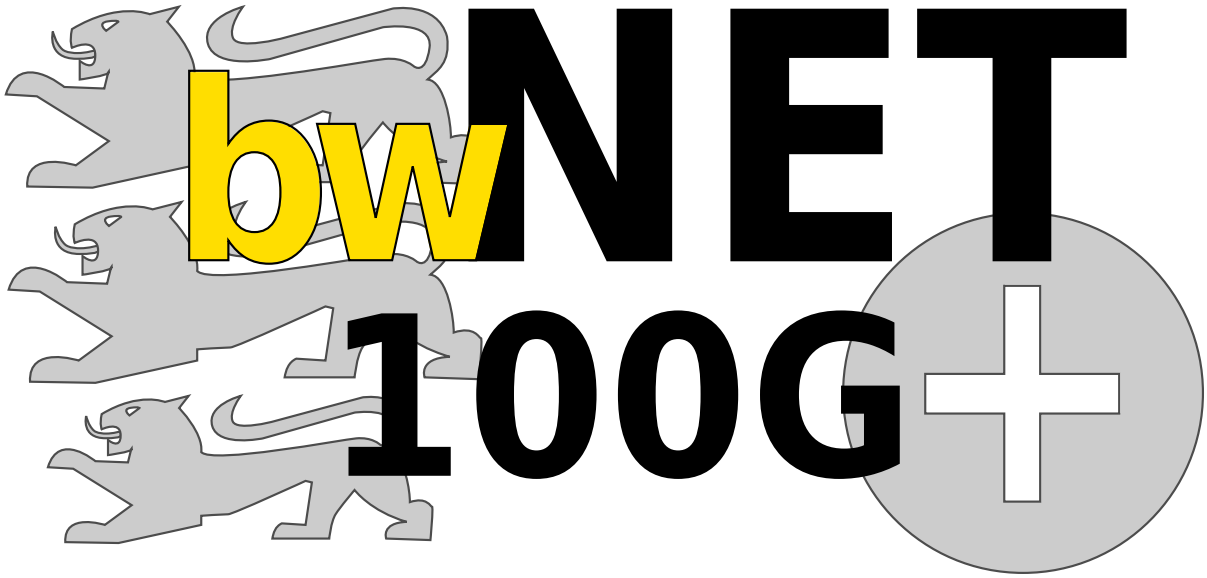
Begleitprojekt zum 100G-Ausbau des Landeshochschulnetzes BelWü



Forschung und innovative Dienste für ein flexibles 100G-Netz in Baden-Württemberg

Optimierungsleitfaden: 10G Tuning Guide

Andre Hafner[***], Dino Rezes[**], und Philipp Wolter[*]

[*]Karlsruher Institut für Technologie, Steinbuch Computing Centre
[**]Universität Tübingen, ZDV
[***]Universität Ulm, BelWü

18. Oktober 2017

Koordinator und Ansprechpartner ALWR:
Prof. Dr. Stefan Wesner <stefan.wesner@uni-ulm.de>

Wissenschaftliche Koordination:
Prof. Dr. Martina Zitterbart <zitterbart@kit.edu>

Inhaltsverzeichnis

1 Einleitung

Dieser "Tuning-Guide" genannte Optimierungsleitfaden soll Betreibern von 10-Gigabit-Infrastruktur Informationen an die Hand geben, wie die Bandbreite in verschiedenen Situationen in wenigen Schritten bestmöglich nutzbar gemacht wird. Unsere Erfahrungen fußen dabei auf systematische Benchmarks unter Variation zahlreicher Parameter, die wir bei der Leistungsbestimmung der ersten Technologieiteration (10*10G-Technik) des Landesprojekts bwNET100G+ vornehmen konnten. Zum Zwecke der Nachvollziehbarkeit legen wir nach einer kurzen Beschreibung der Testinfrastruktur den Versuchsaufbau dar und fokussieren uns dann auf die Optimierungsparameter im Endsystem. In der Zusammenfassung beleuchten wir abschließend die Ergebnisse, insbesondere die Unterschiede zwischen Optimierung im LAN- und WAN-Umfeld. Wir wünschen dem Leser gutes Gelingen beim Optimieren.

2 Hintergrund

In den Werkseinstellungen gängiger Unix-Distributionen, wie z.B. Ubuntu, wird der Augenmerk darauf gelegt, dass diese für Bandbreiten von 1GBit/s und darunter, sowie kurze Latenzen, die Bandbreite möglichst vollständig ausgeschöpfen. Aus Sicht der Distributoren ist dies ein nachvollziehbares Verhalten, da höhere Bandbreiten vor allem im Heimbereich heute überwiegend keine Anwendung finden.

Für die Anwendung in Netzen mit 10GBit/s und darüber, wie sie in Netzen von Hochschulen schon heute üblich sind, müssen einige Einstellungen angepasst werden um diese Bandbreite ausschöpfen zu können. Ursache ist, dass in einem festen Zeitintervall bei höherer Datenrate mehr Daten übertragen werden sollen, was die voreingestellten Puffer und die Maximum Transmission Unit nicht zulassen. Die Puffer müssen größer sein, damit die Empfangs- und Sendefenster der Layer-4-Protokolle der Bandbreite entsprechen wachsen können. Eine niedrige Maximum Transmission Unit (MTU) impliziert höheren Overhead für gegebene Nutzdaten und damit eine relativ hohe erforderliche Rechenleistung. Insbesondere bei älteren Servern stellt dies ein Problem bei der Nutzung von 10 Gbit/s dar. Diese Probleme werden im Folgenden evaluiert.

Für unsere Messungen wurden Rechner mit jeweils zwei Intel E5-2630v3 CPUs, 128GB DDR4 PC2133 Reg. ECC RAM, sowie einem X10DRW-i Mainbord von Supermicro verwendet. Die eingebaute Netzwerkkarte war die Intel X710-DA2 oder -DA4. Diese unterscheiden sich nur in der Anzahl der Ports, bei gleichem Chipsatz. Sie verfügen im Gegensatz zu älteren Broadcom QLogic-basierten Karten über mannigfaltige Offloading-Fähigkeiten. Das verwendete Betriebssystem war ein Ubuntu 14.04 (64Bit) mit Kernelversion 4.4.0-31, sowie IPerf in Version 3.0.11. Der Netzwerkkartentreiber i40e war in Version 1.4.25-k installiert und die Firmware der Netzwerkkarte auf auf dem Versionsstand 4.53.

3 Versuchsaufbau

Zur Nachvollziehbarkeit der Messdaten unterscheiden wir zwei Versuchsaufbauten. Versuchsaufbau "LAN" steht prototypisch für einen Server-zu-Server-Aufbau mit geringer Latenz. Das "WAN"-Setup umfasst ein Weitverkehrsnetz von ca. 536km Länge und einer Latenz von 6ms.

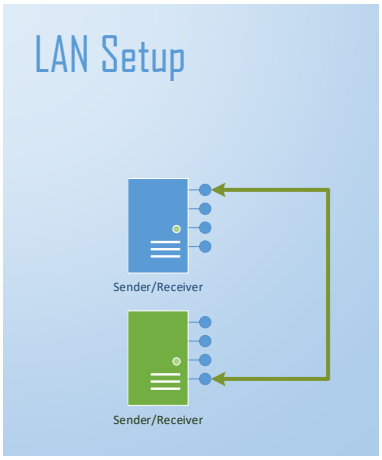


Abbildung 1: Grundsätzliches Schema des LAN-Aufbaus

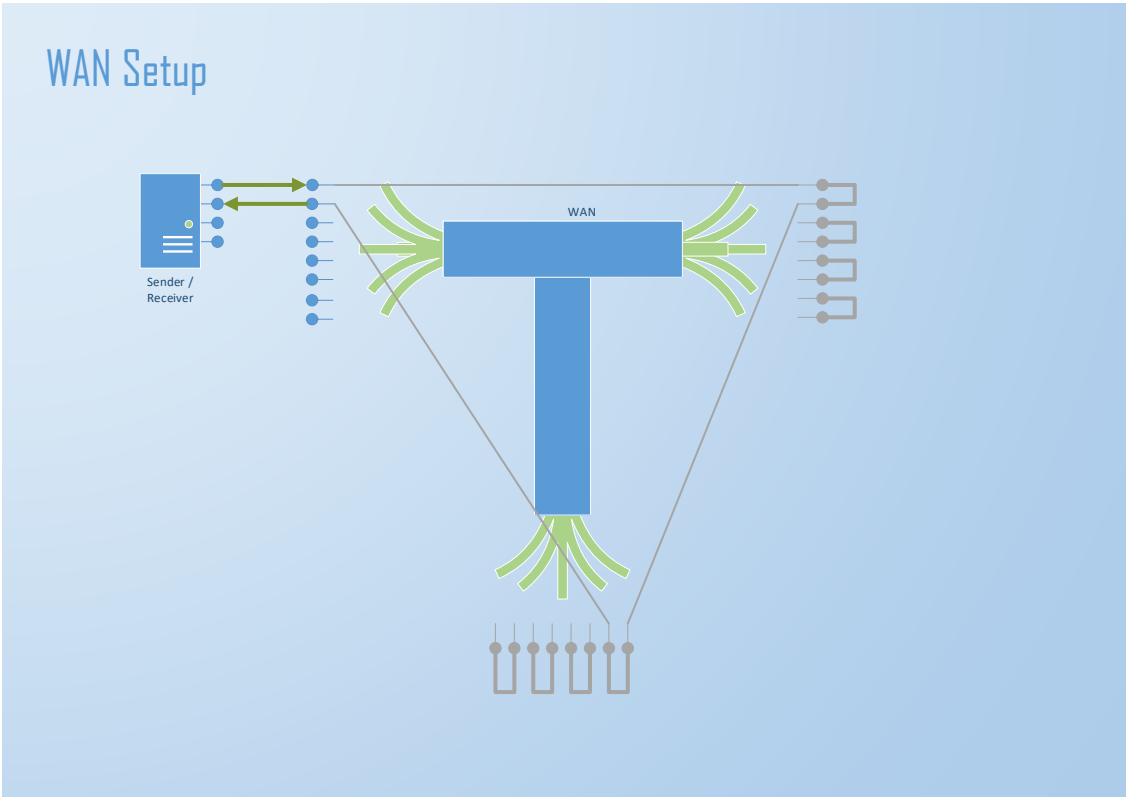


Abbildung 2: Grundsätzliches Schema des WAN-Aufbaus

4 Messungen und Ergebnisse

Im Folgenden werden die Messergebnisse mit und ohne Optimierung dargestellt. Die optimierten Kernelparametern resultieren zum einen aus der ebenfalls im Rahmen des Projekts entstandenen Bachelor-Abschlussarbeit [1] von Leonard Bradatsch, zum anderen aus Versuchsreihen im Vorfeld der Benchmarks.
Zudem werden die korrespondierenden Einstellungen aufgelistet. In jedem Fall wurde vom Kernel-Tuning ab stets die MTU und TXQL (TX Queue Length) variiert, um deren Einfluss unabhängig vom Tuning für LAN- und WAN-Umgebungen erkennbar zu machen.

4.1 Benchmarks ohne Optimierung

Um die Ausgangssituation kennen zu lernen wurden Benchmarks mit den in Ubuntu 14.04 voreingestellten Kernelparametern gemacht. Diese sind in Listing 1 dargelegt.

4.1.1 Kernelparameter ohne Optimierung

Listing 1: Ubuntu Defaultparameter

```
1  # buffers
2  sysctl: name=net.core.rmem_max value=212992
3  sysctl: name=net.core.wmem_max value=212992
4  # autotuning TCP buffer limit
5  sysctl: name=net.ipv4.tcp_rmem value="4096 87380 6291456"
6  sysctl: name=net.ipv4.tcp_wmem value="4096 16384 4194304"
7  # length of the processor input queue
8  sysctl: name=net.core.netdev_max_backlog value=1000
9  sysctl: name=net.ipv4.tcp_congestion_control value=cubic
10 sysctl: name=net.ipv4.tcp_mtu_probing value=0
11 sysctl: name=net.ipv4.tcp_timestamps value=0
12 sysctl: name=net.ipv4.tcp_low_latency value=0
```

Im Benchmark soll zwischen der Situation im lokalen Netzwerk mit kurzen Latenzzeiten und der Situation in Weitverkehrsnetzen unterschieden werden. Wie in Abbildung 3 zu sehen ist, werden in lokalen Netzen bereits ohne Tuning werte von über 9.3 GBit/s erreicht. Es fällt auf, dass der Einfluss der MTU hier deutlich ins Gewicht fällt, wohingegen der Einfluss der TXQL vernachlässigt werden kann.
Abbildung 4 stellt die Ausgangssituation in Weitverkehrsnetzen dar. Wie auch bei der Situation im lokalen Netz werden mit einer MTU von 9000 Bytes Datenraten erzielt, die bereits nah am theoretischen Maximum sind. Während eine MTU von 1500 Bytes in lokalen Netzen nur zu einem geringfügigen Geschwindigkeitseinbruch führt, ist diese Effekt in Weitverkehrsnetzen deutlicher. Es werden hiermit nur noch 5,5 bzw. 5,7 GBit/s erreicht.

4.2 Benchmarks mit Optimierung

Als Ansatz für weiterreichende Optimierungen sind die Puffergrößen des Kernels zu sehen, die für die Datenmengen die in 10G-Netzwerken auflaufen können klein ausgelegt sind. Als Richtwert gilt hier[2][3]

$$\text{Puffergröße} = 2 \cdot \text{Bandbreite} \cdot \text{Verzögerung}.$$

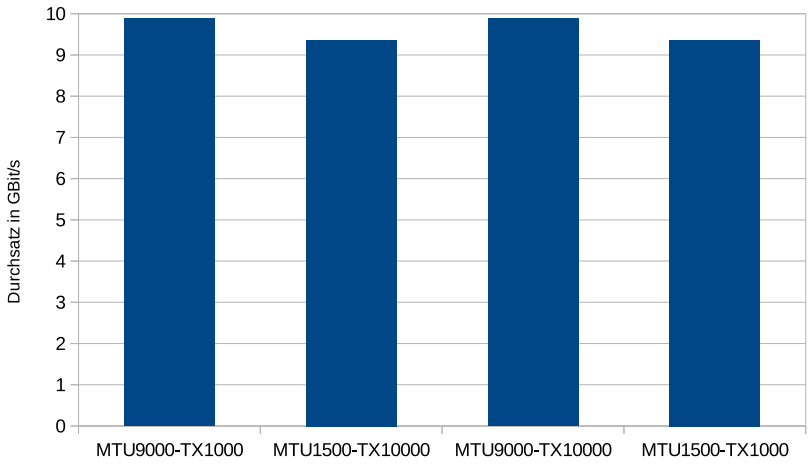


Abbildung 3: Benchmarks der Ausgangssituation im LAN

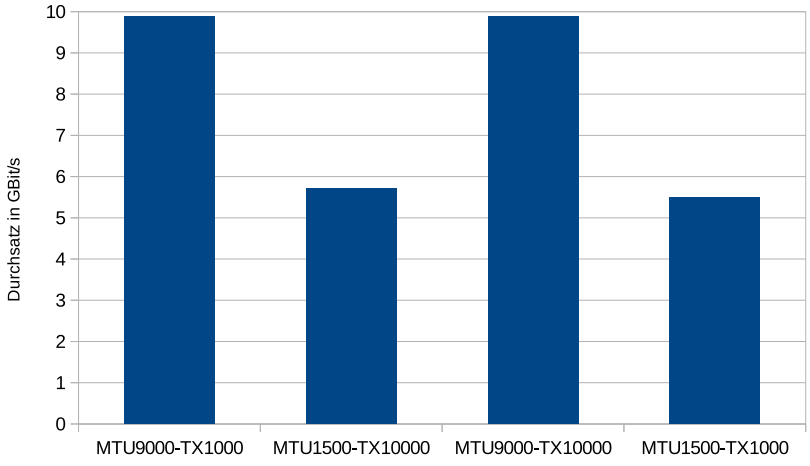


Abbildung 4: Benchmarks der Ausgangssituation im WAN

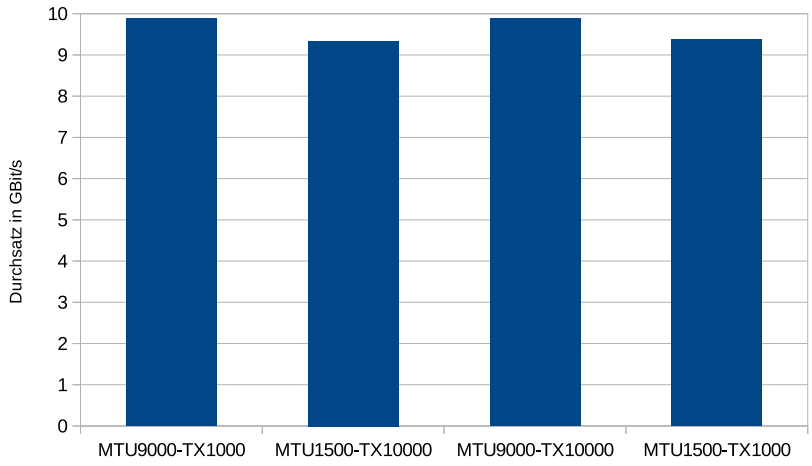


Abbildung 5: Benchmarks im LAN nach Kerneltuning

4.2.1 Kernelparameter optimiert

Durch das ändern von Kernelparametern wurden Puffer vergrößert, sowie die TCP Congestion Control auf HTCP[4] umgestellt.

Listing 2: Geänderte Kernelparameter

```
1 sysctl: name=net.core.rmem_max value=67108864
2 sysctl: name=net.core.wmem_max value=67108864
3 sysctl: name=net.ipv4.tcp_rmem value="4096 87380 33554432"
4 sysctl: name=net.ipv4.tcp_wmem value="4096 65536 33554432"
5 sysctl: name=net.core.netdev_max_backlog value=30000
6 sysctl: name=net.ipv4.tcp_congestion_control value=htcp
7 sysctl: name=net.ipv4.tcp_mtu_probing value=1
8 sysctl: name=net.ipv4.tcp_timestamps value=0
9 sysctl: name=net.ipv4.tcp_low_latency value=1
```

Nach Anpassung der genannten Parameter wurde erneut ein Benchmark durchgeführt um diese evaluieren zu können. Für die Situation im LAN (Abbildung 5) ergeben sich durch diese Maßnahmen keine signifikanten Veränderungen, weder im positiven noch im negativen Sinn. Anders verhält sich dies im WAN, wie in Abbildung 6 zu sehen ist. Auch für eine MTU von 1500 Bytes können nun Datenraten von über 9,3 GBit/s erzielt werden.

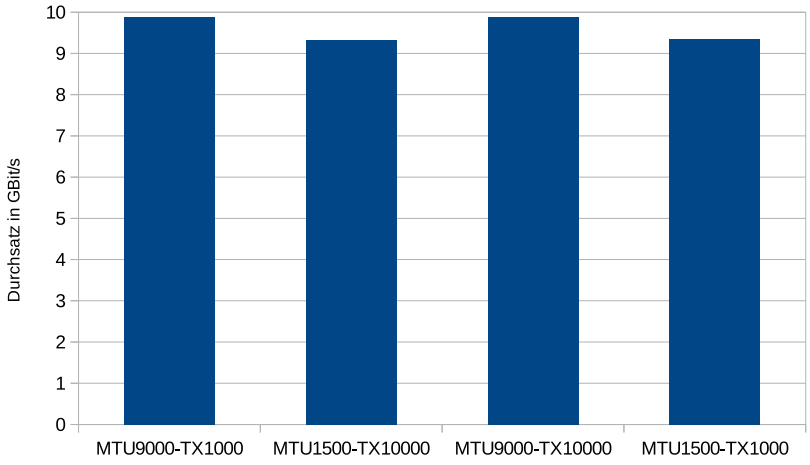


Abbildung 6: Benchmarks im WAN nach Kerneltuning

5 Zusammenfassung

Wie die Benchmarks ergaben, können gerade in Netzen mit kurzen Latenzzeiten, wie LANs bereits ohne vorheriges optimieren Bandbreiten von nahezu 10 GBit/s erzielt werden. Sobald die Latenz jedoch ansteigt bricht die Datenrate mit den Werkseinstellungen deutlich ein, was dazu führen kann, dass anstatt der erwarteten 10GBit/s nur noch knapp über 5 GBit/s erreicht werden. Durch die Benchmarks konnte gezeigt werden, dass bereits ein Anpassen der MTU auf 9000 Bytes den möglichen Datendurchsatz deutlich erhöht.
Durch weitere Einstellungen im Linuxkernel konnten darüber hinaus auch für MTUs von 1500 Bytes Ergebnisse nah am theoretischen Maximum erzielt werden.

6 Appendix: NeIF

Im Folgenden wird beschrieben, in welchem Aufbau die Benchmarks am Netzwerk für Innovation und Forschung (NeIF) durchgeführt wurden. Das NeIF ist die zentrale, vom BelWü bereitgestellte, Komponente des bwNET100G+-Projektes. Über das NeIF werden $10 \times 10G$ an den Universitätsstandorten bereitgestellt. Universitäten, die direkte Verbindungen zu mehreren anderen Universitäten haben (beispielsweise Tübingen: ist direkt mit Stuttgart und Konstanz verbunden), erhalten an ihrem Standort eine Schaltmatrix vom Typ MRV Media Cross Connect (MCC). Diese

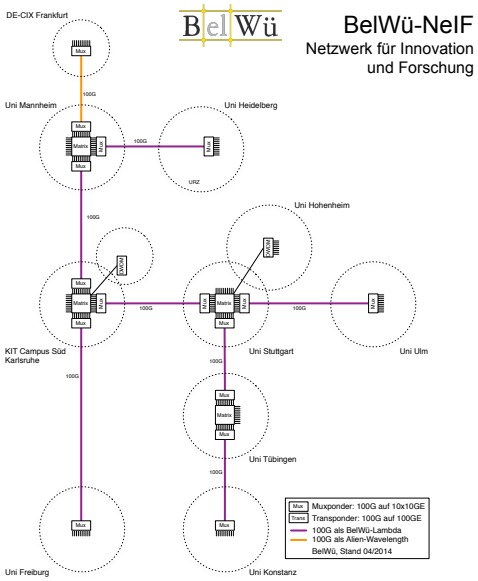


Abbildung 7: Schematischer Aufbau des Netzwerk für Innovation und Forschung (NeIF)

Schaltmatrizen verfügen über je 36 SFP+ Ports, die per Software dynamisch zueinander gemappt werden können. End-Standorte, die nur eine direkte Verbindung zu einer weiteren Universität haben (Beispielsweise Ulm: ist direkt nur mit Stuttgart verbunden), werden mit einem Optelian MPX-9110 Muxponder an das NeIF angeschlossen. Größere Universitätsstandorte (beispielsweise das KIT), verfügen über zwei Schaltmatrizen. Die Verbindungen zweier benachbarter Universitäten zueinander werden ebenfalls über die Optelian-Muxponder realisiert. Der Aufbau ist in Abbildung 7 dargestellt.

Im Laufe diverser Experimente im Jahr 2015 stellten die Forscher die Anforderung an möglichst hohen Latenzen zwischen den Knoten im NeIF. Aus dieser Anforderung entstand das Konzept der "NeIF-Ringe". Ein solcher Ring besteht aus insgesamt drei in Reihe geschalteten NeIF-Strecken zwischen allen Standorten. Datenpakete können so durch einen Ring beispielsweise von Karlsruhe nach Ulm, von Ulm weiter nach Tübingen und von Tübingen zurück nach Karlsruhe gesendet werden. Ein solcher Ringumlauf hat eine Strecke von etwa 536km und erzeugt auf der Strecke eine Latenz von 6ms. Im Maximum stehen fünf solcher Ringe zur Verfügung. Abzüglich produktiv genutzter NeIF-Strecken bleiben vier Ringe für Versuche erhalten. Werden die verfügbaren vier Ringe in Reihe geschaltet, entsteht eine Gesamtstrecke von mehr als 2100km mit einer Latenz von 24ms.

Literatur

[1] L. Bradatsch, “Verhalten von TCP in Hochgeschwindigkeitsnetzen,” 2015.

[2] ESnet, “Host tuning,” https://fasterdata.es.net/host-tuning/.

[3] D. Borman, B. Braden, V. Jacobson, and E. R. Scheffenegger, “TCP Extensions for High Performance,” RFC 7323 (Proposed Standard), Internet Engineering Task Force, Sep. 2014.

[4] D. Leith and R. Shorten, “H-tcp: Tcp for high-speed and long-distance networks,” in *Proceedings of PFLDnet*, vol. 2004, 2004.