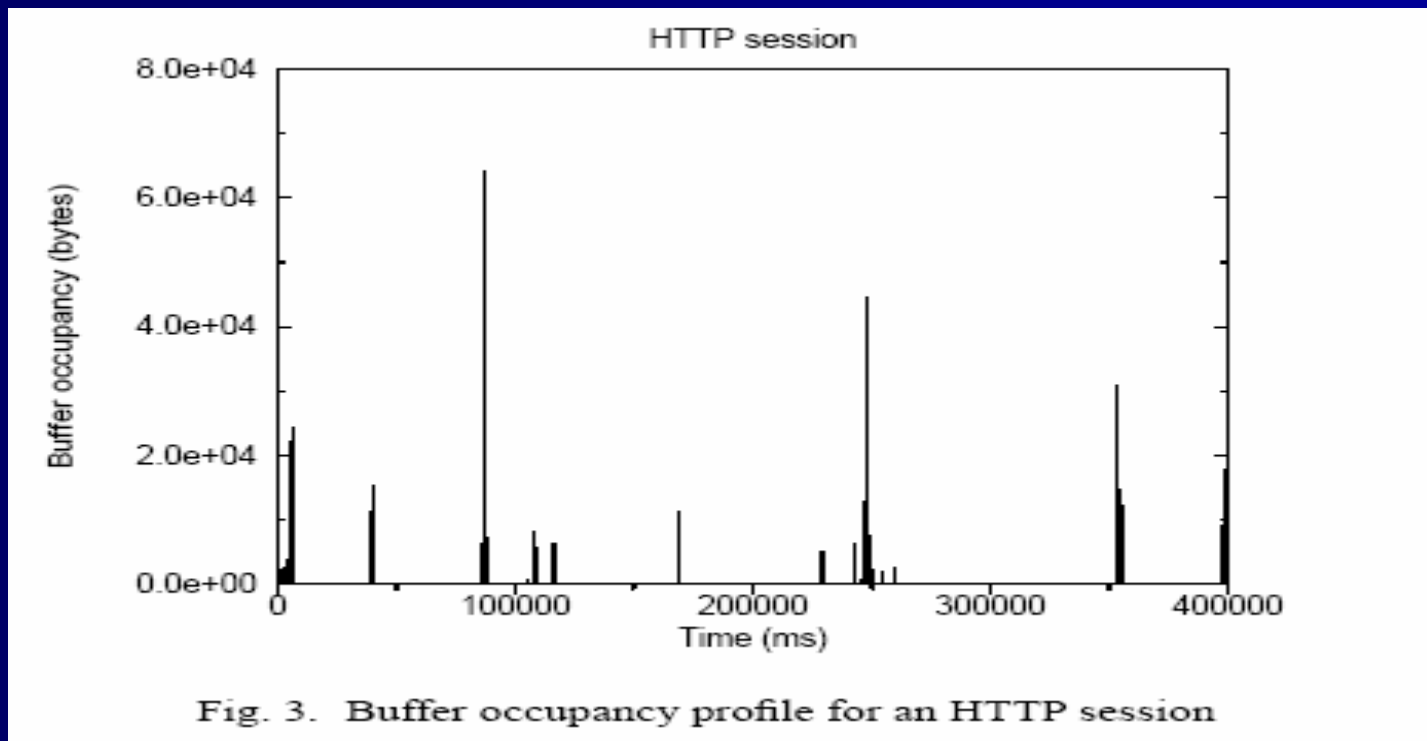


Fig. 3. Buffer occupancy profile for an HTTP session

B. TCP receive buffer empty: Exploited Flavor

- The scenario considered in this case to shut down the network card is; when the buffer is empty as there are no sockets actively using the network.
- This situation is detected in the following way:
 - A timer is activated whenever the buffer becomes empty
 - This timer is reset by subsequent read-write operations
 - Else the timer handler turns off the network interface card
 - The buffer empty case is detected by monitoring the "*tcp_recvmmsg*" which is the function which LinuxTCP makes use of when any application level read function is called.
 - Whenever this function is called, a packet is read from the buffer and freed from it.

B. TCP receive buffer empty: Exploited Flavor

- Therefore checking for buffer occupancy is done every time this function is called.
- The card is reactivated when a new TCP transmission is initiated, or when an application level read call is issued
- If new packets arrive at the interface card, before the "*read*" from the application process is scheduled then those packets will be dropped, as the network interface card will be turned off.
- However, these will only be the initial packets and will be re-transmitted at the expiration of the re-transmission timeout.

Outline

- Introduction
- Related work
- Implementation of Low-power TCP buffering
 - TCP receive buffer full
 - TCP receive buffer empty
- **Experimental results**
- Conclusions
- References

Experimental Results

- All results are based on actual measurements performed using a Compaq iPAQ H3800 PDA running the 'familiar Linux Distribution'.
- The results are a set of comparisons between the Vanilla Linux TCP (LinuxTCP) and the Low power TCP (IpTCP), on top of three MAC level power management modes, namely CAM, PSPCAM and PSP
 - CAM -> Constantly Awake mode
 - PSP -> Power Save Polling
 - PSPCAM -> This mode switches between PSP and CAM

Experimental Results

A. Card shutdown at buffer full

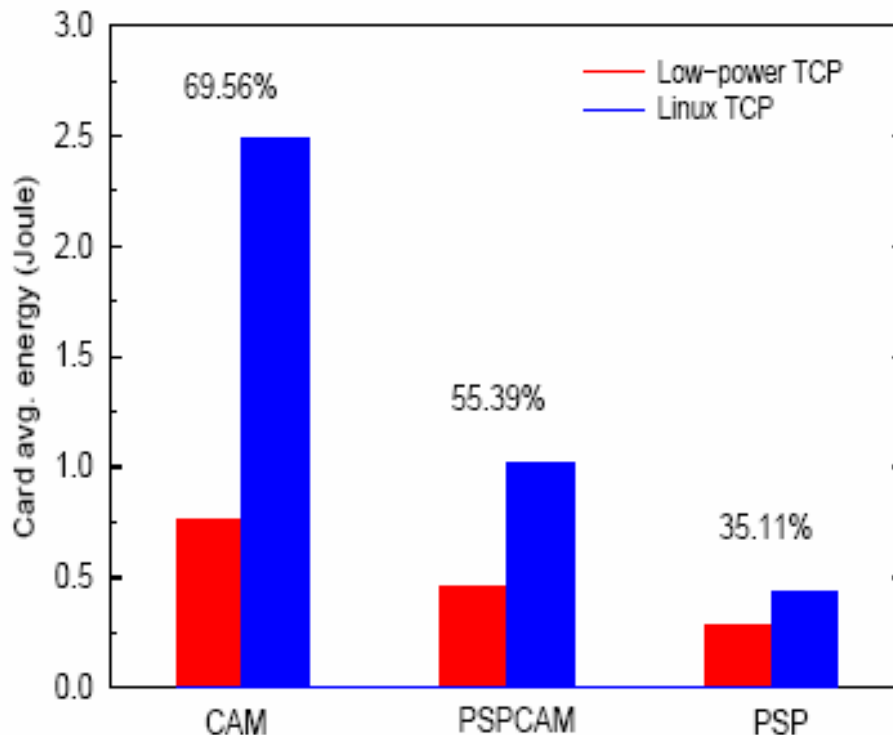


Fig. 4. Energy savings of the proposed lpTCP compared to Linux TCP

An FTP transfer Of 600 kByte file Was executed. The download was Repeated 25 Times to eliminate Spurious network And OS related Variations. Buffer Occupancy Threshold was set To one fourth The default threshold

Experimental Results

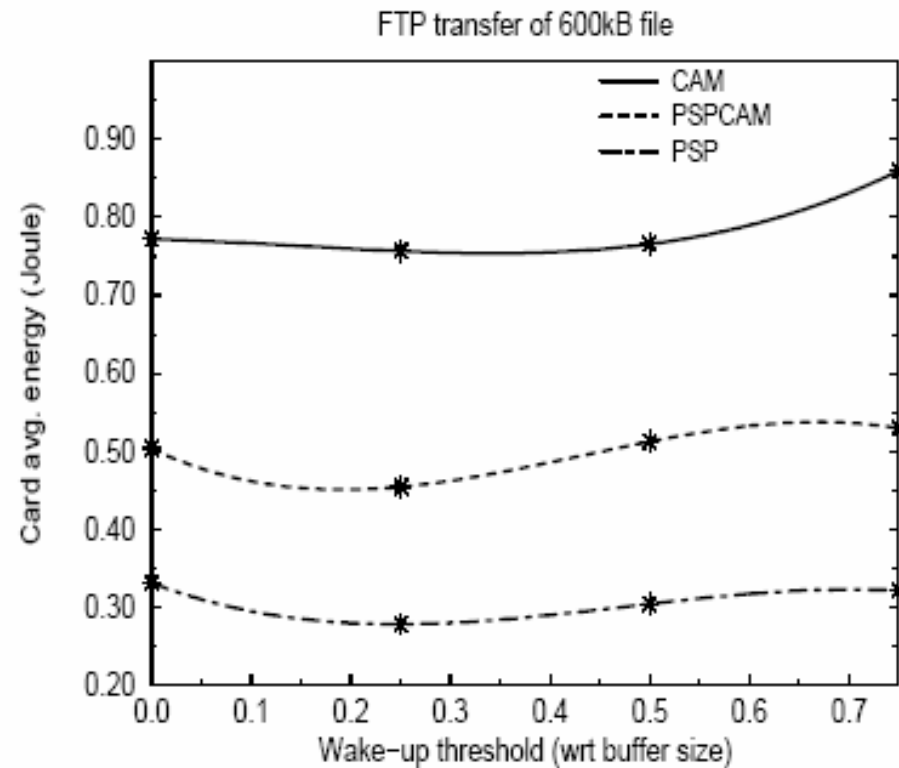


Fig. 5. Energy consumed by the network adapter for a FTP transfer as a function of the wake-up threshold

Experimental Results

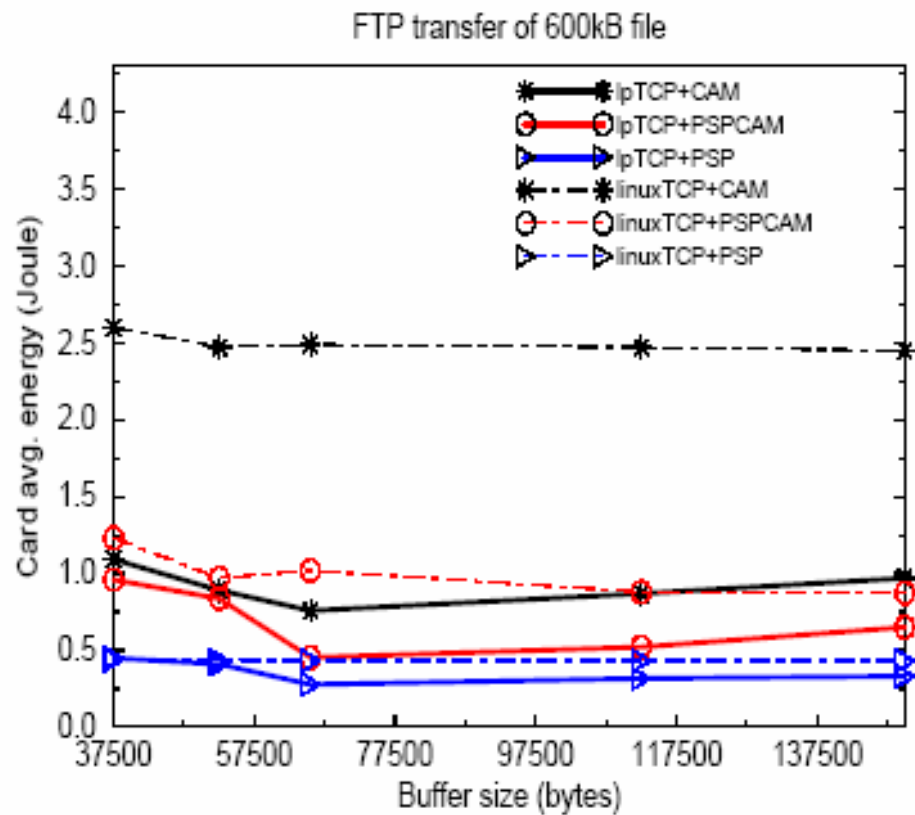


Fig. 6. Dependence of energy on TCP receive buffer size

Experimental Results

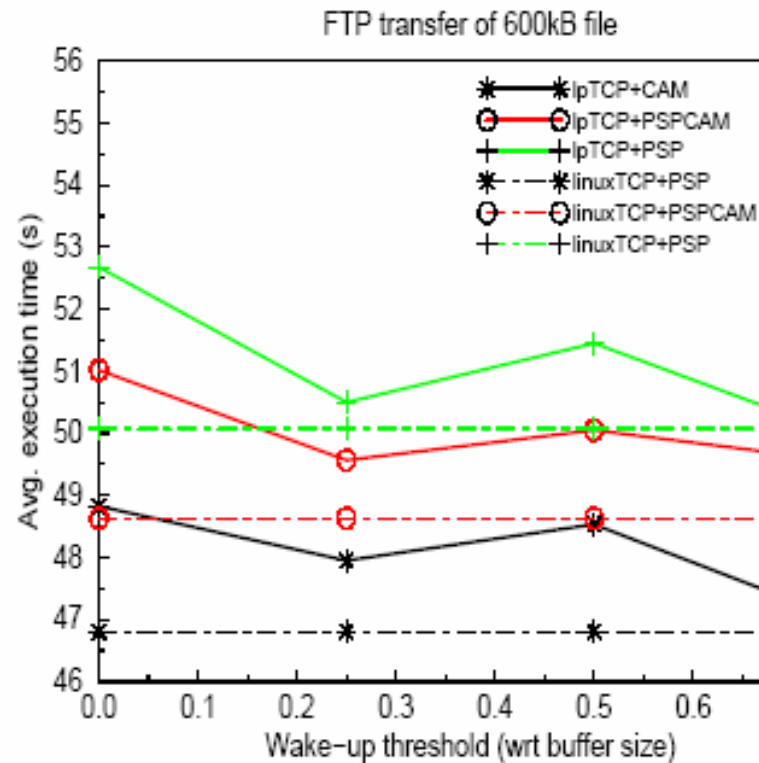


Fig. 7. Impact of our technique on performance, as a function of the card wake-up threshold

Experimental Results

B. Card shutdown at buffer empty

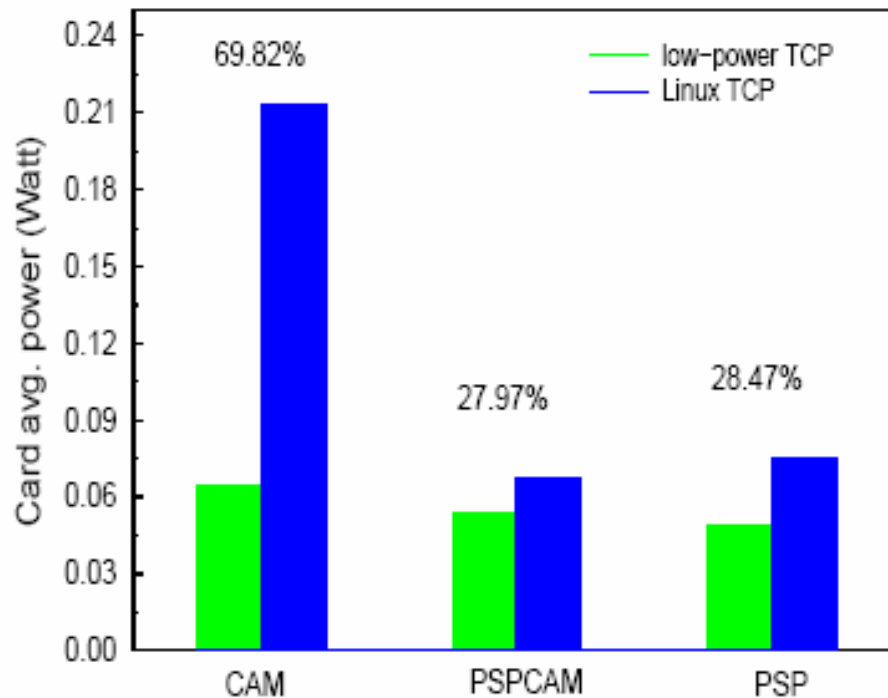


Fig. 8. Power consumption of lpTCP vs. linuxTCP for a HTTP session.

Experiment done
Web browser
(called Dillo)
Session.
The results are
Based on a 25 min
Repeated web-
Browsing sessions

Conclusions

- Energy efficiency of wireless embedded systems can be brought about with optimizations in the network protocols
- This paper has demonstrated an energy efficient TCP implementation.
- As the results are based on actual measurements, this work can be a promising technique for power management for PDAs.

Conclusions: Problems

- As these results are based on specific scenarios, like FTP and Web browsing, might not be as effective for other applications
- Can think of ways to manipulate the sender's network interface card as well.
- ???

References

- Davide Bertozzi, Anand Raghunathan, Luca Benini and Srivaths Ravi, "*Transport Protocol Optimization For Energy Efficient Wireless Embedded Systems*" Proceedings of the Design, Automation and Test in Europe (DATE'03) 2003, IEEE.
- The Linux LXR, *<http://lxr.linux.no/>*