On Efficient and TCP/IP Friendly Link Layers for Wireless WANs

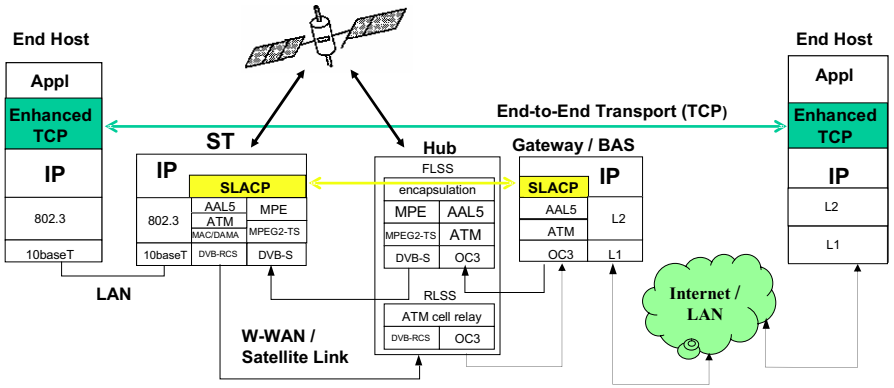
Markku Kojo, Lecturer, MSc

Department of Computer Science
University of Helsinki

Outline

- Motivation and overview of the approach
- TCP/IP-friendly link layer principles
- Using FEC
- Test arrangements
- Test results
- Conclusions

End-to-end Approach with TCP/IP-Friendly Link Layer



DVB-RCS Satellite system:

RLSS : Return Link Sub-System
FLSS : Forward Link Sub System

SLACP – Satellite-Link Aware Communication Protocol

TCP/IP-friendly Link Layer Principles

- Reduce the residual packet-error rate but minimize the additional delay
- Use partially reliable in-order delivery of frames
- Combine ARQ and FEC to reduce residual packet loss rate
- Implement several logical channels with different QoS parameters (incl. priority)
  - Use of a single link channel is vulnerable to the head-of-line blocking as delay sensitive flows possibly must wait for (re)transmission of less urgent data to complete
  - Treat separately IP flows with different QoS requirements and turn off ARQ/FEC mechanisms for flows not benefiting from it

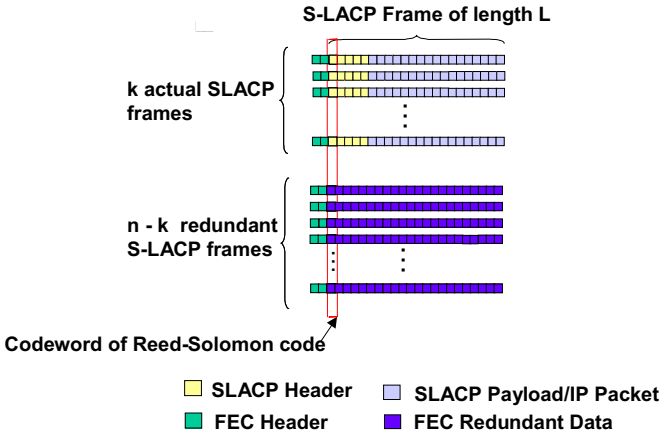
TCP/IP-friendly Link Layer Principles
(cont'd)

- **Avoid interaction with TCP timers →**

  Specific mechanisms to minimize extra delay due to error recovery:
  - **Cumulative acknowledgements with selective repeat requests**
  - **No link sender retransmission timers; ACK timer is used at link receiver to repeat the latest ACK if no new frames are arriving**
  - **Original transmission of frame with no FEC redundancy**
  - **Add incremental FEC redundancy to retransmitted frames**
  - **Higher priority for retransmissions and link ACKs**
- **Avoid unnecessary buffering at link layer**
  - **Only one unsent frame in the link send queue**
  - **Flow control between IP ←→ LLC and LLC ←→ MAC**

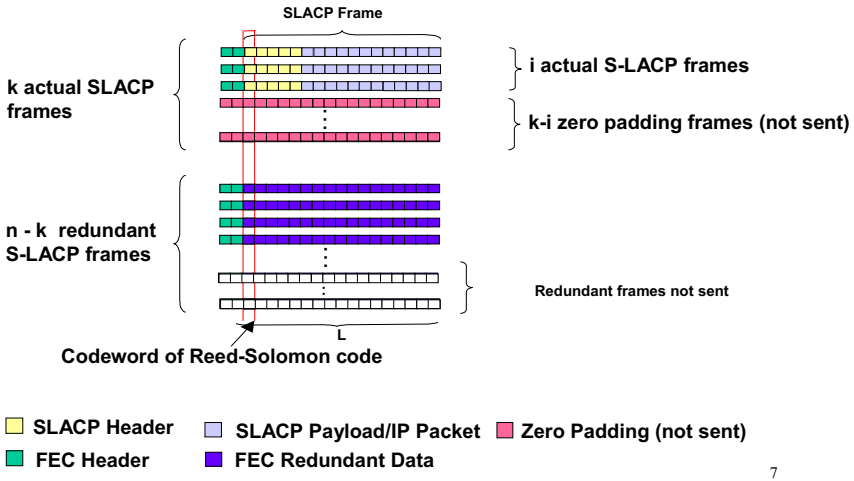
SLACP FEC Block Encoding

- A block of (retransmitted) frames is protected with FEC encoding
  - Reed-Solomon code RS(n,k): n = 255, k = number of data frames
    - $t \leq k$ lost frames can be recovered if t redundancy frames are received



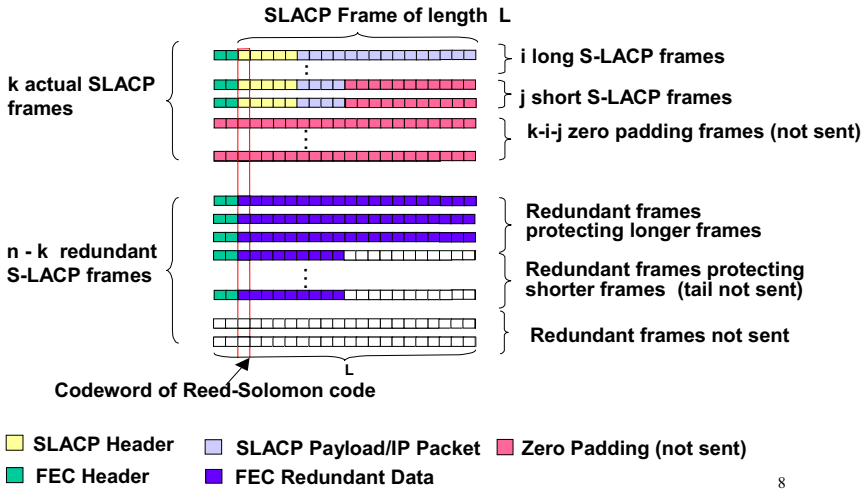
SLACP FEC Block Encoding

i < k actual data frames



SLACP FEC Block Encoding

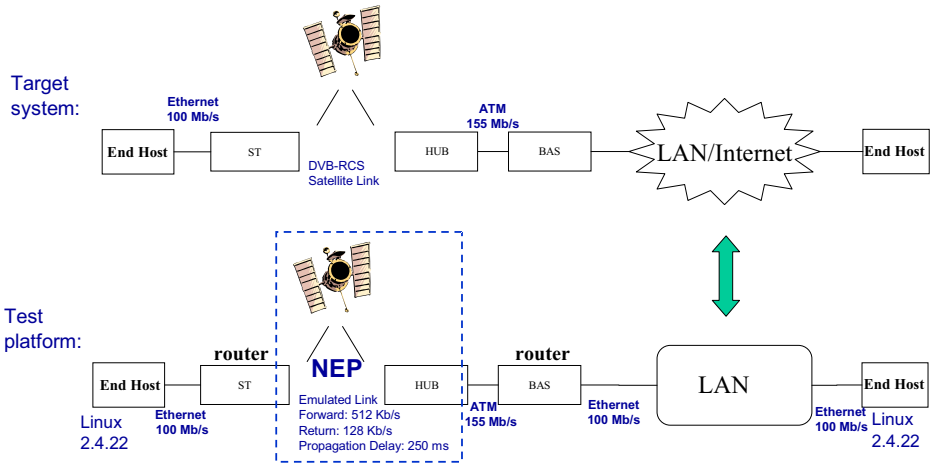
Optimization with different frame lengths



Test Arrangements

Network Topology

– NEP emulator is used emulate the satellite sub-system



TCP variants under study

- Reference TCP
  - Initial window of 1 segment
  - Congestion Control procedures as defined in RFC 2581
  - Fast Retransmit and Fast Recovery – New Reno (RFC 2582)
  - Delayed ACK threshold – 200 ms
  - SACK option (RFC2018) disabled

- Enhanced TCP (contains selected enhancements)
  - Initial window of 4 segments (RFC 3390)
    - Allows more rapid opening of congestion window during initial Slow-Start
  - SACK option (RFC2018) enabled; Conservative SACK algorithm (RFC 3517)
    - Allows TCP to recover more efficiently from multiple segment losses in a window

TCP variants under study

- Enhanced TCP (cont'd)
  - Window Scaling
    - Allows large windows on high bandwidth*delay product paths
  - Limited Transmit (RFC 3042)
    - Allows TCP sender to transmit new data for the first two dupacks
  - F-RTO - Forward-RTO Recovery (draft-ietf-tcpm-frto-02.txt)
    - Avoids unnecessary retransmissions after Spurious RTOs
  - DAASS - Delayed Acks After Slow-Start (RFC 2760)
    - Allows more rapid opening of congestion window during initial Slow-Start (each TCP segment is immediately acknowledged)
  - CBI - Control Block Interdependence reuse of ssthresh (applied to a reduced set of tests)
    - Mitigates Slow-start Overshoot

Error models

- Experimented a wide range of error models & controllable
- Two-state Markov model is applied to simulate GEO satellite failures and cause packet errors
- A set of Time Series define good and bad state lengths to be repeated in each test case

Error Model	Avg PER	Time between Bursts		Burst Lenght	
		Min	Max	Min	Max
None	0%	error-free link			
Low	1-1,5%	10 s	30 s	60ms	60ms
Medium	3-4%	5 s	15 s	100ms	300ms
High	5-6%	2 s	10 s	100ms	300ms
VHigh	7-8%	0 s	10 s	100ms	300ms
Huge	13-15%	0 s	7 s	100ms	100ms

Satellite Link Parameters

- Three different DAMA Mechanism on Return Link
  - CRA – Constant Rate Allocation
  - RBDC – Rate Based Dynamic Allocation
  - VBDC – Volume Based Dynamic Allocation

DAMA	Return	Forward
CRA	128 Kb/s	512 Kb/s
RBDC	128 Kb/s	512 Kb/s
VBDC	128 Kb/s	512 Kb/s

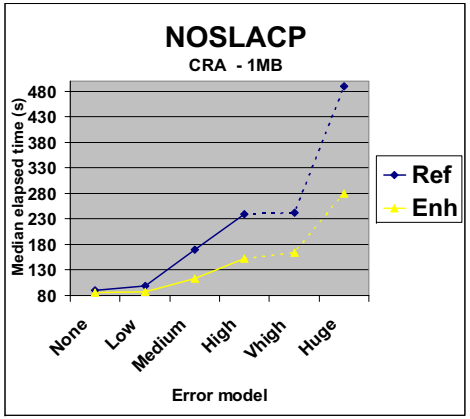
- **Propagation delay: 250 ms**
- **MTU Size: 1450 bytes**
- **MAC buffer size (ST):**
  - **RBDC/VBDC buffer: 28,8 kbytes / 14,4 kbytes (NOSLACP case / SLACP)**

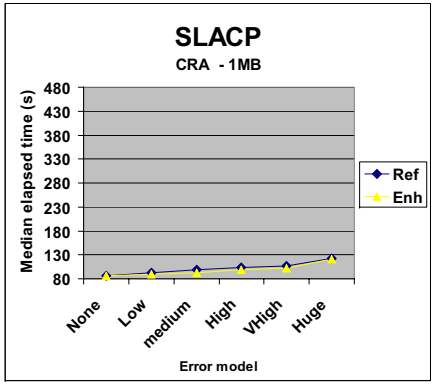
Workload

- Two types of workload over Return Link:
  - Single Unidirectional TCP Bulk Data Transfer
    - The transfer size is 1 MB
  - Four Parallel Unidirectional TCP Bulk Data Transfers
    - The transfer size is 250 KB for each TCP flow
    - Total transfer size equaling 1 MB
- Each test case is replicated 32 times

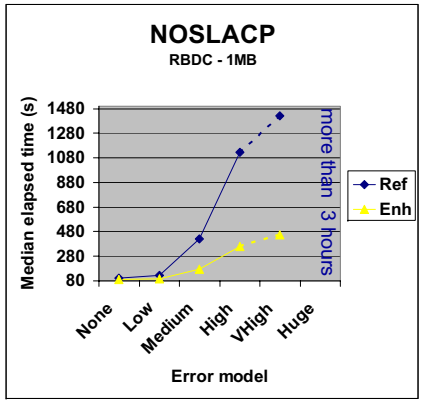
Test results

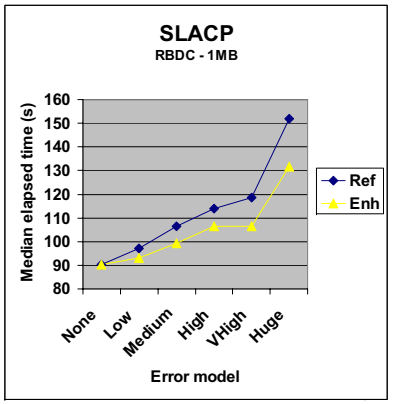
CRA



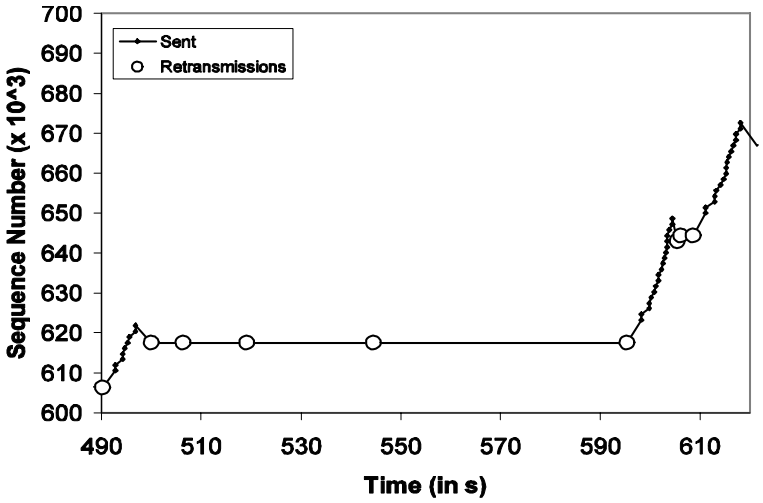


RBDC

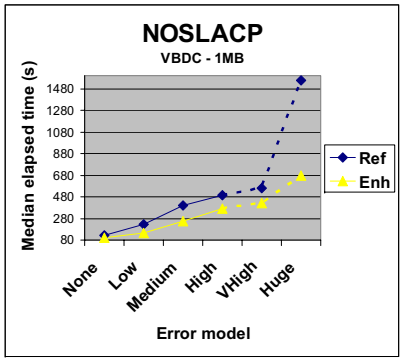


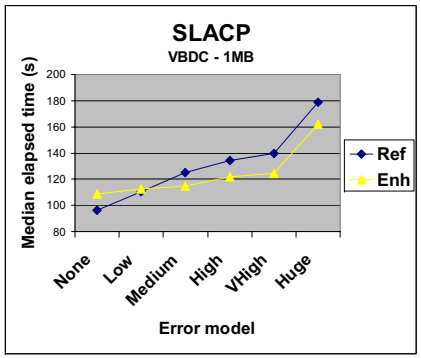


RBDC Interaction with TCP Error Recovery



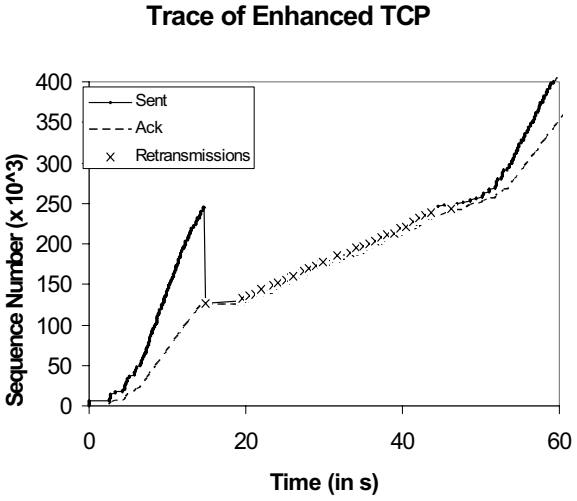
VBDC



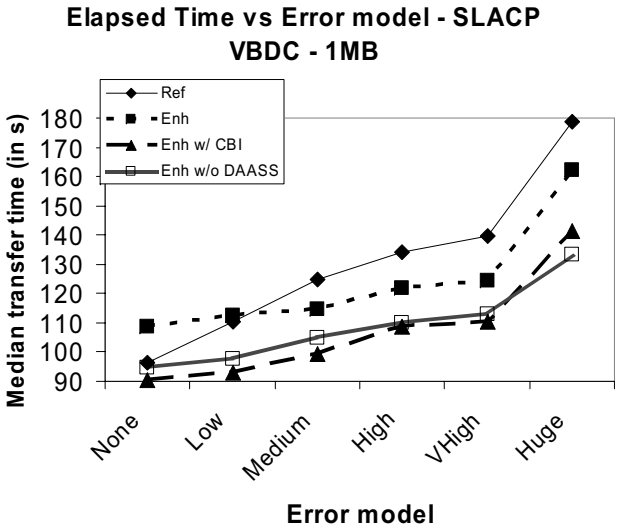


Slow-Start overshoot problem

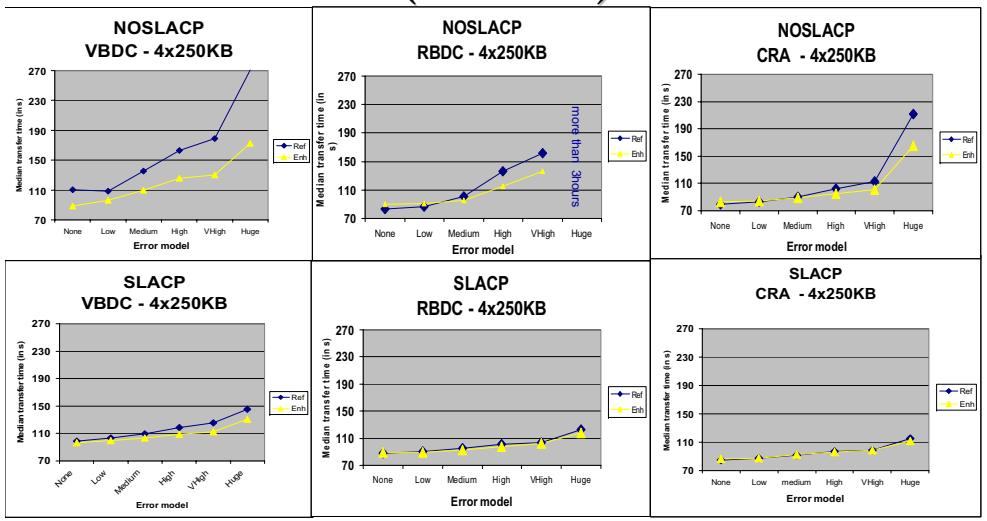
- The exponential opening of the congestion window results in overshooting the router queue in ST and loosing multiple segments
- Overshoot more serious for the Enhanced TCP as it is more aggressive



Attacking Slow-Start overshoot



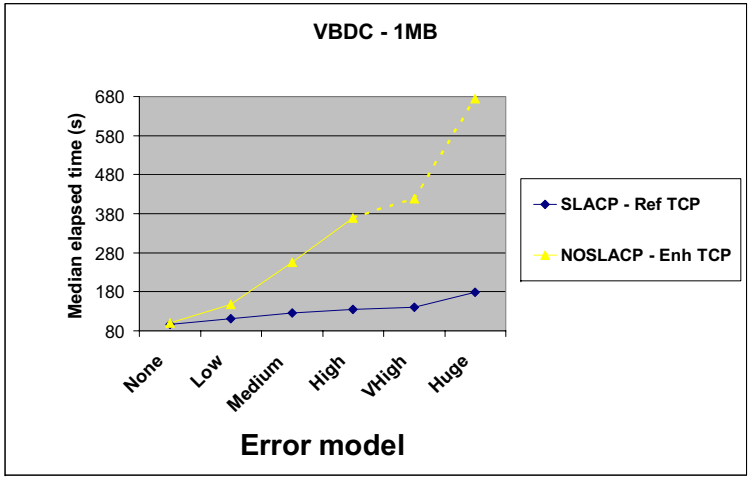
Test Results – 4 flows (4x250KB)



Conclusions

- Enhanced TCP performs better than Reference TCP
  - in particular when link layer error recovery with SLACP is not present
- SLACP is extremely useful when the link-error rate is high, improving TCP performance significantly
- Longer capacity allocation delay → slower error recovery
  - CRA yields best performance, while RBDC performs better than VBDC
  - With high error rates TCP performs very poorly with RBDC

Enhancing TCP is not enough



Conclusions (cont'd)

- Slow-start overshoot → Enhanced TCP performance decreased in some test cases
  - TCP CBI with ssthresh reuse is helpful to attack the problem
  - Acknowledging all segments immediately (DAASS) makes TCP too aggressive → disabling DAASS helps too
- Using four TCP flows to transfer the same amount of data yields better performance compared to the case of one TCP flow
  - makes transfer more aggressive