Theory of path characteristic analysis and applications in the satellite environment

Francesco Zampognaro

University of Rome "Tor Vergata"	Dpt. of Electronics Engineering	Prof. Michele Luglio
UCLA - University of California Los Angeles	Computer Science Dpt.	Prof. Mario Gerla

Satellite Multimedia Group
Rome, Feb 19 2009

1st PART

DEFINITIONS → TECHNIQUES →APPLICATIONS

End to end path characteristic

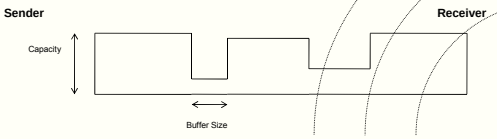
- L. Kleinrock ,S. S. Lam, F.A. Tobagi (70s)
- Network resources
  - Links
  - Buffers
- Network load

Goal

- Detect end-to-end path characteristic to optimize TCP transport protocol.
- Leverage on existing TCP mechanisms and state variables
- Measurement approaches:
  - Direct / indirect
  - Client-server approach
  - Sender side only
  - Statistical methods
  - Variable sized packets
  - Extra probing traffic
  - Carrier Sensing

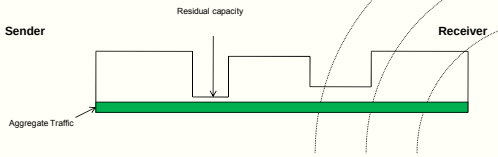
Channel Bottleneck link

- It is the narrowest link encountered during the path.
- In correspondence of the bottleneck link it is possible to define
  - Link capacity
  - buffer size



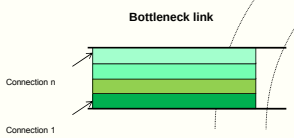
Residual capacity

- Is the amount of capacity on the bottleneck link unused by all the connections which cross the bottleneck link in a given time.



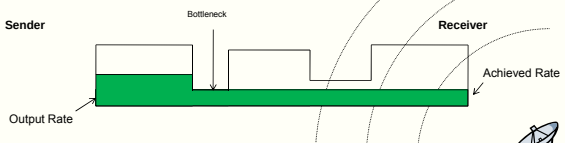
Channel fair share

- Capacity measure tends not to represent a good reference value for optimal TCP connection TX rate: if the channel is shared with several other connections, only a percentage of the bottleneck link capacity is for each of them.
- If TCPs are "fair" the channel is equally divided among all connections.



Achieved Rate

- Represents the effective bit rate delivered by the transport protocol



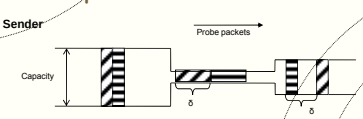
Warning

- Achieved rate, Capacity of the Bottleneck link, residual capacity and Fair Share represent different things, not necessarily in relation among each other and measured with different techniques.

DEFINITIONS → TECHNIQUES →APPLICATIONS

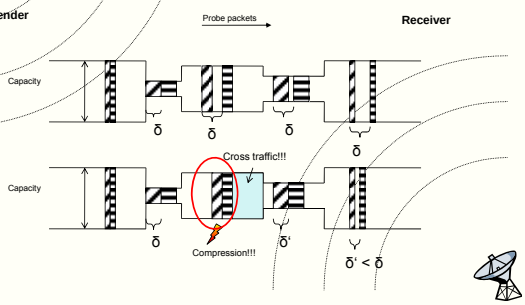
Bottleneck link capacity measure: Packet pair

- The idea is to send two consecutive packets (back to back) and measure the dispersion at receiver side (δ)

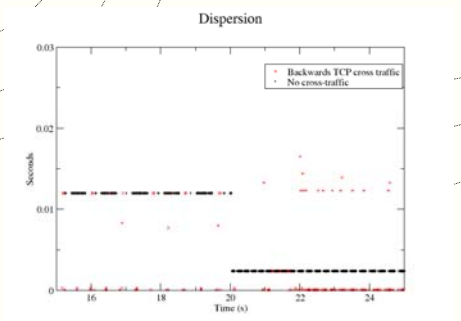


- $\tilde{C} = P_{size}/\delta$

Deterioration of δ measure



δ measure samples (NS2)



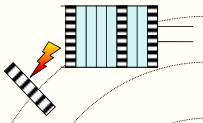
Rate estimation via packet trains

- Measure the Achieved Rate inferred by the flow of ACKs of a TCP connection sending a train of packets:
  - d_k=Amount of acknowledged data (bytes)
  - T= Sampling interval
  - R_k = Rate at time k-th

$$R_k = \frac{d_k}{T}$$

Bottleneck buffer size measure

- Use of RTT information to achieve buffer occupation and adjust rate of TCP
- The RTT slightly before a loss event can give information on the buffer dimension.
- $\tilde{B}_{size} = (RTT_{loss} - RTT_{min}) \cdot C$



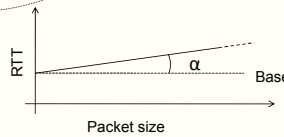
Deterioration of buffer size measures

- RTT_{min} is the condition where all queues are empty, which is difficult to occur
- RTT_{loss} can not represent the case with maximum bottleneck queue usage
- BufSize underestimation: loss on buffer due to cross traffic

DEFINITIONS → TECHNIQUES → APPLICATIONS

Van Jacobson's PathChar

- Sending increasingly sized packets. The RTT will be composed of a constant contribute and a transmission delay proportional to the packet size.

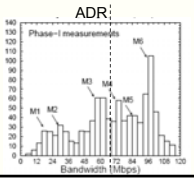


☺ α proportional to capacity!

☹ Needs several samples to filter out cross traffic!

Dovrolis' PathRate

- Statistical method (slow!!)
- Three phases approach:
  1. Send variable sized packet pair to discover main modes
  2. Send packet trains to find average dispersion rate (lower bound)
  3. Select mode with narrowest occurence



☹ Client-server approach, additional injected traffic

UCLA CapProbe

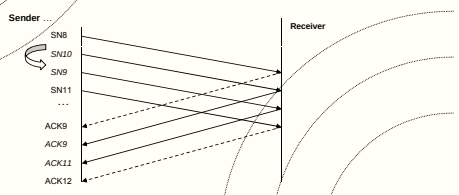
- The principle of sending two packets back-to-back (Packet Pair) and measure their reception dispersion is applied
- Only the packet pair experiencing the minimum RTT are considered for capacity estimation of the bottleneck link:
  - Consider only packets crossing the network through "assumingly" empty buffers

CapProbe implementation

- Sending TCP packets in pair: TCProbe
  - ☹ Delayed ACK can void the approach
  - ☹ Different size of sent packets (usually 1500 bytes) and received packets (usually 40 bytes)
  - ☹ Establishing RTT_{min} can be difficult

TCProbe details

- Counteract Delayed ACK sink:
  - Packet inversion:



- Filtering for ACKs compression and RTT_{min} detection are required!

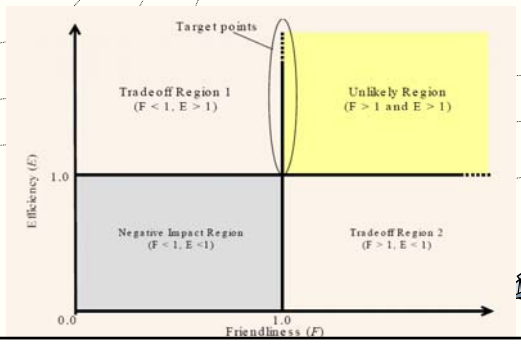
2nd PART

NEW TCP VERSIONS → TCP FOR SATELLITE CONNECTIONS

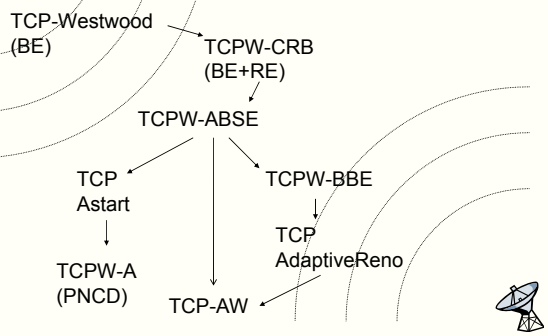
New TCPs

- When we obtain more details about the environment where our TCP is running what do we do?
  - Detect environment changes and adapt swiftly
  - Increase channel utilization and fairness
  - React better to losses not due to congestion
  - Generate statistics of the environment
  - Decide in case of multi-interfaces or multi paths connections (e.g. peer to peer) which link to use according to their ranking
  - Much more…
- **BUT:** keep an eye on friendliness towards reference TCPs!!! (in case they are present in the link)

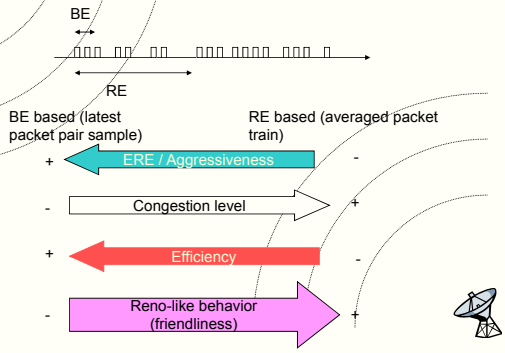
Efficiency/Friendliness trade off



UCLA Westwood Family Enhanced TCPs

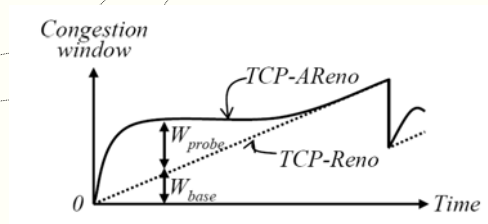


TCP Westwood BBE

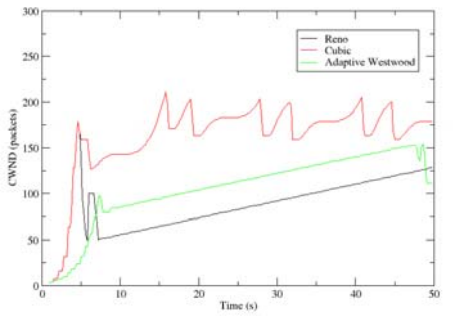


Enhanced TCPs

TCP Adaptive Reno – Congestion avoidance



Enhanced TCPs - Cubic



Other Enhanced TCPs

- Parametric
- BIC
- Highspeed
- Fast
- Hybla
- Compound
- LP
- Hamilton
- Veno
- Quickstart
- Illinois
- Much more!!

NEW TCP VERSIONS → TCP FOR SATELLITE CONNECTIONS

Satellite environment

- Challenging:
  - High latencies
  - Variable bandwidth and access delay (BoD)
  - Link asymmetry
  - Losses in mobility
  - Directional antennas
- Shared resources

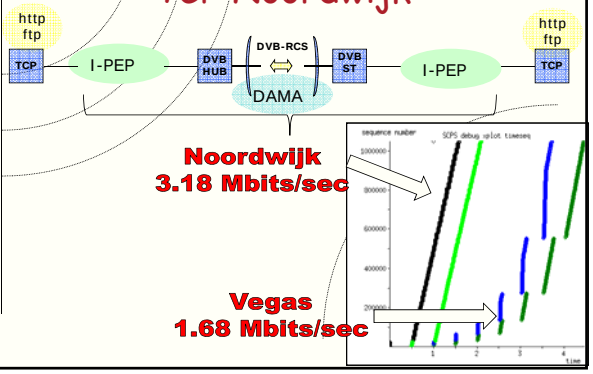
Cont'd

- Split architecture
- Security architectures
- Critical dimensioning of buffer queues for satellite links:
  - High values for optimal CWND≈BDP
- Problems when handover to other networks happens
  - Change in BDP product and buffers sizes

Case study 1 - TCPs for DVB-RCS

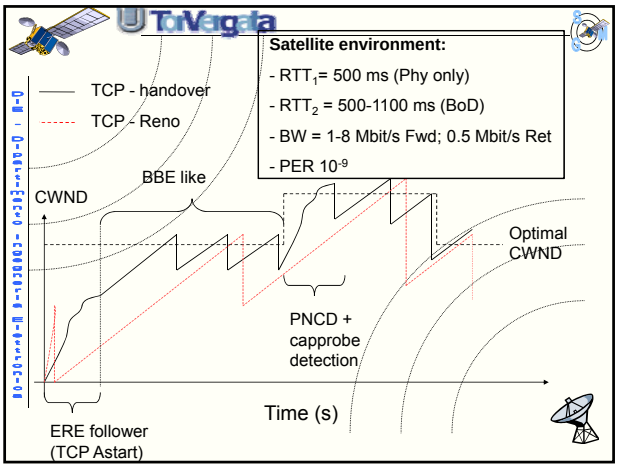
- TCP Noordwijk:
  - ESA project for I-PEP proxy split architecture transport protocol
  - Burst based transmission congestion control
  - Packet train analysis for channel capacity/share estimation
  - Congestion detection based upon buffer occupation
  - Implemented in Linux and NS2!

TCP Noordwijk



Case study 2 - TCP for Handover

- TCP handover detect proposal:
  - React to handover between terrestrial/UAV and satellite backup link
  - Avoid connection reset and partial data received loss
  - Do not affect intermediate hops or require any explicit (cross layer) signalling



Case study 3 – TCP Cross-layer

- Since the adversity of the satellite channel, cross-layer mechanisms could also be proposed:
  - Split architecture taking into account explicit signalling of link conditions
  - Security multi layer
  - Shadowing compensation
  - Smart Resource managment

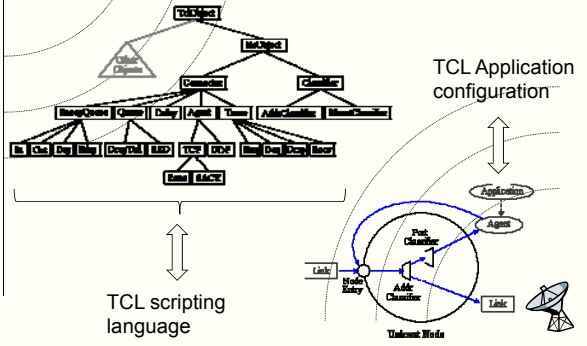
3rd PART

NS2 → EMULATION

NS2 – Network Simulator V2.x

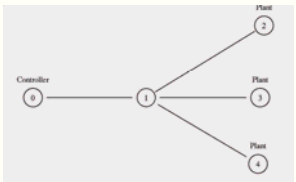
- Open source project:
  - Highly maintained
  - Academic standard for TCP/IP based simulations
  - Strong code contribution
- C++ core functionalities:
  - Simulation speed
  - Object oriented
- TCL scripting language
  - Simulations are written in TCL (text files)
  - No need to recompile the core to create multiple batch simulations with values bracketing
  - Core complexity hidden to the user

NS2 block diagram



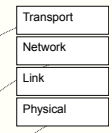
NS2 visual output

- NAM takes as input simulation logs to generate visual outlook of the simulation (packets are moving around on the links):



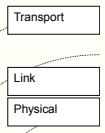
NS2 geostationary Sat Channel

- Transport layer can be any available NS2 end to end Agent (TCP, UDP, etc.)
- Network layer offers direct routing among all terminals via the satellite
- Simple aloha link layer with Queuing.
- Physical layer takes into account position of terminals (lat/long) and the bent pipe satellite for communication delay

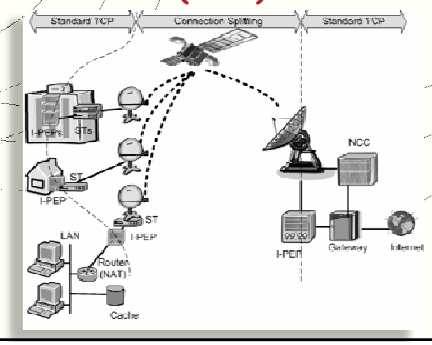


NS2 satellite extensions (1/2)

- TCP splitting at Transport Layer: using ad-hoc TCP protocol on the satellete link (TCP-Noordwijk)
- DVB-RCS DAMA return channel resources sharing among several terminals and central NCC: http://www.tlcsat.uniroma2.it/DAMA/
- Error model for the physical layer to resemble real DVB satellite transmission (taking into account link budget, framing, etc.)

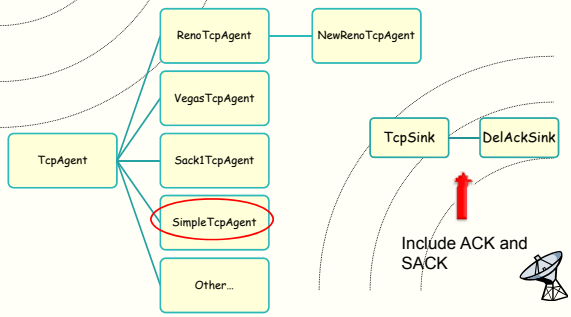


NS2 satellite extensions (2/2)



NS2 TCP hierarchy

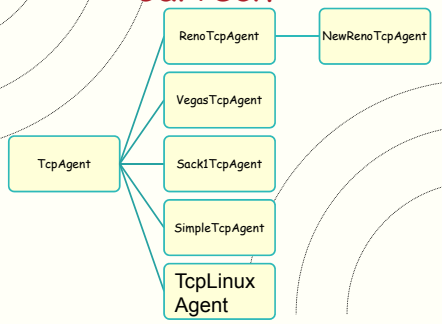
http://www-rp.lip6.fr/ns-doc/ns226-doc/html/index.htm - DoxyGen



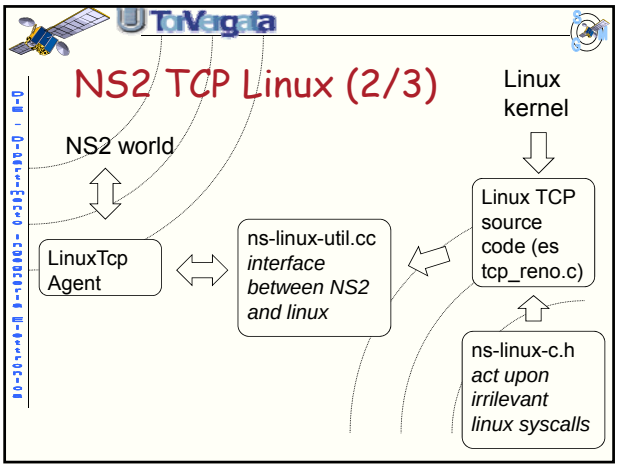
NS2 Simple TCP

- Constructor:
  SimpleTcpClass() : TclClass("Agent/TCP/SimpleTcp") {}
- Send a message (called by the application):
  void SimpleTcpAgent::sendmsg(int bytes, const char*)
- Receive ACKs:
  void SimpleTcpAgent::recv(Packet * pkt, Handler *)
- Interface with TCL scripts:
  int SimpleTcpAgent::command(int argc, const char*const* argv)

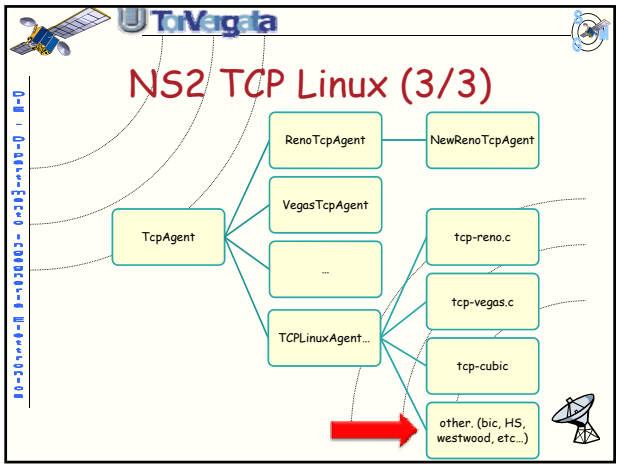
NS2 TCP Linux (1/3) - CalTech



NS2 TCP Linux (2/3)



NS2 TCP Linux (3/3)



NS2 Linux TCP Naïve

Functions for TCP congestion control in Linux:

Optional		Mandatory
init() release() rtt_sample() set_state() min_cwnd()	pkts_acked() cwnd_event() undo_cwnd() get_info() name()	ssthresh() cong_avoid()

TCP Naïve in NS2 Linux:

- Congestion control - void tcp_naive_reno_cong_avoid()
  Window tuning in response to ack received (e.g. slow start and congestion avoidance for TCP Reno)
- Ssthresh reduction on loss –
  u32 tcp_naive_reno_ssthresh()
- CWND reduction on loss –
  u32 tcp_naive_reno_min_cwnd()

Example of loss handling

```
/* ssthreshold should be half of the congestion window after a loss */
u32 tcp_naive_reno_ssthresh(struct tcp_sock *tp)
{
  return max(tp->snd_ssthresh/2 , 2);
}
/* Set congestion window to ssthresh after loss */
u32 tcp_naive_reno_min_cwnd(struct tcp_sock *tp)
{
  return tp->snd_ssthresh;
}
```

Example window growth

```
void tcp_naive_reno_cong_avoid(struct tcp_sock *tp, u32
    ack, u32 rtt, u32 in_flight, int flag)
{
if (tp->snd_cwnd < tp->snd_ssthresh) {
    tp->snd_cwnd = tp->snd_cwnd + 1;
} else {
if (tp->snd_cwnd_cnt >= tp->snd_cwnd) {
    tp->snd_cwnd += 1/tp->snd_cwnd;
    }
}
```

←Exponential slow-start:

←Linear Congestion avoidance

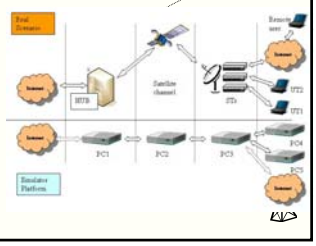
NS2 - Experiences

- Modify transmission part to handle packet swap: easy in NS2 not trivial for Linux wrapper
- Interface NS2 simulation with real traffic from ethernet cards
- NS3 is on the way… phyton scripting, real time, rewritten from scratch!
- Move NS2 Linux code to a real emulated DVB-RCS environment!

NS2 → **EMULATION**

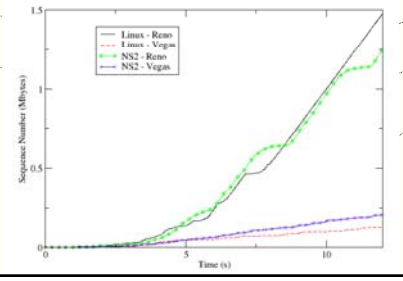
DVB Emulator platform

- In-house emulation of a DVB-RCS platform with DAMA resource allocation using Linux PCs:
  - Test new transport protocols and applications
  - Security architecture implementation
  - Real Traffic and OS



DVB Emulator: comparison vs NS2

- RBDC (rate based) resource allocation



Conclusions

- TCP is still a live research topic (since V. Cerf's RFC674, dated 1974) and newer congestion control algorithms are constantly introduced
- TCP can use additional information to perform better in several conditions
- The satellite systems can be seen as a new and challenging environment for TCP
- Simulation and Emulation can give strong support to research and development of protocols, validating ideas and consolidating the results achieved

Thanks for your attention!!!

Questions ?

zampognaro@ing.uniroma2.it
http://www.tlcsat.uniroma2.it