Project Proposal
PCC Vivace: Online-Learning Congestion Control

Girish Singh Thakur (2025MCS2973), Nitish Kumar (2025MCS2100)

1 Paper Details

- **Paper Title:** PCC Vivace: Online-Learning Congestion Control
- **Authors:** Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, P. Brighten Godfrey, Michael Schapira
- **Conference:** 15th USENIX Symposium on Networked Systems Design and Implementation (NSDI '18)
- **Publication Date:** April 2018
- **Location:** Renton, WA, USA
- **Number of Citations:** 310+ (as of 2025)
- **Pages:** 343-356
- **ISBN:** 978-1-939133-01-4

2 Project Title

Adaptive Multi-Objective Congestion Control: Extending PCC Vivace for Heterogeneous Networks and Application-Aware Optimization

3 Problem Statement

Traditional congestion control protocols, including TCP variants (Reno, Cubic, BBR) and their derivatives, exhibit fundamental limitations that prevent them from achieving optimal network utilization:

3.1 Architectural Limitations

1. **Reactive Heuristics:** TCP-based protocols use predefined control rules triggered by specific network events. For example, TCP Reno halves the congestion window upon packet loss, while TCP Illinois adjusts based on RTT changes using the formula $w \leftarrow f(\Delta RTT) \cdot w$. These hardcoded responses cannot adapt to diverse network conditions.
2. **Suboptimal Performance Trade-offs:** Loss-based algorithms (TCP Reno/Cubic) maintain high throughput but cause bufferbloat, resulting in latency spikes of 100-500ms. Delay-based algorithms (TCP Vegas, Copa) achieve low latency but significantly underutilize available bandwidth when competing with loss-based flows.

3. **Slow Convergence:** When flows dynamically join or leave, TCP requires tens of seconds to converge to fair bandwidth allocation due to conservative AIMD dynamics. In data center incast scenarios with n senders, convergence time scales poorly.

4. **Fixed Optimization Objectives:** Existing protocols optimize for implicit objectives (e.g., maximize throughput subject to AIMD fairness), but cannot adapt to varying application requirements or network policies.

3.2 Prior Work Limitations

Recent approaches attempted to address these issues but fell short:

- **PCC Allegro:** First application of online learning to congestion control but suffered from overly long convergence times (10+ seconds) due to cautious exploration strategy
- **BBR:** Model-based approach that estimates bottleneck bandwidth and RTT, but performance degrades when models don't match reality (e.g., in cellular networks with variable capacity)
- **Remy:** Uses offline RL to generate lookup tables, but requires prior knowledge of network parameters and cannot adapt to conditions outside its training distribution

4 Proposed Solutions/Extensions

4.1 Original PCC Vivace Approach

Vivace reformulates congestion control as an online convex optimization problem:

Core Algorithm:

1. Divide time into Monitor Intervals (MIs) of duration T (typically 1-2 RTTs)
2. Within each MI, test multiple sending rates $r_1, r_2, \ldots, r_k$
3. Measure performance metrics: throughput T_i, latency L_i, loss rate l_i
4. Compute utility: $U_i = f(T_i, L_i, l_i)$ where f balances performance objectives
5. Apply gradient ascent: $r_{t+1} = r_t + \alpha \cdot \nabla U(r_t)$
6. Use momentum and rate limiting for stability

Key Innovation - Utility Function Design:

$$U = T \cdot S(L) - \lambda \cdot T \cdot l$$

where $S(L)$ is a sigmoid function penalizing latency inflation and λ controls loss sensitivity.

Theoretical Guarantees: The paper proves that Vivace achieves sublinear regret $O(\sqrt{T})$ under the online gradient descent framework, ensuring convergence to optimal sending rate even in dynamic environments.

4.2 Proposed Extensions

Extension 1: Hierarchical Multi-Objective Learning Framework

Current Vivace uses a single utility function for all flows. We propose a hierarchical approach:

- **Traffic Classification:** Use the first few packets to classify flows into categories (bulk transfer, real-time communication, streaming, interactive) based on packet size distribution and inter-arrival times
- **Utility Function Bank:** Maintain specialized utility functions:

$$U_{bulk} = \alpha_1 T - \alpha_2 l \quad \text{(maximize throughput)}$$
$$U_{realtime} = \beta_1 T \cdot S(L_{max}) \quad \text{(minimize tail latency)}$$
$$U_{streaming} = \gamma_1 T \cdot I(T > T_{min}) - \gamma_2 Var(T) \quad \text{(stable rate)}$$

- **Meta-Learning:** Train a meta-controller using reinforcement learning to select the appropriate utility function based on observed traffic patterns and application feedback
- **Evaluation:** Compare application-level QoE metrics (video stall rate, game latency percentiles, file transfer completion time) against single-utility Vivace

Extension 2: Loss-Resilient Vivace for Wireless Networks

Wireless networks exhibit random loss unrelated to congestion (typical rates: 1-5% for WiFi, 0.5-2% for LTE). We propose:

- **Loss Differentiation:** Implement statistical tests to distinguish loss types:
  - Congestion loss: correlated with RTT increase, bursty patterns
  - Random loss: uncorrelated with RTT, uniform distribution

  Use cross-correlation analysis: $\rho(loss, RTT)$ over sliding window
- **Adaptive Utility Weights:** Dynamically adjust λ in utility function based on estimated random loss rate p:

$$\lambda_{effective} = \lambda \cdot (1 - p) + \lambda_{wireless} \cdot p$$

  where $\lambda_{wireless} < \lambda$ to reduce penalty for wireless loss
- **Probing Strategy:** Use occasional higher-rate probes to detect capacity increases in cellular networks (handoff, improved channel conditions) without excessive loss
- **Implementation:** Integrate with 3GPP radio measurements when available for enhanced context

Extension 3: Explicit Fair Queuing via Distributed Learning

Enable multiple Vivace flows to coordinate for faster convergence and explicit fairness:

- **Implicit Signaling:** Flows infer presence of competing traffic by observing utility gradient direction changes. When ∇U frequently alternates sign, it indicates contention
- **Cooperative Exploration:** Implement alternating exploration where flows take turns probing higher rates, reducing collision overhead

- **Fairness-Augmented Utility:** Add penalty term for deviation from fair share:

$$U_{fair} = U_{original} - \mu \cdot |r_i - r_{fair}|$$

where r_{fair} is estimated from aggregate throughput measurements

- **Virtual Queue Tracking:** Each sender maintains estimate of bottleneck queue occupancy using delay measurements, enabling coordination without explicit signaling

Extension 4: Multipath Rate Allocation
Extend Vivace to scenarios where traffic can split across multiple paths (MPTCP, MPQUIC):

- **Joint Optimization:** Treat multipath rate allocation as multi-armed bandit problem where each path is an arm with time-varying reward (utility)
- **Path Utility:** Compute per-path utility U_p and use softmax allocation:

$$r_p = R_{total} \cdot \frac{e^{U_p/\tau}}{\sum_{p'} e^{U_{p'}/\tau}}$$

where τ controls exploration-exploitation tradeoff

- **Correlation Handling:** Account for paths that share bottleneck links by learning correlation structure through observation

5 Evaluation Metrics

5.1 Primary Performance Metrics

1. **Throughput:** Average goodput (application-layer throughput) in Mbps, measured over entire flow duration and in steady state (after convergence)
2. **Latency:**
   - Mean RTT during data transfer
   - 95th and 99th percentile RTT (tail latency)
   - Latency inflation: $\frac{RTT_{loaded}}{RTT_{baseline}}$
3. **Loss Rate:** Fraction of packets lost, computed as: $loss\_rate = \frac{lost\_packets}{sent\_packets}$
4. **Convergence Time:** Time to reach within 5% of fair bandwidth allocation when:
   - New flows join existing traffic
   - Available bandwidth changes suddenly
   - Flows terminate
5. **Fairness:** Jain's fairness index for n flows:

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

where x_i is throughput of flow i. Perfect fairness: $J = 1$

5.2 Secondary Metrics

1. **TCP-Friendliness Ratio:** $\frac{Throughput_{Vivace}}{Throughput_{TCP}}$ when competing against legacy TCP flows. Acceptable range: 0.8-1.2
2. **Buffer Occupancy:** Average queue length at bottleneck routers (packets or bytes)
3. **Utility Score:** Aggregate utility over time: $\int U(t)dt$
4. **Rate Stability:** Coefficient of variation: $CV = \frac{\sigma_{rate}}{\mu_{rate}}$. Lower is more stable.
5. **Application QoE:**
    - Video: Stall ratio, average bitrate, bitrate switches
    - Gaming: 99th percentile latency, jitter
    - Bulk: Flow completion time (FCT)

5.3 Experimental Setup

Testing Environments:

- **Emulation:** Mahimahi for controlled replay of network traces (LTE, WiFi, satellite), Mininet for data center topologies
- **Simulation:** NS-3 for large-scale scenarios (100+ flows), parameter sweeps
- **Real Networks:**
    - WAN transfers between geographically distributed servers
    - Cellular networks (4G/5G) with mobile clients
    - Campus WiFi with varying load

Comparison Baselines:

- Loss-based: TCP Cubic, Compound TCP
- Delay-based: TCP Vegas, Copa
- Model-based: BBR, BBRv2
- Learning-based: PCC Allegro, Remy (when applicable), Aurora (DRL-based)

Test Scenarios:

1. Static bottleneck (10-100 Mbps links, 10-200ms RTT)
2. Dynamic bandwidth (sudden capacity changes $\pm 50\%$)
3. Competing traffic (1-32 flows, homogeneous and heterogeneous protocols)
4. Random loss (0-5% non-congestion loss)
5. Buffer size variations (shallow: 5 packets, deep: 1000 packets)
6. Real traces from Pantheon dataset

6 Key Takeaways

1. **Paradigm Shift from Reactive to Learning-Based Control:** Vivace demonstrates that treating congestion control as a continuous optimization problem with online learning outperforms decades of manually-tuned TCP heuristics. The online gradient descent framework provides both practical performance and theoretical convergence guarantees.

2. **Utility Functions as First-Class Abstractions:** By explicitly defining a utility function that captures performance objectives, Vivace provides a principled way to balance conflicting goals (throughput vs. latency vs. loss). This abstraction is more flexible than implicit objectives embedded in protocol rules.

3. **Empirical Performance Gains:** Vivace achieves 2-3x throughput improvement over TCP Cubic in many scenarios, maintains 95th percentile latency below 50ms (vs. 200-500ms for TCP), and converges 10x faster (sub-second vs. tens of seconds). These improvements are consistent across diverse network conditions.

4. **Bufferbloat Mitigation Without AQM:** Unlike TCP which requires Active Queue Management (CoDel, FQ-CoDel) to prevent excessive queuing, Vivace's utility function naturally penalizes latency inflation, keeping queues small even with deep buffers. This is crucial for deployment scenarios where router configuration is not controllable.

5. **Theoretical Foundations Matter:** The online convex optimization framework provides regret bounds ($O(\sqrt{T})$) that guarantee convergence properties. This theoretical grounding distinguishes Vivace from purely empirical DRL approaches whose behavior may be unpredictable outside training distributions.

6. **Practical Deployment Feasibility:** Vivace has been successfully implemented in real systems (QUIC transport, Linux kernel module) and integrated into Pantheon evaluation platform. This demonstrates that learning-based approaches can be practical, not just theoretical exercises.

7. **Open Challenges for Extension:** While Vivace performs excellently in many scenarios, opportunities exist for improvement in wireless networks (loss differentiation), application-specific optimization (adaptive utility functions), multipath scenarios (joint rate allocation), and explicit fairness mechanisms (distributed coordination). These extensions represent natural evolution of the framework.

8. **Broader Implications:** Vivace's success suggests that other network protocols (routing, load balancing, resource allocation) could benefit from similar learning-based reformulations. The key insight is to define appropriate utility functions and leverage online optimization rather than designing complex rule-based systems.