HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Bachelorarbeit: A Real-time Streaming Protocol for Large-Scale Peer-to-Peer Networks

Christopher Probst

Institut für Informatik
Heinrich-Heine-Universität Düsseldorf

3.12.2014

Gliederung

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Einleitung
- Problemstellung
- Motivation
- Zielsetzung

Umsetzung
- Theorie
- Implementierung

Evaluation
- Methodik
- Ergebnisse
- Fazit

Problemstellung

- Möchte man Daten von einem einzelnen Computer aus zu beliebig vielen anderen Computern transferieren, so wird häufig ein Client / Server Netzwerk genutzt.
- Diese Methode ist einfach, zuverlässig und lang erprobt. Jedoch gibt es Skalierungsprobleme:
  - Jeder Client erhöht die Last des Servers
  - Mehr Clienten benötigen mehr Server und mehr Bandbreite
  - Uploadbandbreite der Clienten bleibt genutzt

Motivation

- Um ohne ein skalierbares Serversystem Daten dennoch schnell zu verbreiten, kann ein Peer-to-Peer Netzwerk benutzt werden, wo jeder Computer einen Peer darstellt und bei der Datenverbreitung hilft. Einige Anwendungsfälle sind:
  - File Sharing (BitTorrent)
  - Audio und Video Streaming (Skype)
  - DHTs (Kademlia, Chord)
- Geräte mit einer geringen Uploadbandbreite können so dennoch schnell Daten verbreiten, wie z.B. ein Live-Videostream von einem Smartphone aus.

Zielsetzung

Implementierung einer Peer-to-Peer Anwendung, die Daten ausgehend von einem Peer, auch genannt Super-Peer, an beliebig viele Peers versendet. Dabei sollen folgende Bedingungen eingehalten werden:

- Möglichst hohe Auslastung der Uploadbandbreite der einzelnen Peers.
- Jeder Peer soll die Daten möglichst zur gleichen Zeit fertigstellen.
- Gesamtdauer unabhängig von der Anzahl der Peers.
- Gesamtdauer kleiner als $2 * T_0$.

Umsetzung - Theorie

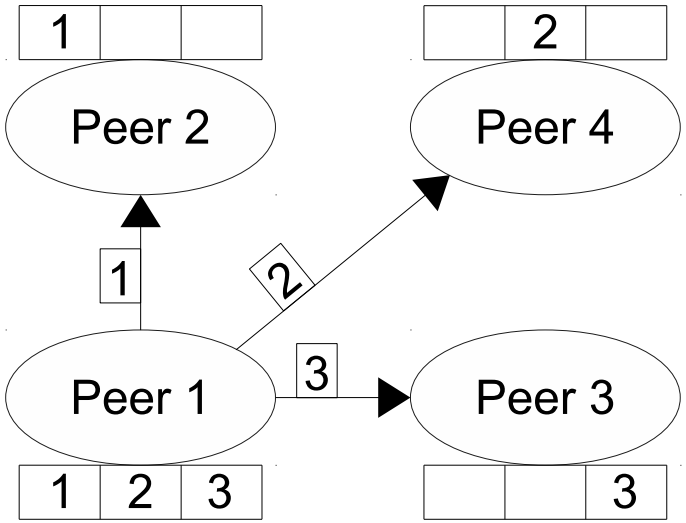
Um die Gesamtdauer gering zu halten, muss jeder Peer bei der Verteilung des Datensatzen mithelfen. Das im Folgenden erklärte Verfahren nennt sich Chunked-Swarm:

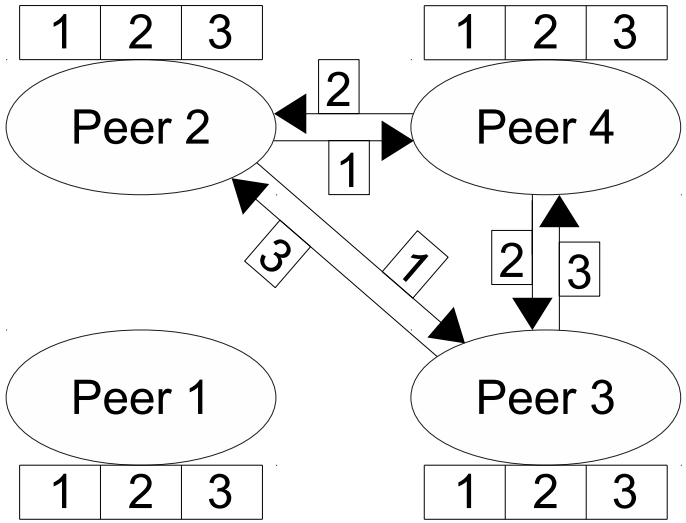
- Die Daten werden vor dem Transfer in kleine Teile (Chunks) geteilt. Es muss mindestens so viele Chunks geben, wie es Peers im Netzwerk gibt.
- Der Super-Peer verschickt disjunkte Chunks parallel an alle übrigen Peers.
- Jeder Peer verschickt seinen Chunk an alle anderen Peers.
- Es wird vereinfacht angenommen, dass jeder Peer die gleiche Uploadbandbreite hat.

Umsetzung - Theorie

Wenn es genauso viele Chunks wie Peers gibt, kann man den Transfer in zwei Stufen betrachten. Peer 1 ist in diesem Fall der Super-Peer.

- Links: Alle Peers bekommen einen disjunkten Chunk vom Super-Peer.
- Rechts: Die Peers schicken sich ihren eigenen Chunk gegenseitig zu.

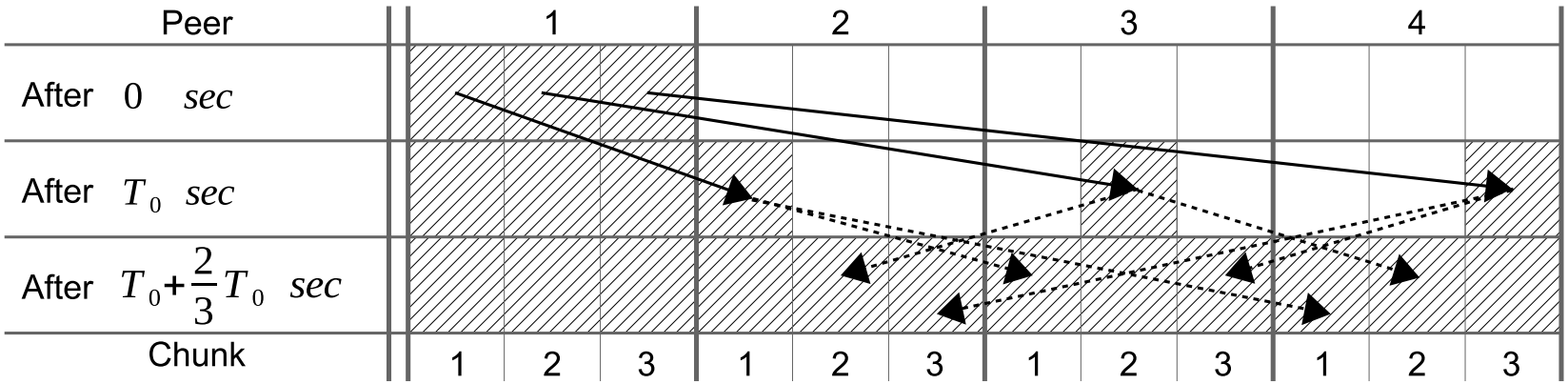




Umsetzung - Theorie

Das folgende Bild zeigt den zeitlichen Ablauf:

- In den ersten T_0 Sekunden, schickt der Super-Peer einen disjunkten Chunk an jeden Peer. Jeder Chunk ist daher $\frac{\textit{Datengröße}}{3}$ groß.
- Anschließend schickt jeder Peer seinen Chunk an die anderen beiden Peers, was $\frac{2}{3} * T_0$ Sekunden dauert.
- Nach $T_0 + \frac{2}{3} * T_0$ Sekunden besitzt also jeder Peer die Daten.



Umsetzung - Theorie

- Verdoppelt man die Anzahl der Chunks, halbiert man die Zeit zwischen T_0 Sekunden und dem Ende.
- Die allgemeine Gesamtdauer lässt sich daher mit $T(n,c) = T_0 + \frac{n}{c} * \frac{n-1}{n} * T_0 = (1 + \frac{n-1}{c}) * T_0$ berechnen, wobei n die Anzahl der Peers, jedoch ohne den Super-Peer, und $c = n * 2^i, i \in \mathbb{N}_0$ die Anzahl der Chunks ist.
- Mit dieser Methode kann man immer unter $2 * T_0$ Sekunden kommen. Verdoppelt man die Anzahl der Chunks beliebig oft, so kann man sogar beliebig nah an T_0 Sekunden herankommen.

Umsetzung - Implementierung

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Mesh Topologie: Jeder Peer ist mit jedem Peer verbunden.
- Pull-Based: Chunks werden nur auf Wunsch übertragen.
- Announcements: Jeder Peer kündigt an, welche Chunks vorliegen.
- Automatic (Re-)Connect: Peers finden andere Peers durch Super-Peer.

Umsetzung - Implementierung

- Peer-Loss Detection: Super-Peer verschickt Chunks von Peers, die das Netzwerk verlassen haben, erneut.
- Jeder Peer versucht von möglichst vielen Peers gleichzeitig Chunks zu beziehen. Dies ist wichtig, da ein Pull-Based Ansatz verwendet wird.
- Die Bandbreite der Peers kann beliebig gedrosselt werden.
- Implementiert in Java mit Netty5.

Evaluation - Methodik

- Ein Szenario ist eine bestimmte Konfiguration aller verfügbaren Parameter.
- Das erste Szenario (Default) dient als Vergleichsszenario. Jedes weitere Szenario ändert nur einen Parameter des Default-Szenarios.
- Jedes Szenario wird 10-mal durchlaufen.
- Geplottet wird der Durchschnitt und das Konfidenzintervall mit einem Konfidenzniveau von 95%.

Evaluation - Methodik

Besonderheiten:

- Die Szenarios laufen nicht mit TCP. Alle Verbindung sind nur virtuell und benötigen keine Sockets.
- Es findet keine (De-)Serialisierung der Netzwerkpakete statt, da die virtuellen Verbindungen das direkte Verschicken von Referenzen unterstützen.
- Jedes Szenario simuliert dafür pro verschickte Referenz eine gewisse Datengröße, damit die genutzte Bandbreite beeinflusst wird und die Benchmark näher an der Realität sind.
- Durch diese Methoden wird Speicher und CPU Auslastung massiv verringert.

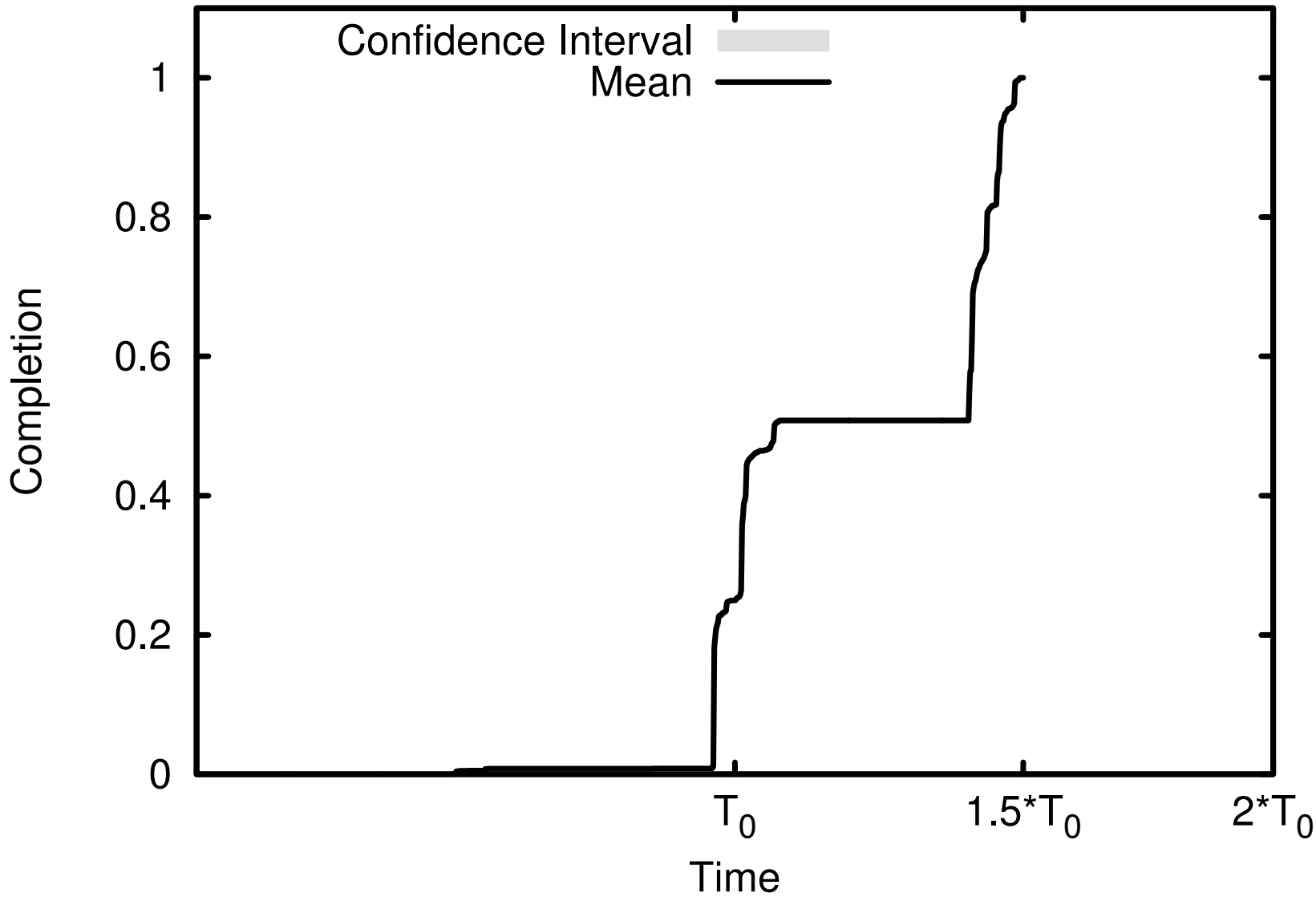
Ergebnisse - Default Szenario

Das Default-Szenario simuliert ein Peer-to-Peer Netzwerk mit folgenden Eigenschaften:

- Es gibt einen Super-Peer und 63 Peers.
- Doppelt so viele Chunks wie Peers (126 Chunks).
- Gleiche Uploadbandbreite für Super-Peer und Peers.
- Datengröße so gewählt, dass $T_0 = 10$ Minuten gilt.

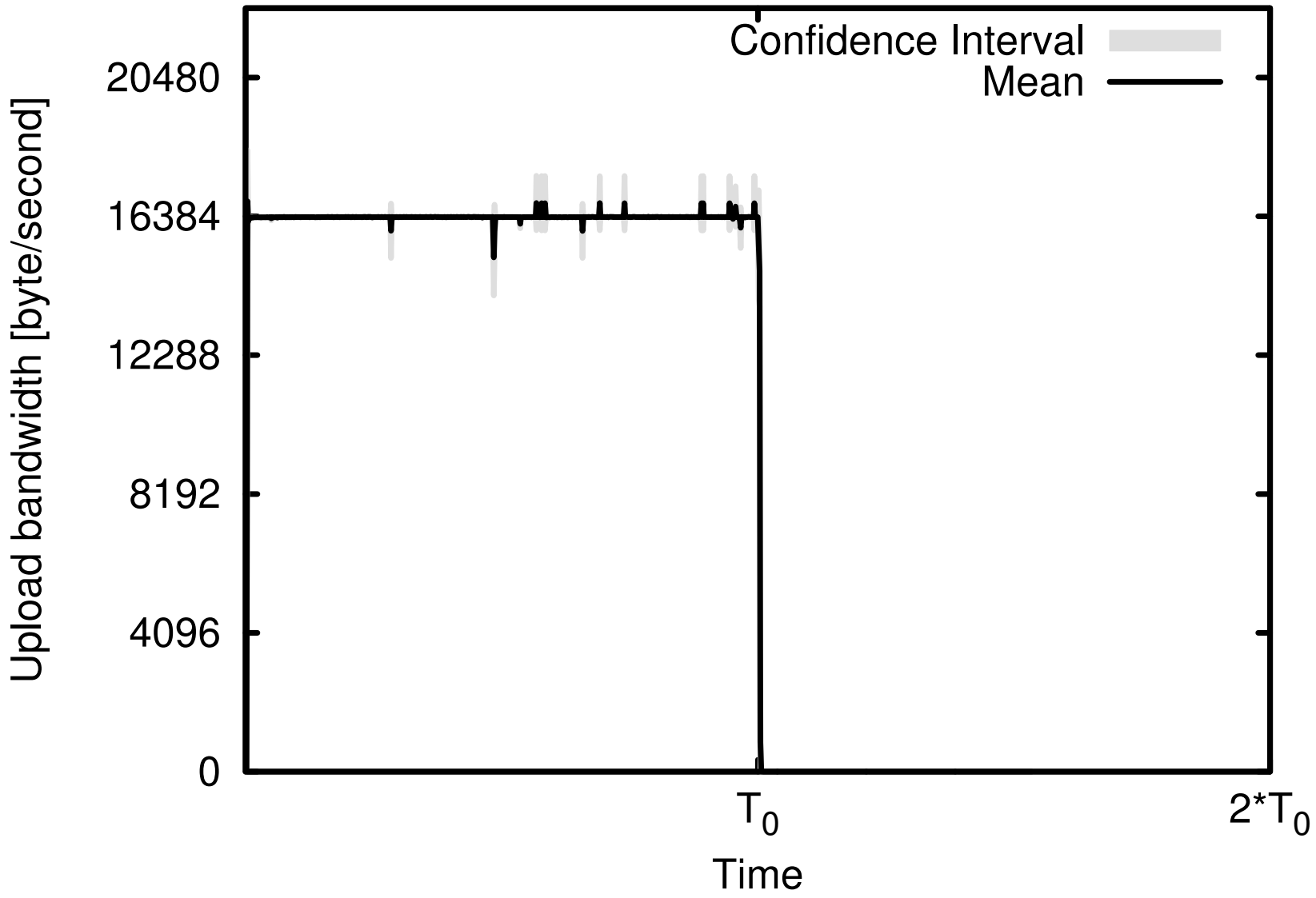
Ergebnisse - Default Szenario

Completion Graph:



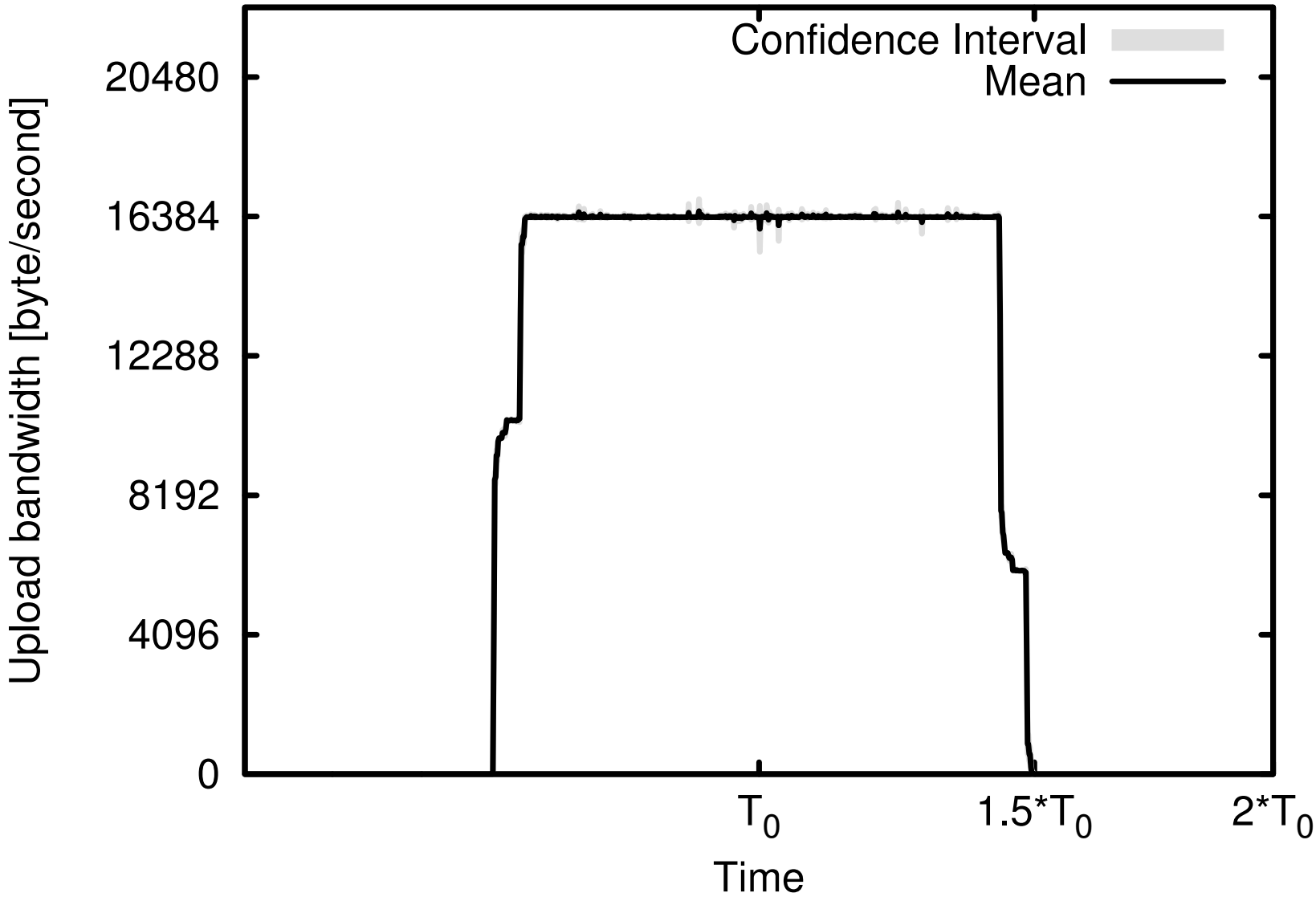
Ergebnisse - Default Szenario

Super-Peer Upload Bandwidth:



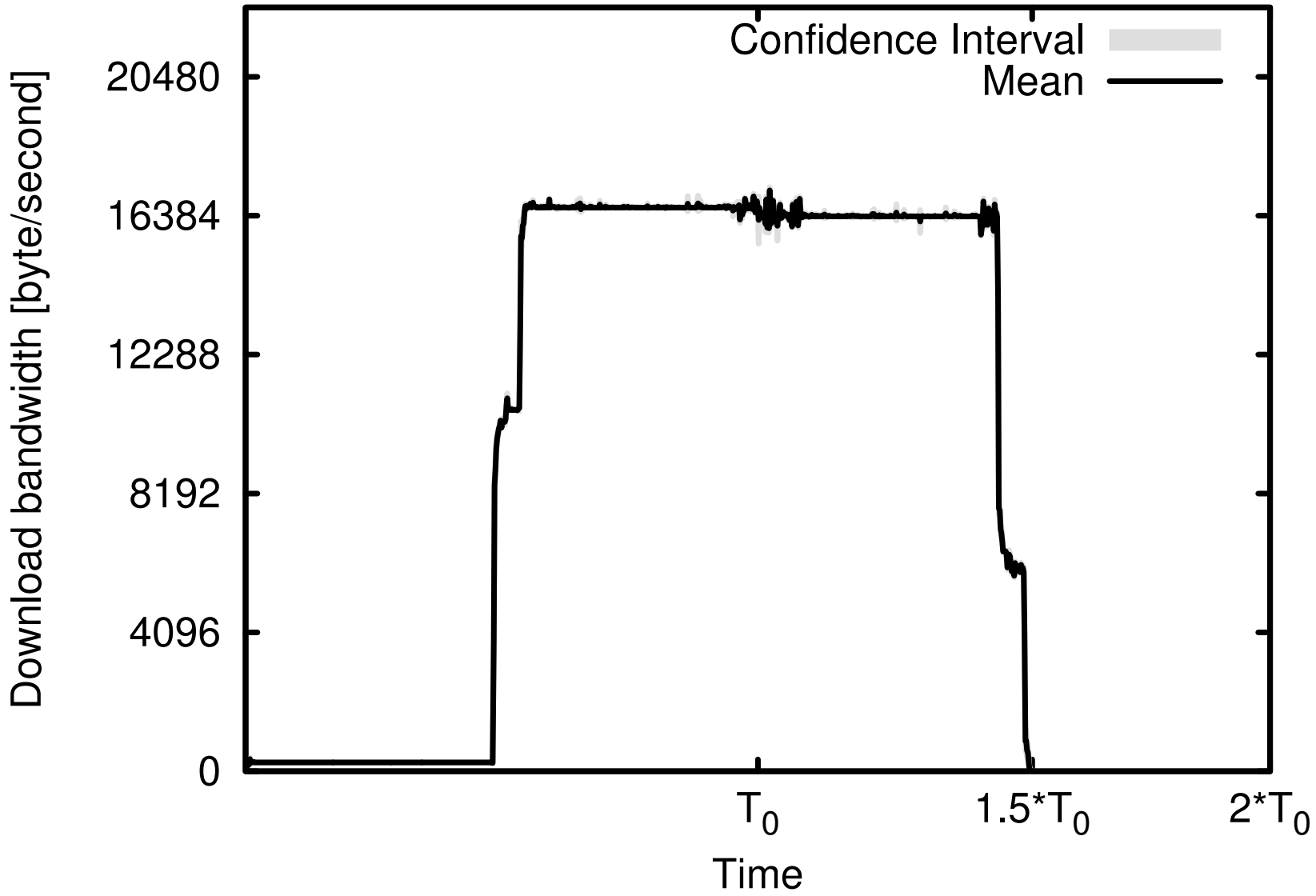
Ergebnisse - Default Szenario

Peer Upload Bandwidth:



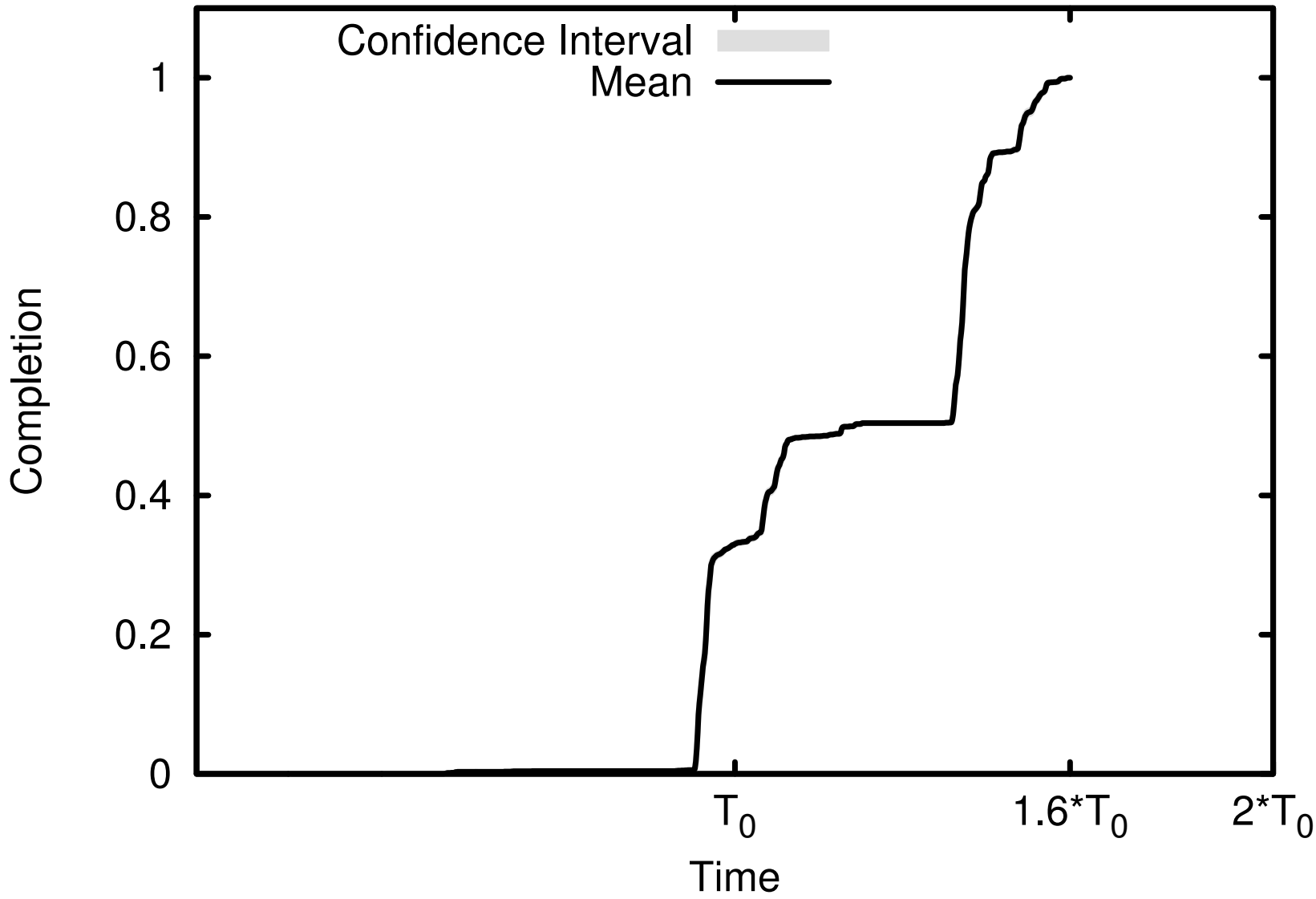
Ergebnisse - Default Szenario

Peer Download Bandwidth:



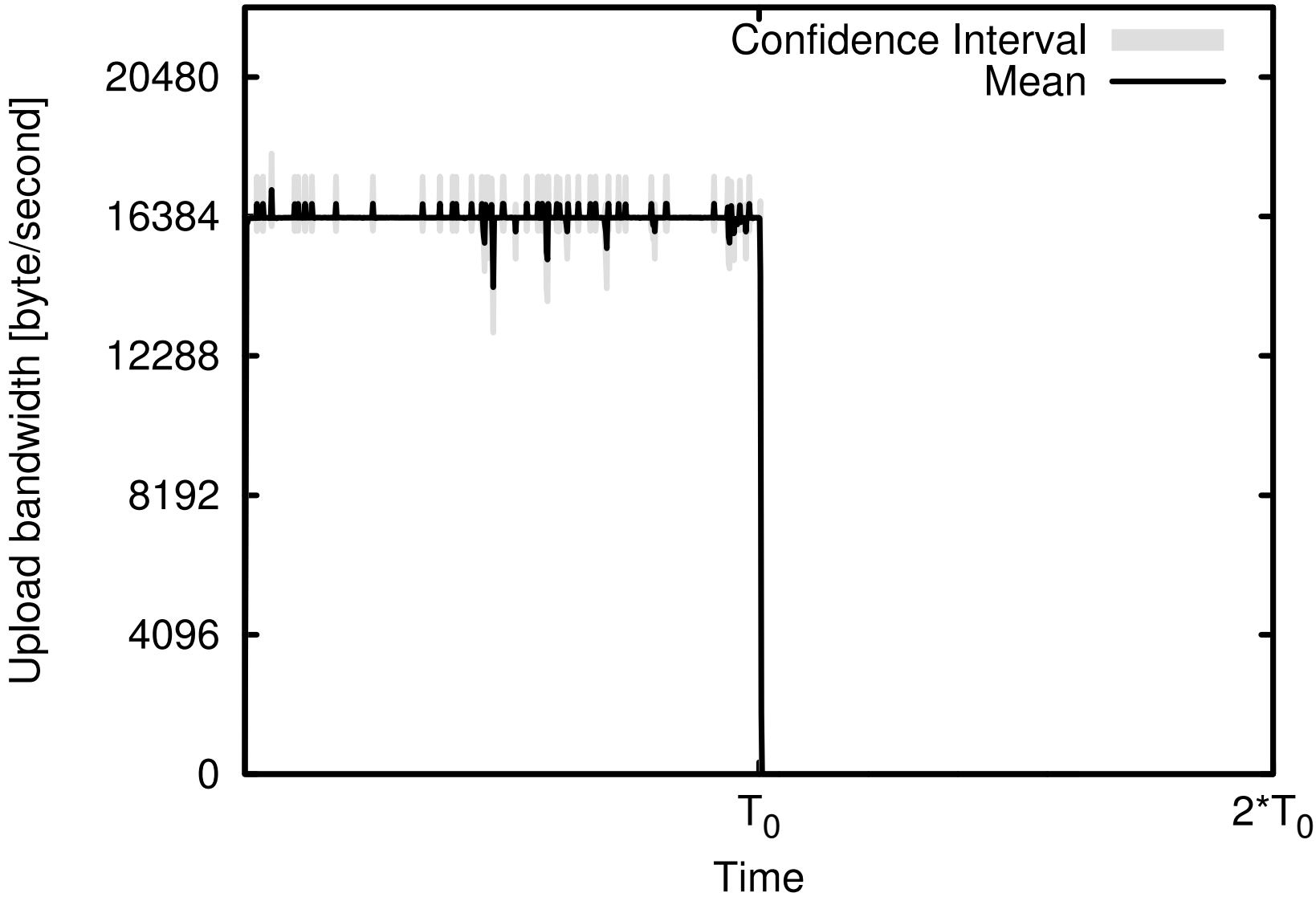
Ergebnisse - Default Szenario mit 128 Peers

Completion Graph:



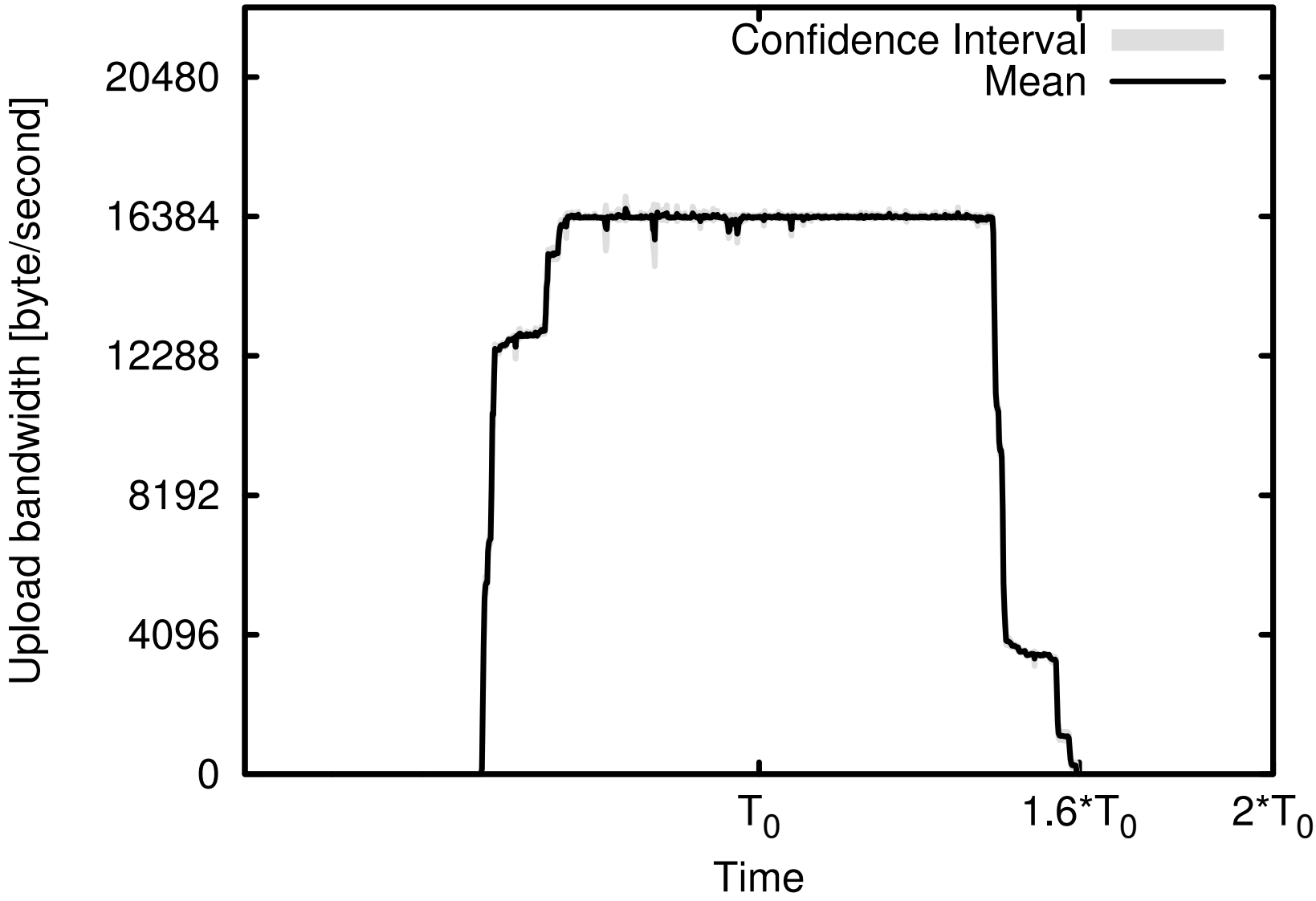
Ergebnisse - Default Szenario mit 128 Peers

Super-Peer Upload Bandwidth:



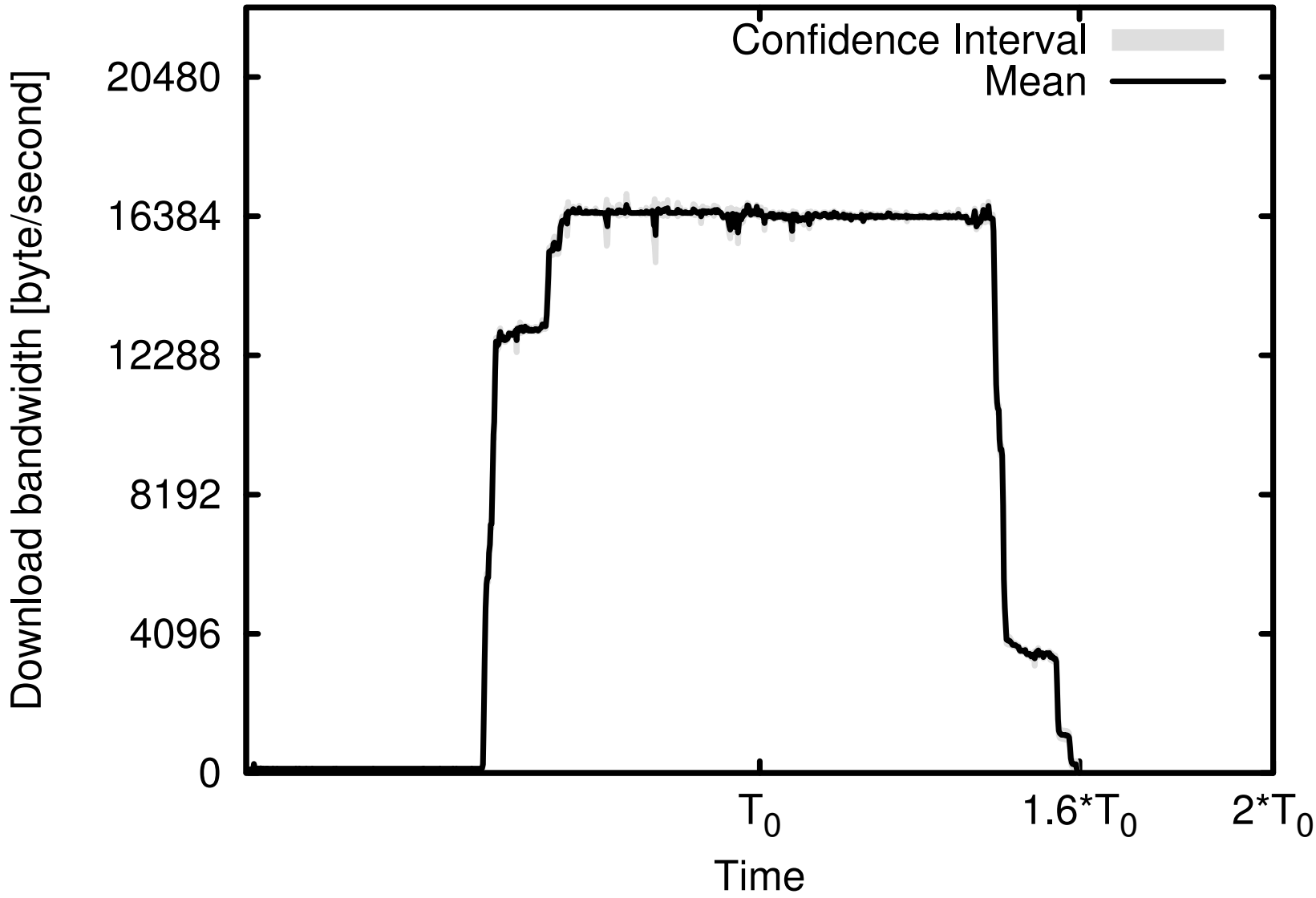
Ergebnisse - Default Szenario mit 128 Peers

Peer Upload Bandwidth:



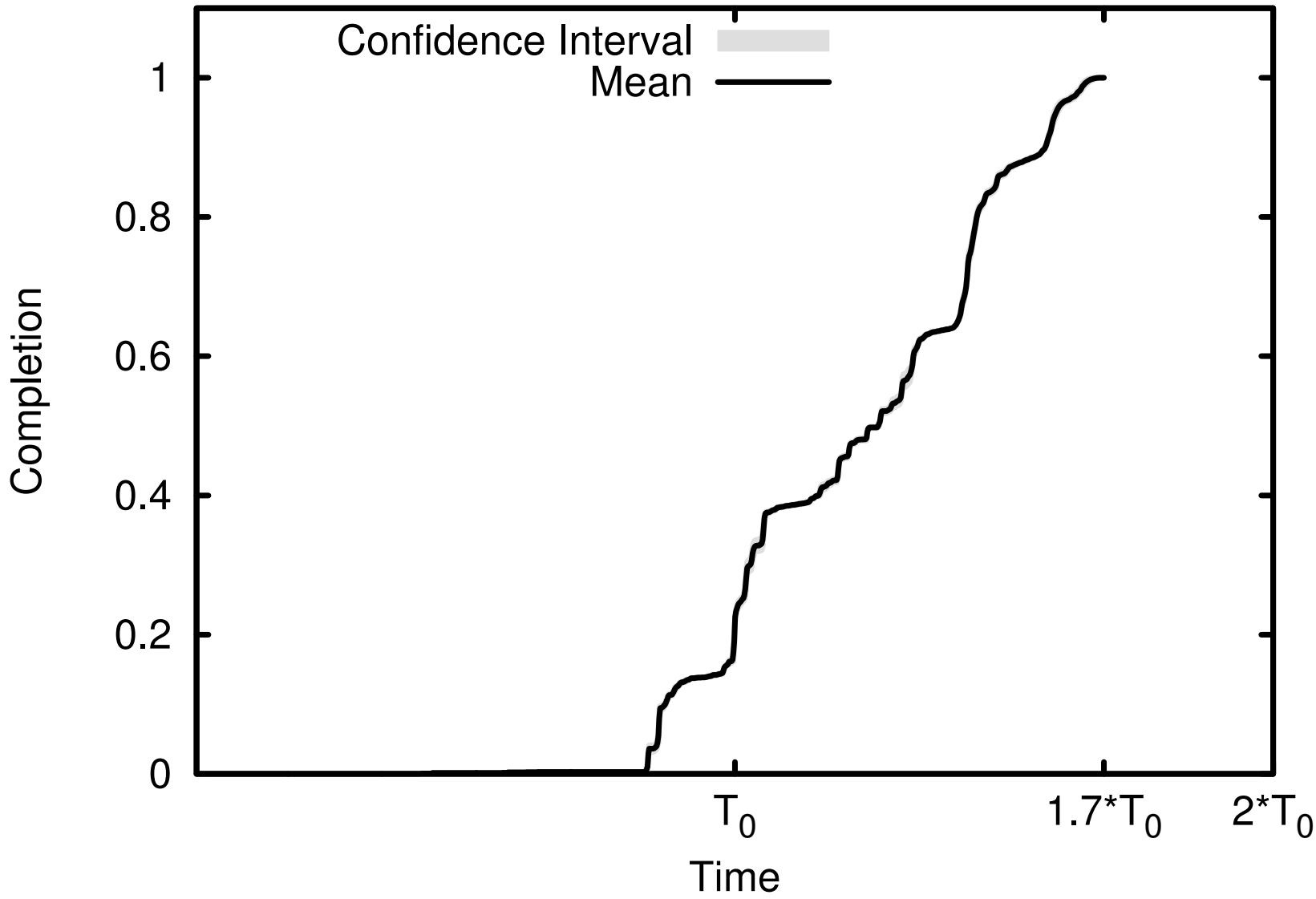
Ergebnisse - Default Szenario mit 128 Peers

Peer Download Bandwidth:



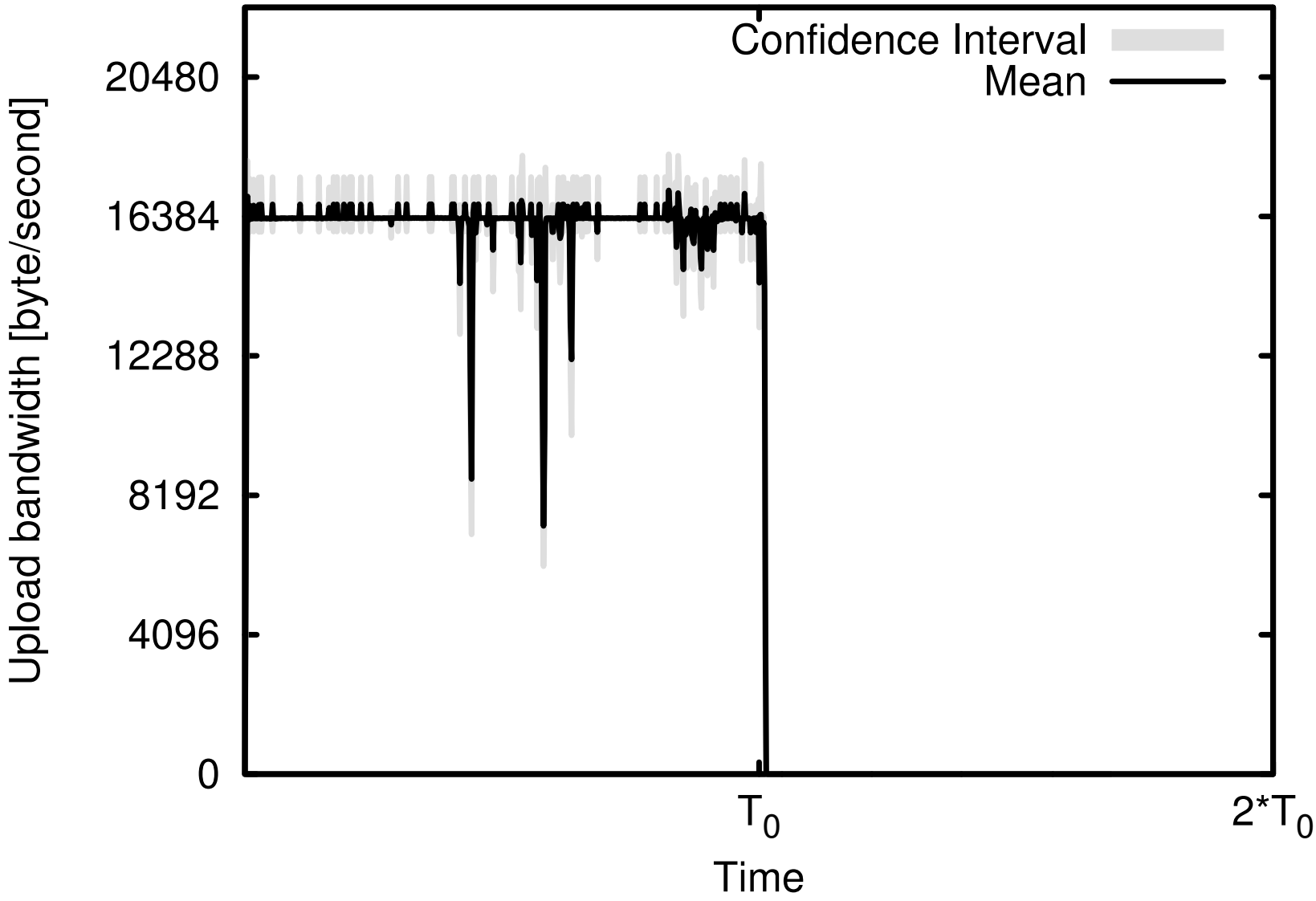
Ergebnisse - Default Szenario mit 192 Peers

Completion Graph:



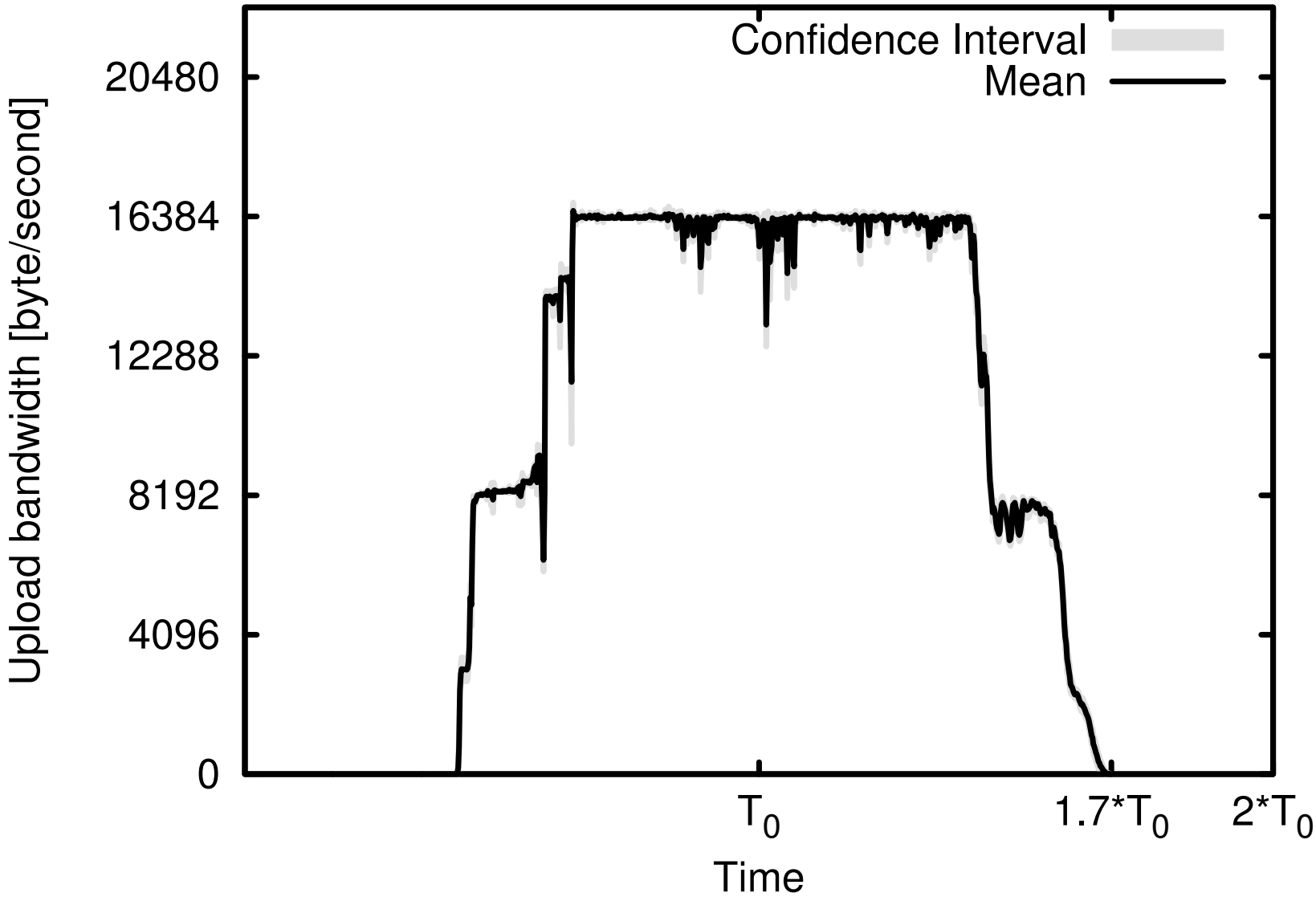
Ergebnisse - Default Szenario mit 192 Peers

Super-Peer Upload Bandwidth:



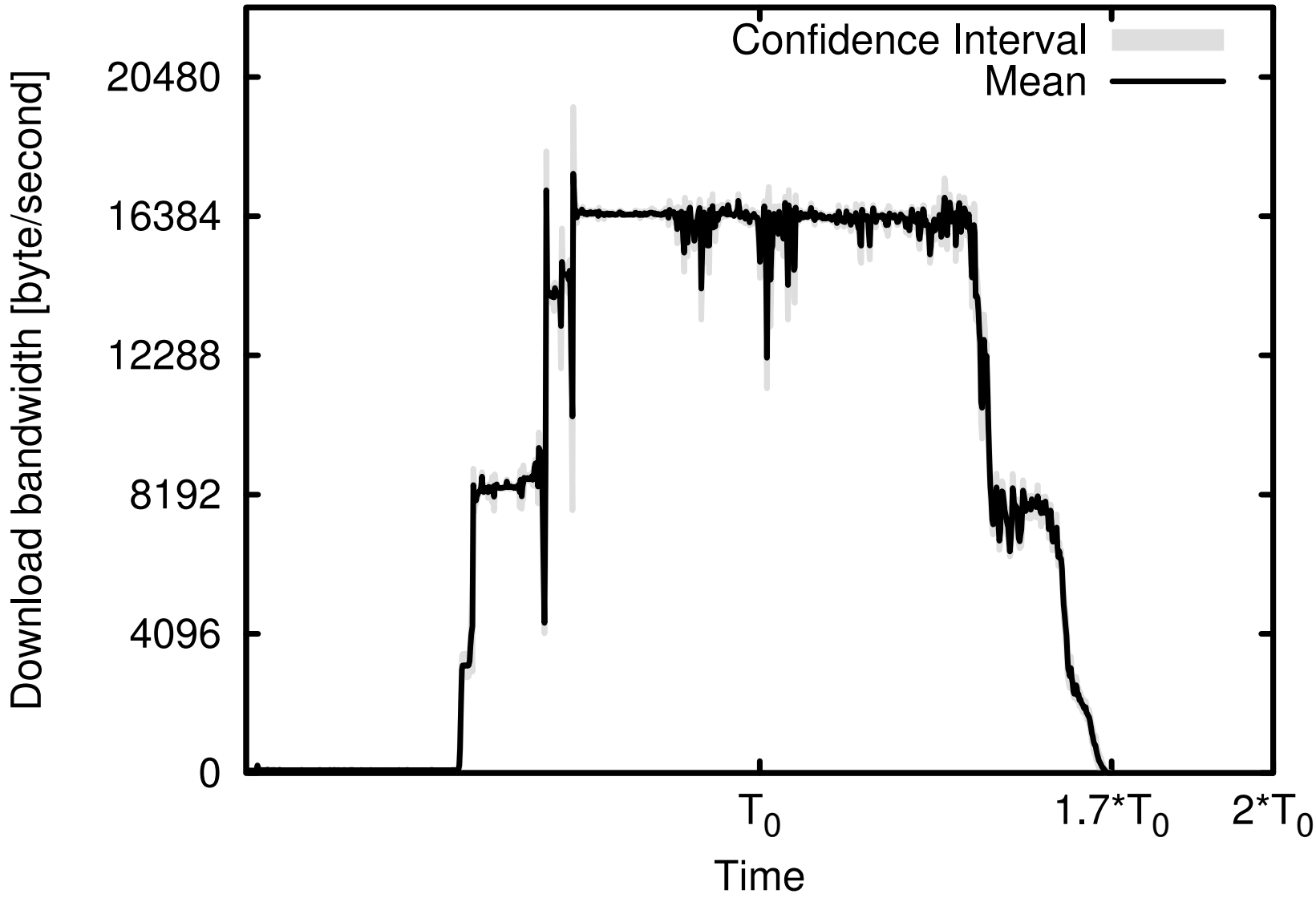
Ergebnisse - Default Szenario mit 192 Peers

Peer Upload Bandwidth:



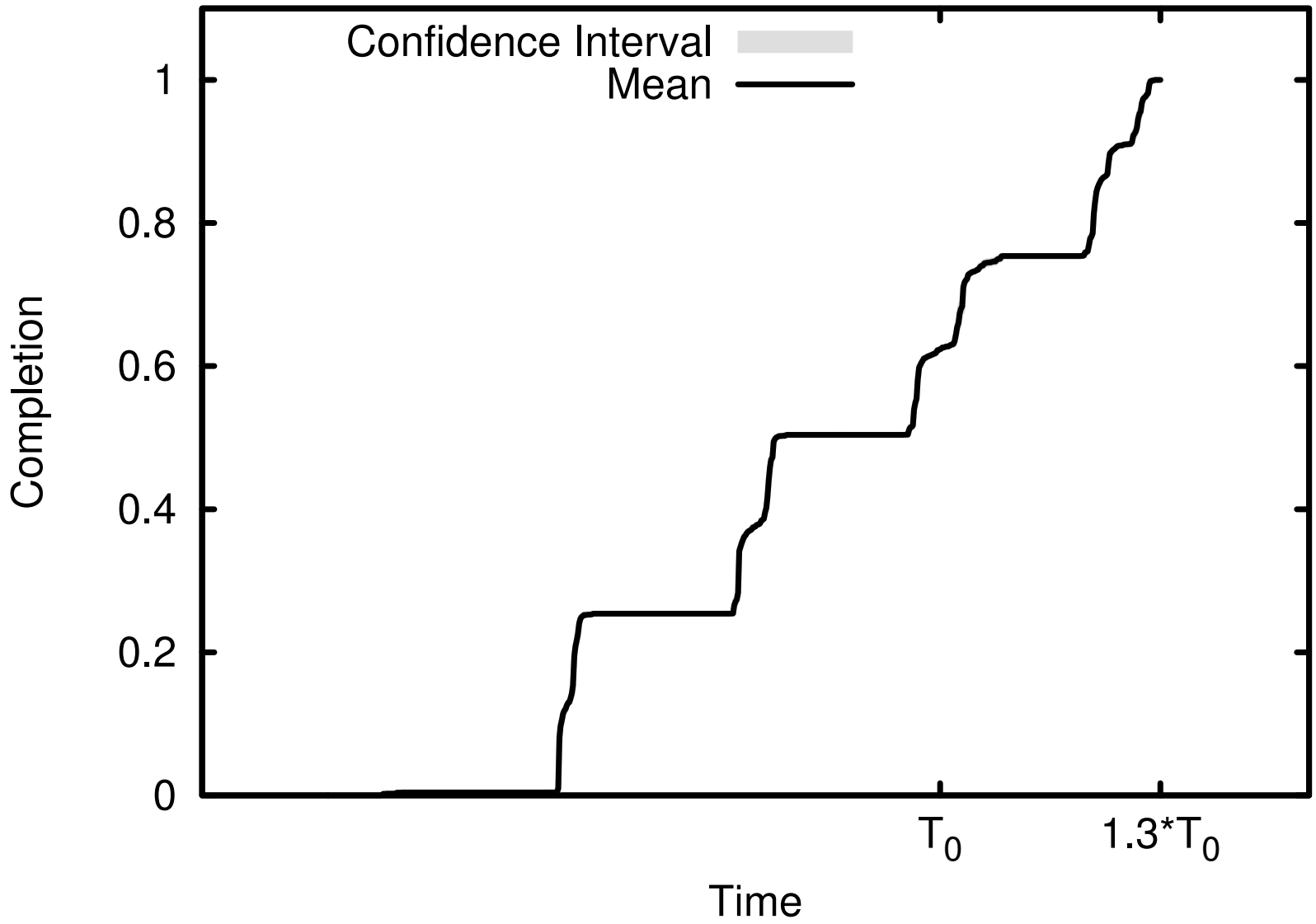
Ergebnisse - Default Szenario mit 192 Peers

Peer Download Bandwidth:



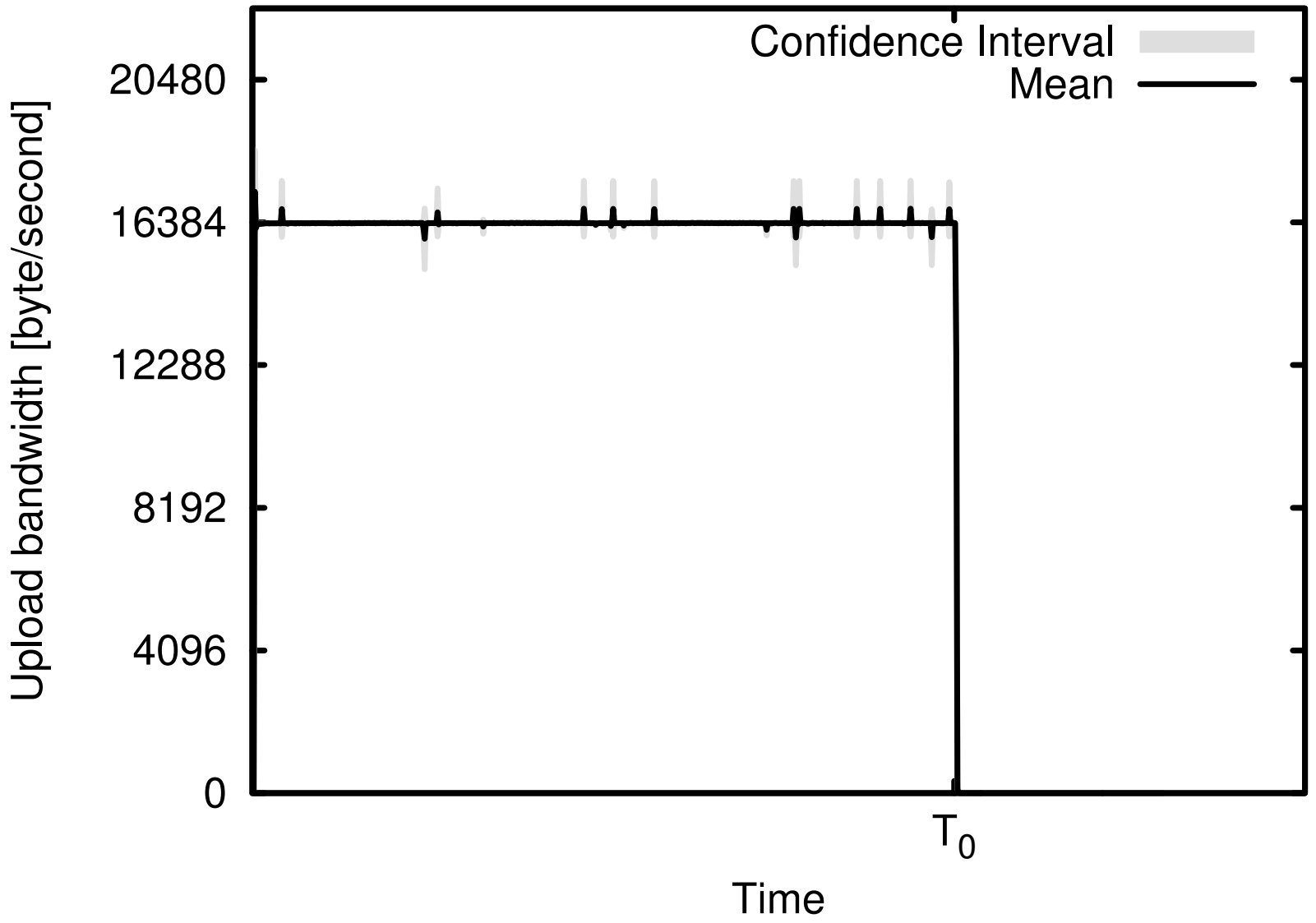
Ergebnisse - Default Szenario mit 4x Chunkanzahl

Completion Graph:



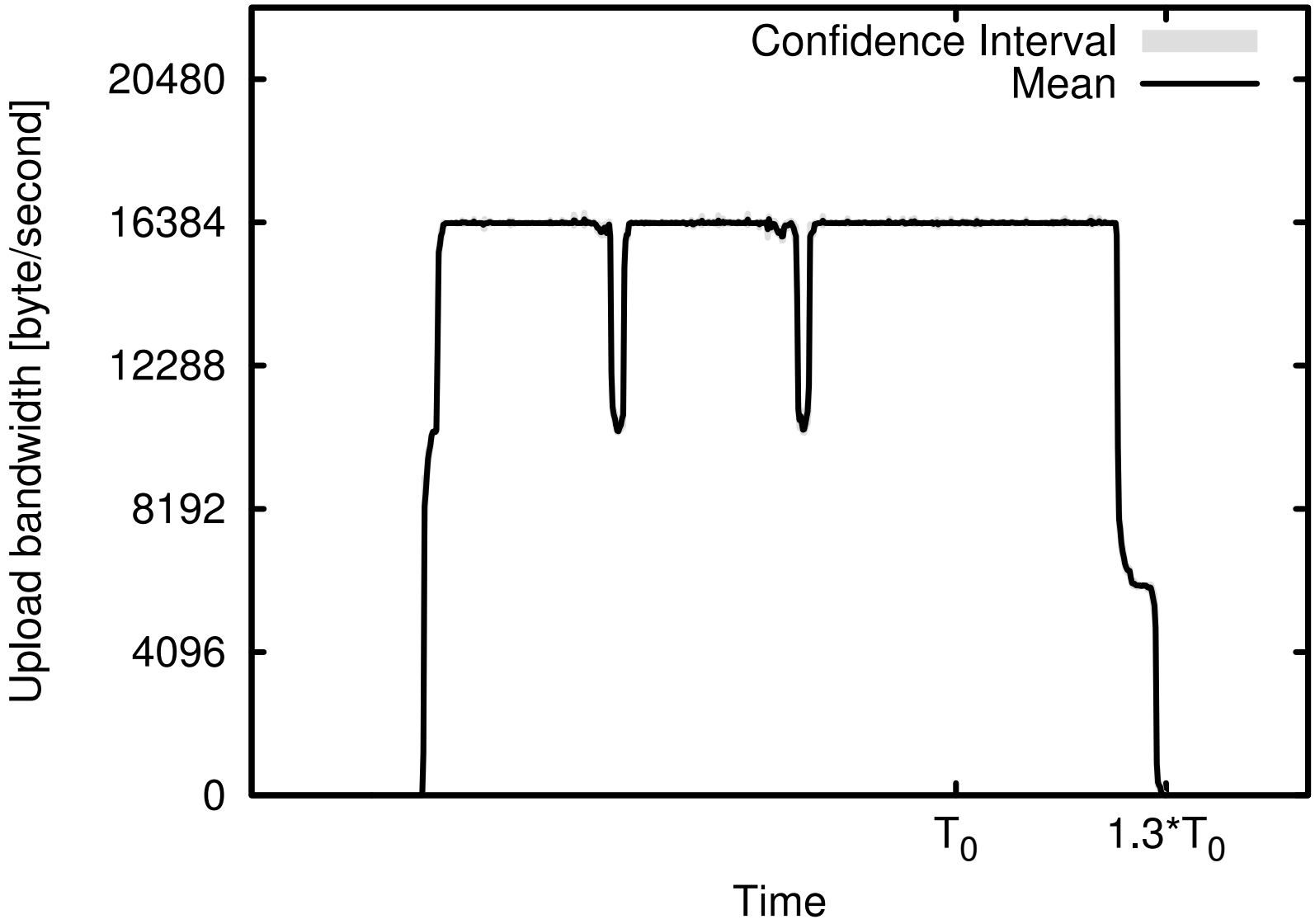
Ergebnisse - Default Szenario mit 4x Chunkanzahl

Super-Peer Upload Bandwidth:



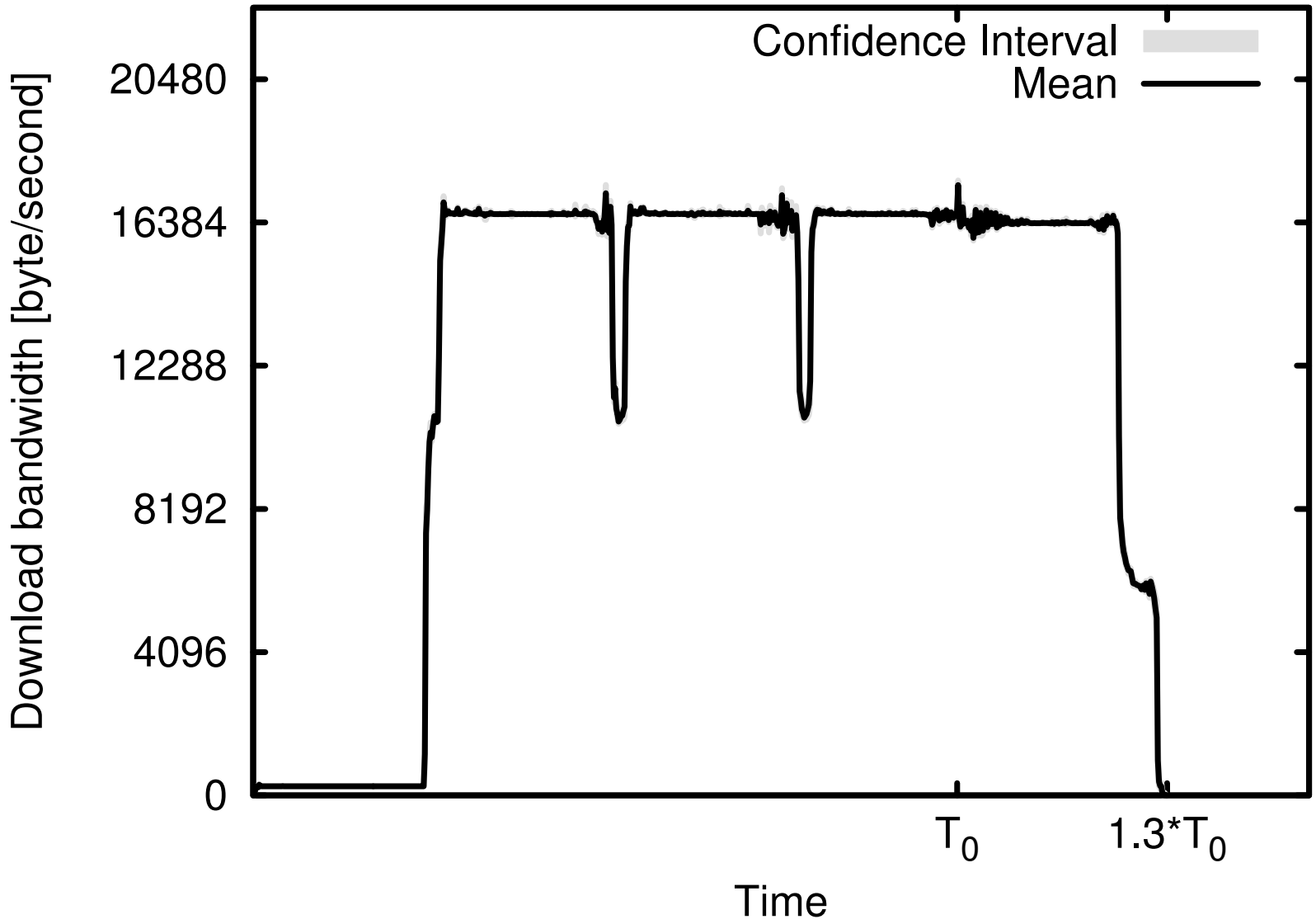
Ergebnisse - Default Szenario mit 4x Chunkanzahl

Peer Upload Bandwidth:



Ergebnisse - Default Szenario mit 4x Chunkanzahl

Peer Download Bandwidth:



Ergebnisse - Default Szenario mit 20 Datensätzen (Stream)

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

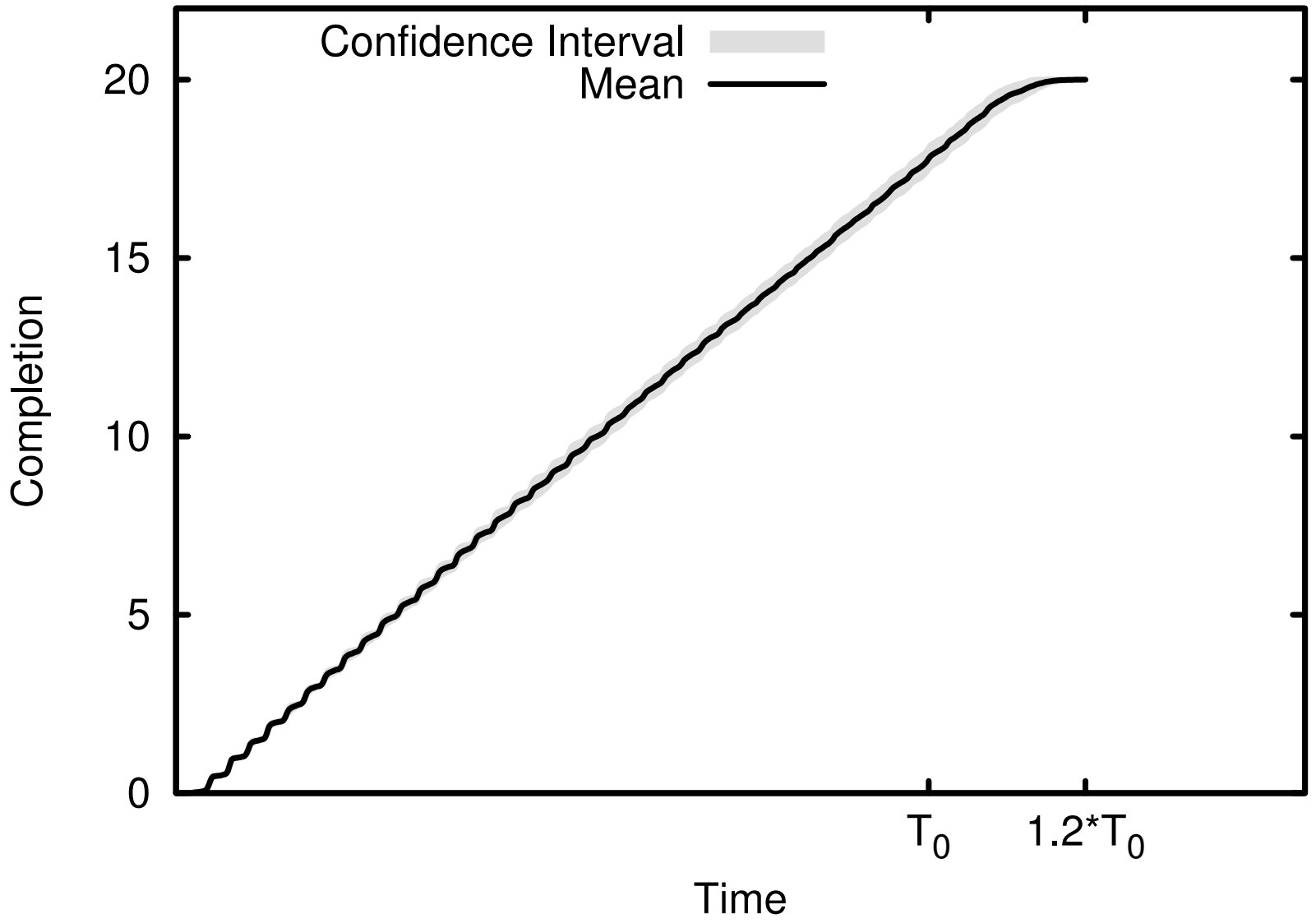
Bislang gab es immer nur einen Datensatz, der verteilt wurde. Nun wird der gesamte Datensatz in 20 weitere Sub-Datensätze geteilt.

- Jeder Sub-Datensatz hat wieder doppelt soviele Chunks wie Peers.
- Die Sub-Datensätze sind durchnummeriert mit IDs.
- Sub-Datensätze mit kleiner ID haben Vorrang.
- Streaming!

Ergebnisse - Default Szenario mit 20 Datensätzen (Stream)

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

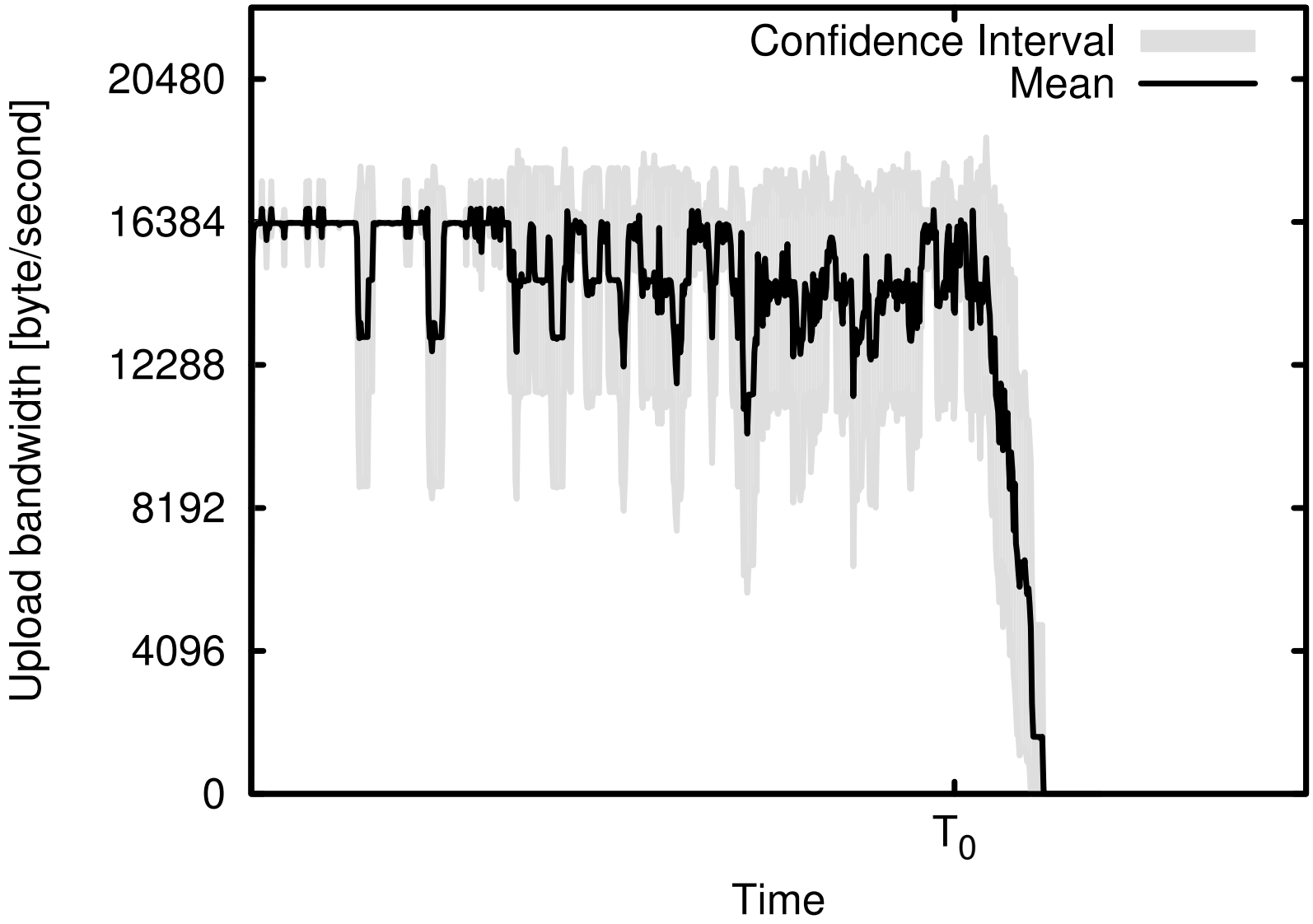
Completion Graph:



Ergebnisse - Default Szenario mit 20 Datensätzen (Stream)

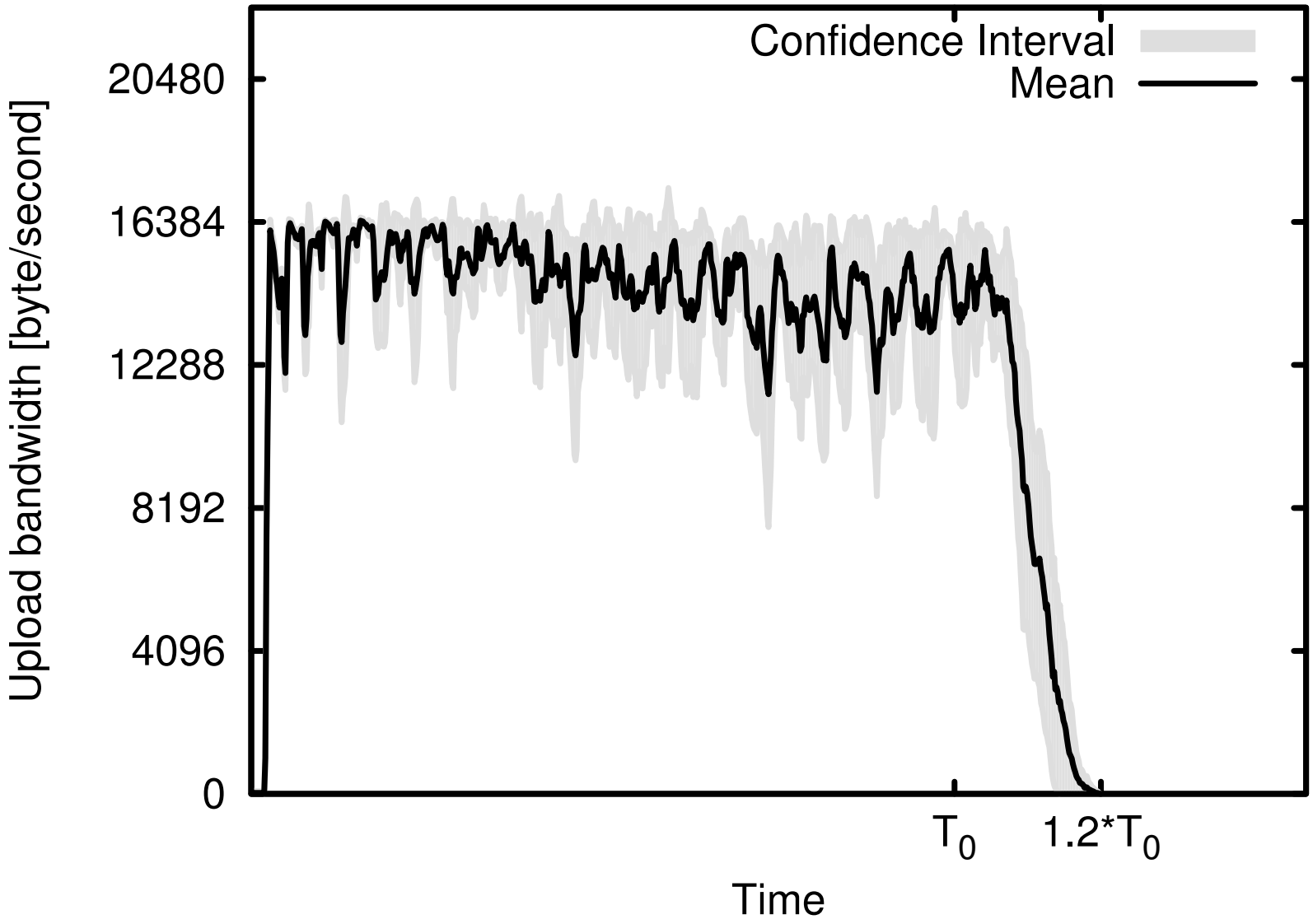
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Super-Peer Upload Bandwidth:



Ergebnisse - Default Szenario mit 20 Datensätzen (Stream)

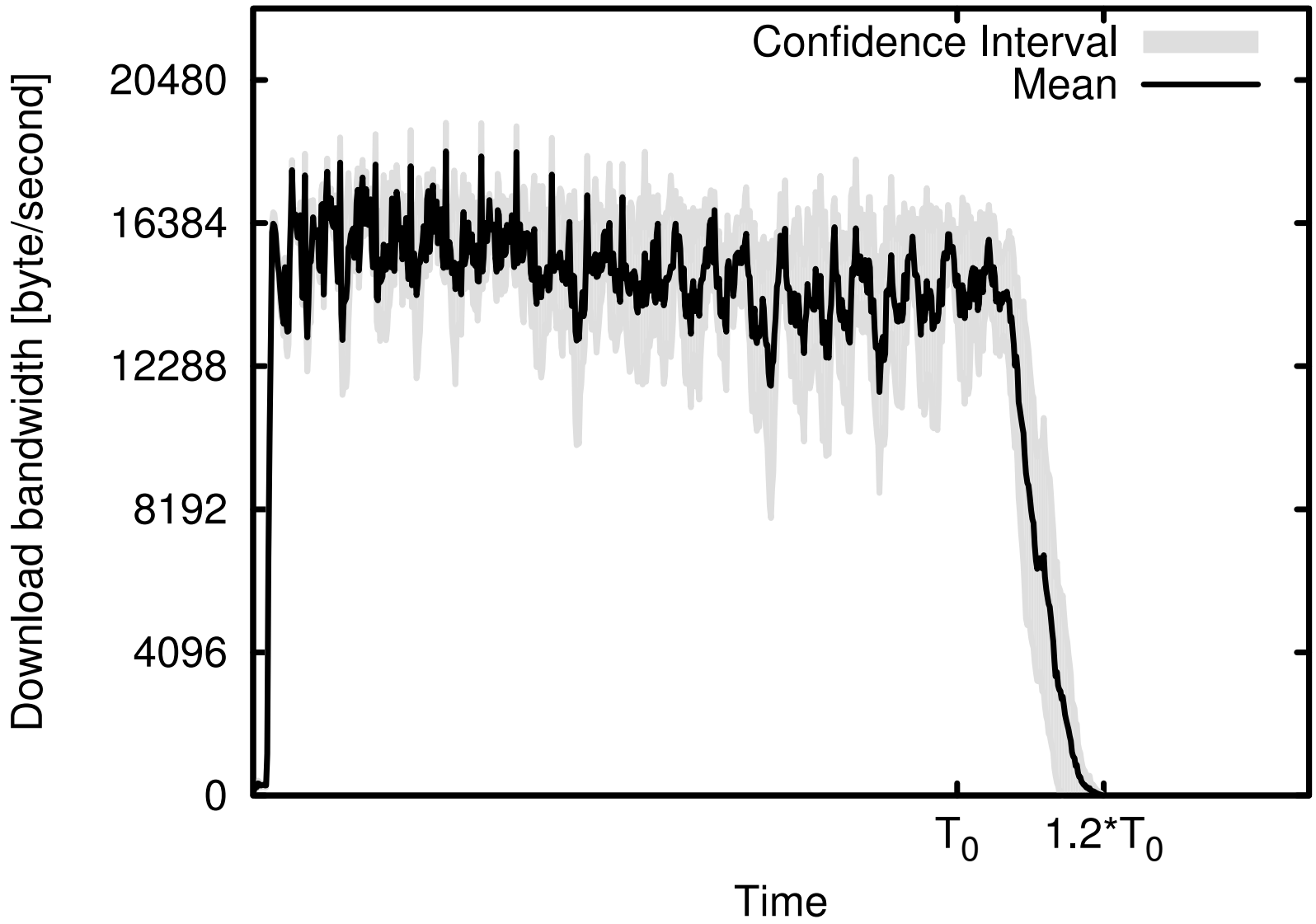
Peer Upload Bandwidth:



Ergebnisse - Default Szenario mit 20 Datensätzen (Stream)

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Peer Download Bandwidth:



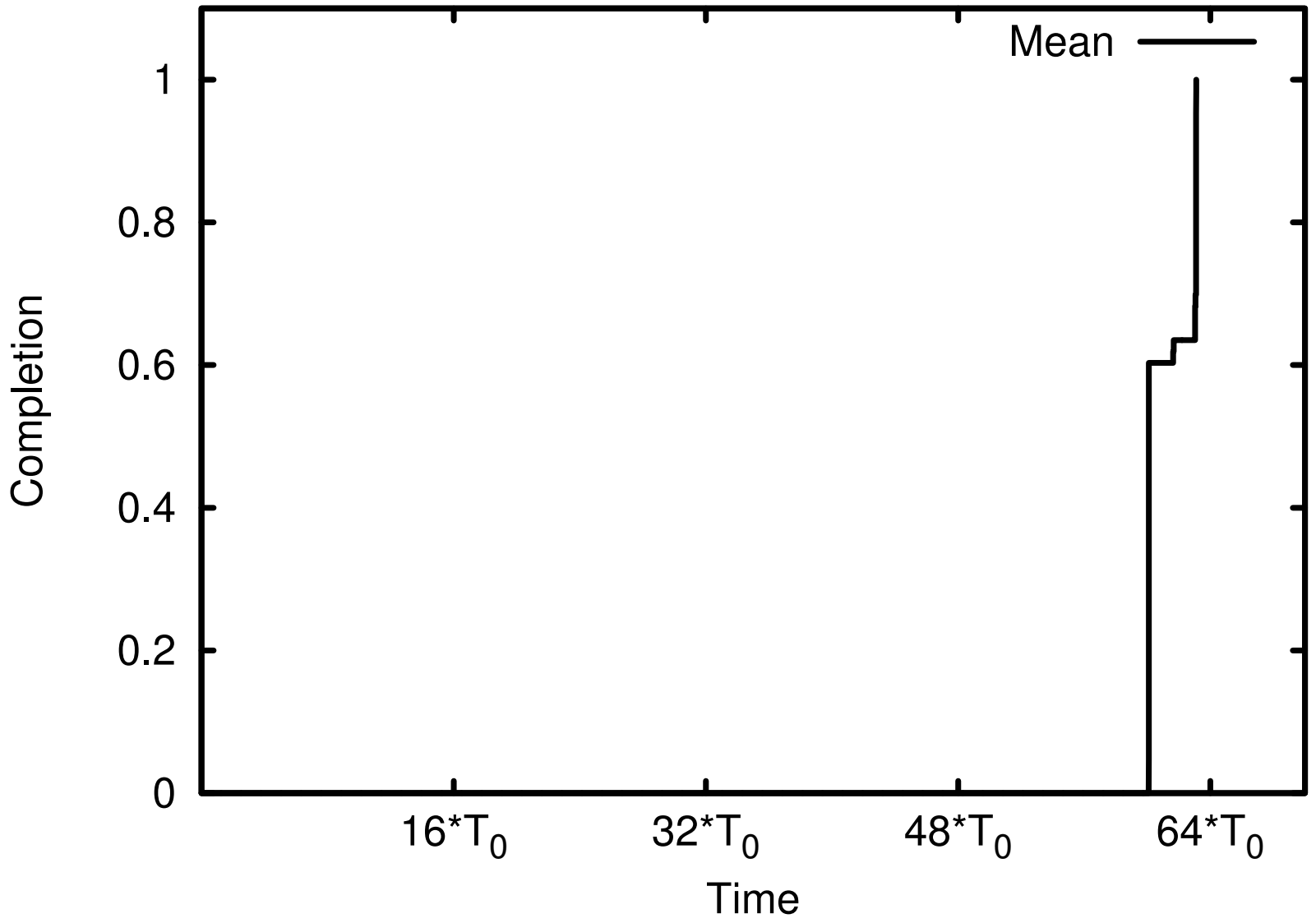
Ergebnisse - Szenario Sequential

Dieses Szenario implementiert zum Vergleich ein Client / Server System:

- Es gibt einen Super-Peer (Server) und 63 Peers (Clienten).
- Die Peers sind nicht untereinander verbunden.
- Anzahl der Chunks spielt hier keine Rolle.
- Datengröße so gewählt, dass $T_0 = 10$ Minuten gilt.

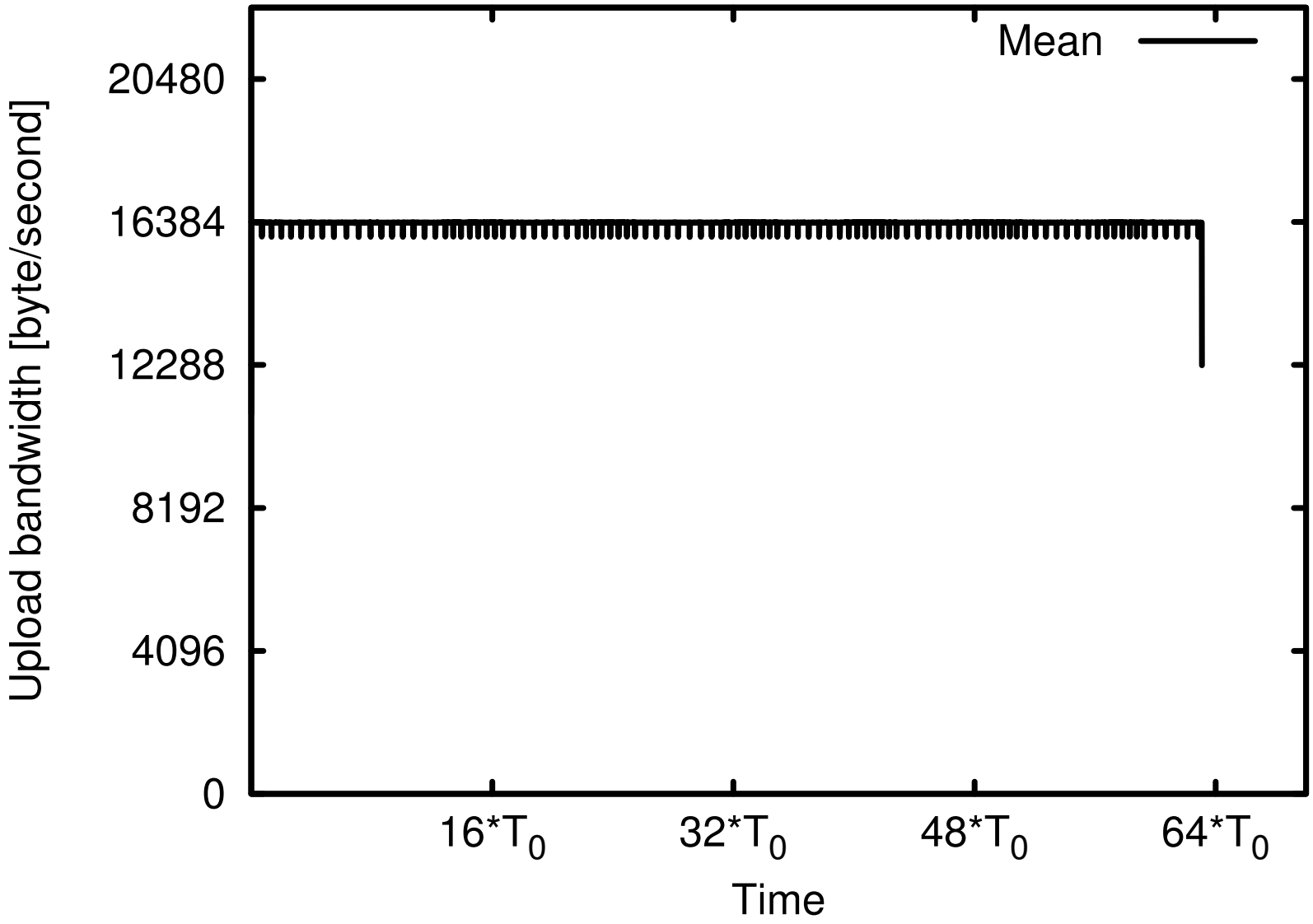
Ergebnisse - Szenario Sequential

Completion Graph:



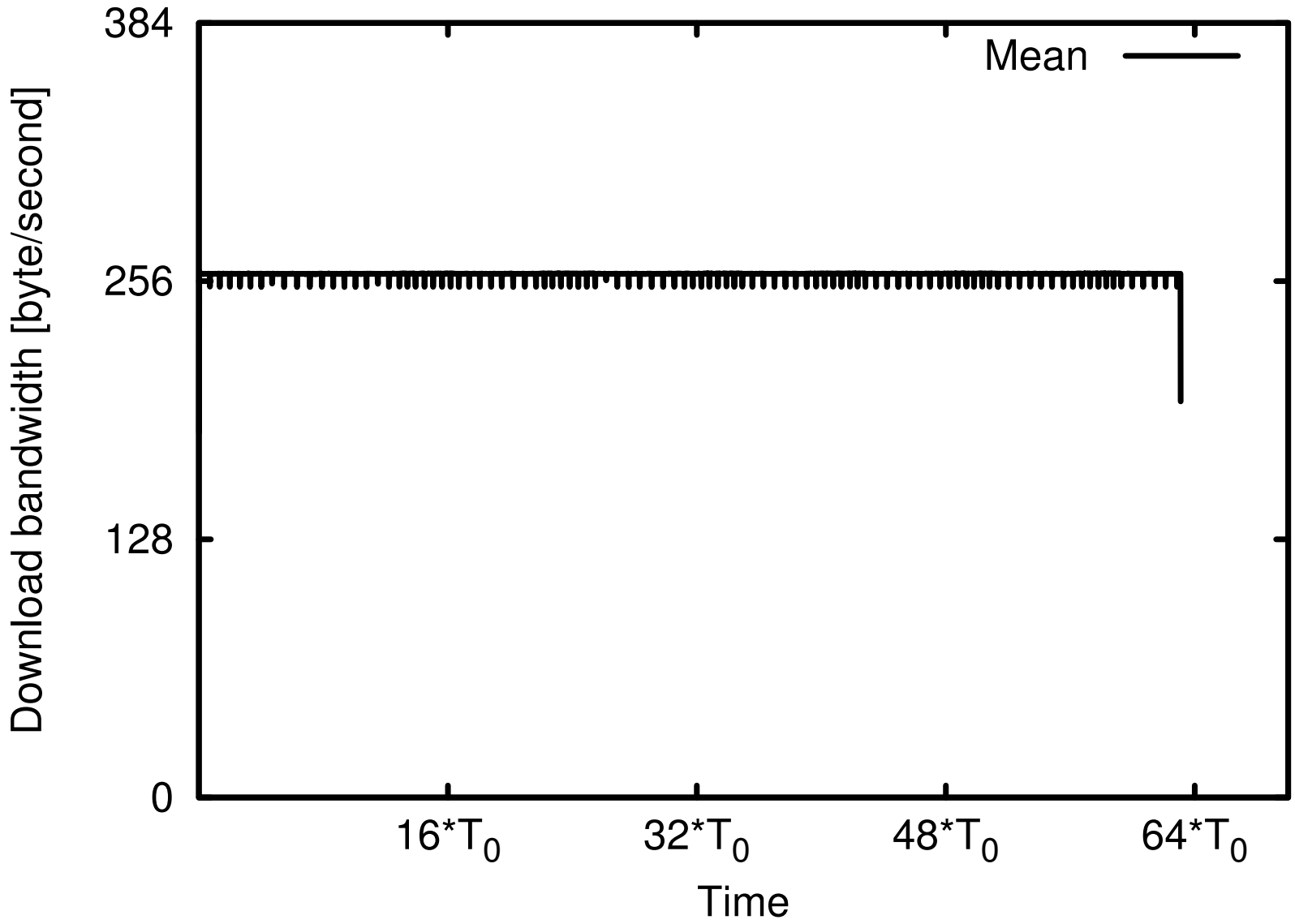
Ergebnisse - Szenario Sequential

Super-Peer Upload Bandwidth:



Ergebnisse - Szenario Sequential

Peer Download Bandwidth:



Evaluation - Fazit

- Mit Hilfe des Chunked-Swarm Verfahren kann man im Idealfall Lieferfristen weit unter $2 * T_0$ garantieren, unabhängig von der Anzahl der Peers.
- Die Anzahl der Peers sollte bekannt sein, um eine passende Anzahl an Chunks zu wählen.
- Das System skaliert nicht endlos, da die Anzahl der Verbindungen quadratisch wächst.
- Mit Hilfe von Sub-Datensätzen kann Streaming implementiert werden.

Future Work

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Einführung einer Simulationszeit, damit bei vielen Peers und sehr hoher CPU Auslastung die Messungen nicht beeinflusst werden.
- Implementierung auf Basis eines Push-Based Ansatzes. Es gäbe keine Notwendigkeit mehr für Announcements.
- Verwendung einer hierarchischen Struktur, um das Problem der quadratisch wachsenden Anzahl an Verbindungen zu verringern.

Demo

Anhang

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

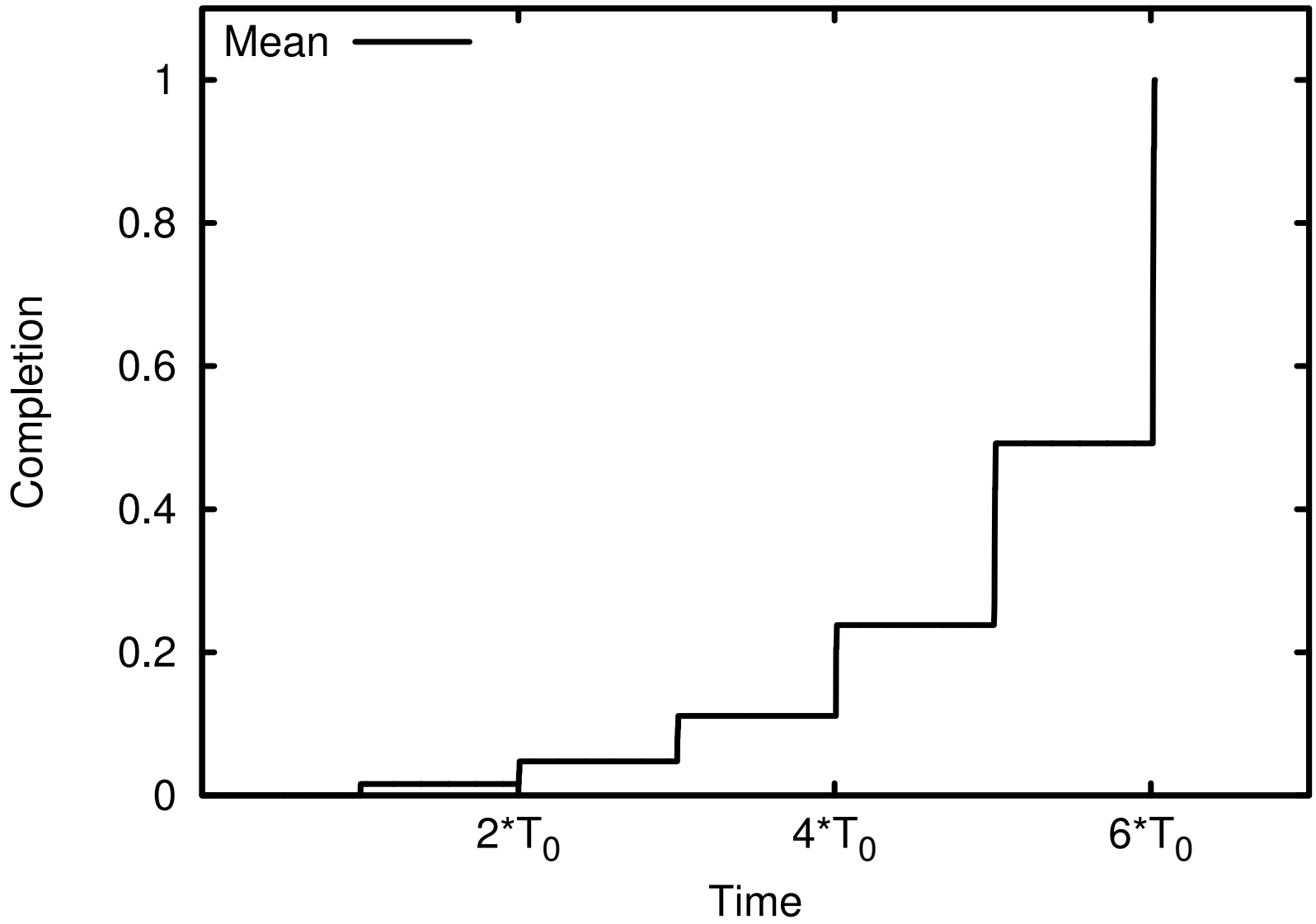
Anhang - Szenario Logarithmic

Dieses Szenario implementiert zum Vergleich ein logarithmisches Verteilungsmodell, dass jedoch ebenfalls auf einem Peer-to-Peer Netzwerk basiert:

- Es gibt einen Super-Peer (Server) und 63 Peers (Clienten).
- Jeder Peer (auch Super-Peer) darf nur an einen Peer parallel senden.
- Der Datensatz wird nur vollständig übertragen (kein Chunking).
- Anzahl der Peers, die den Datensatz ausliefern, wächst exponentiell.
- Datengröße so gewählt, dass $T_0 = 10$ Minuten gilt.

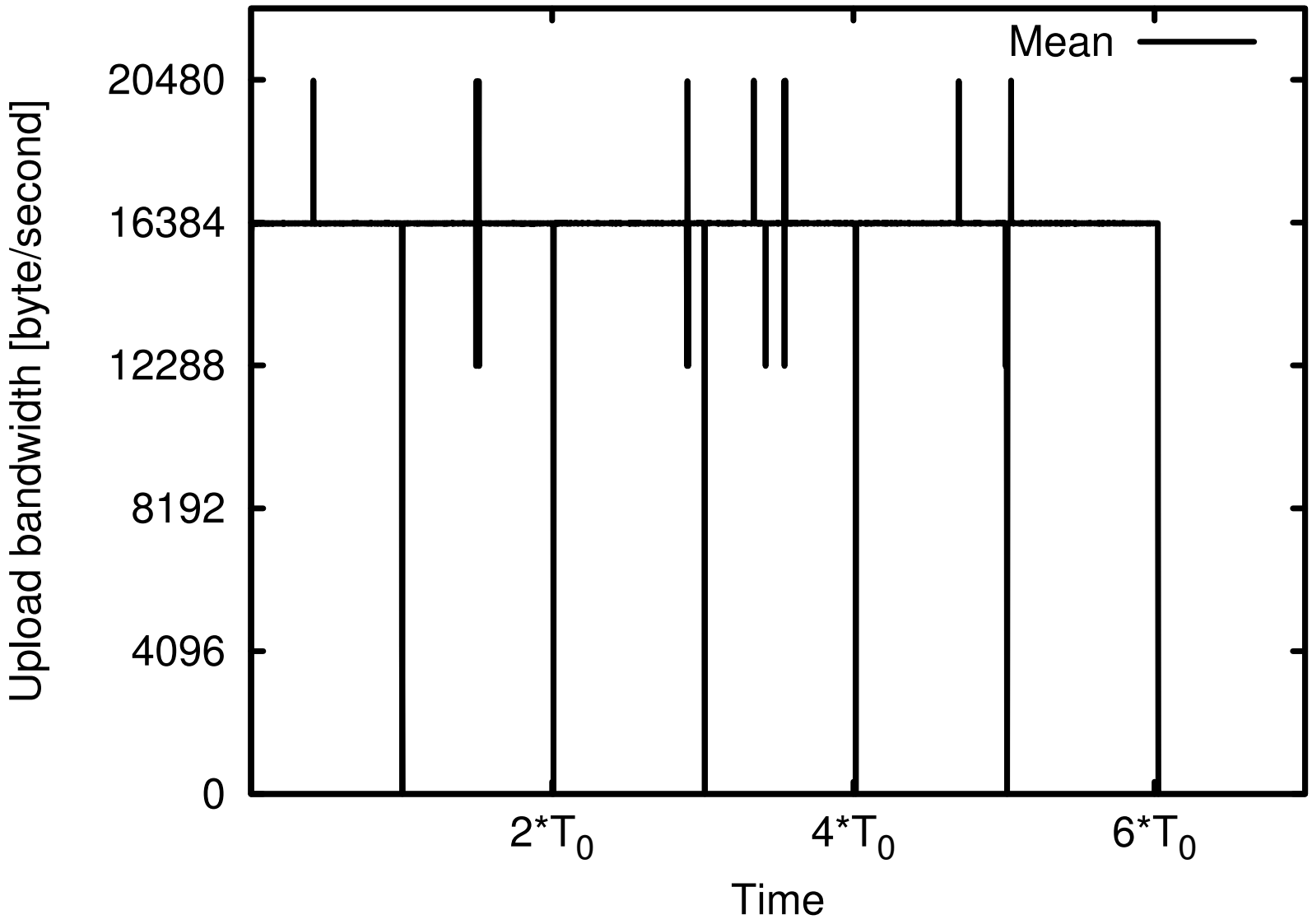
Anhang - Szenario Logarithmic

Completion Graph:



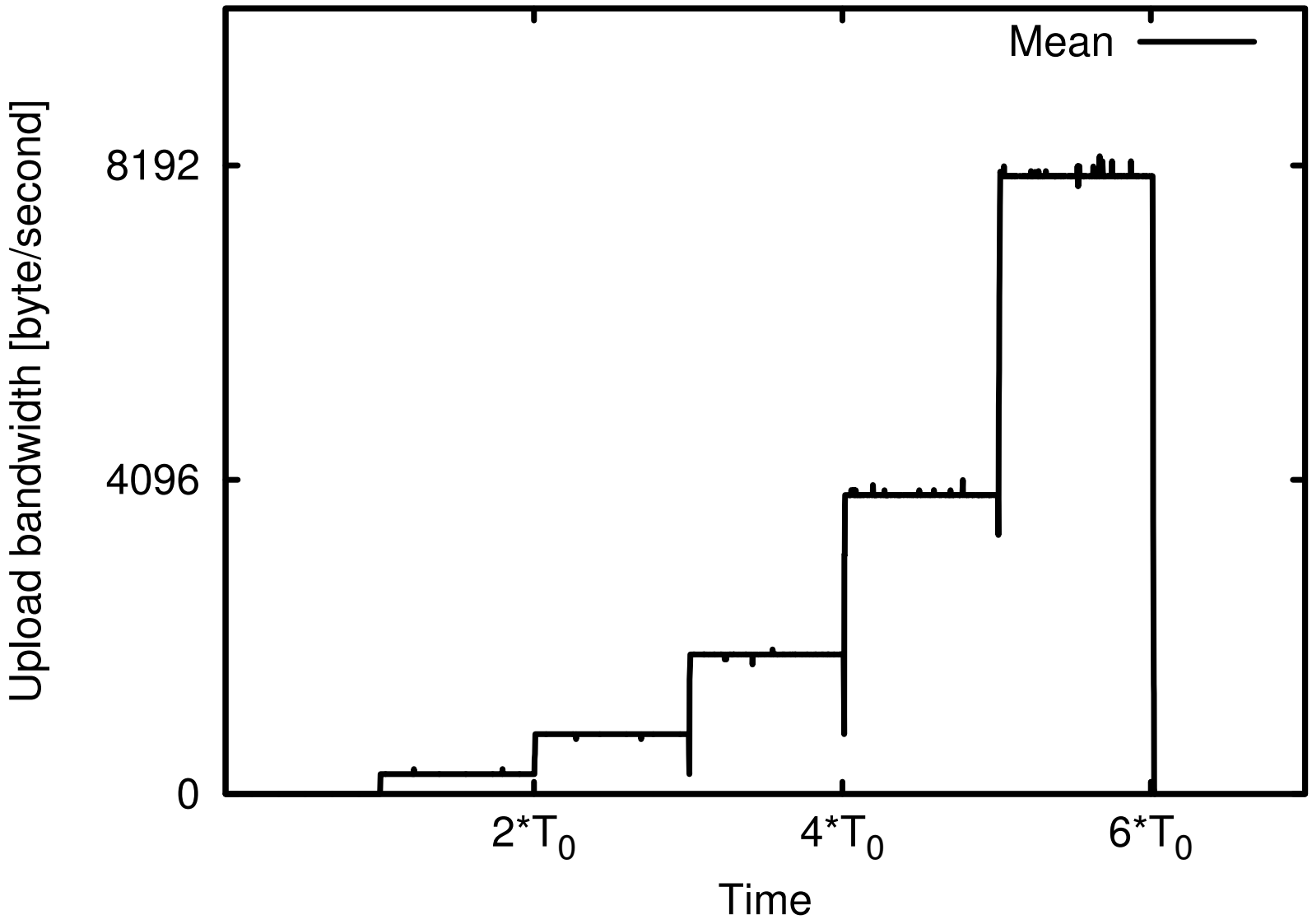
Anhang - Szenario Logarithmic

Super-Peer Upload Bandwidth:



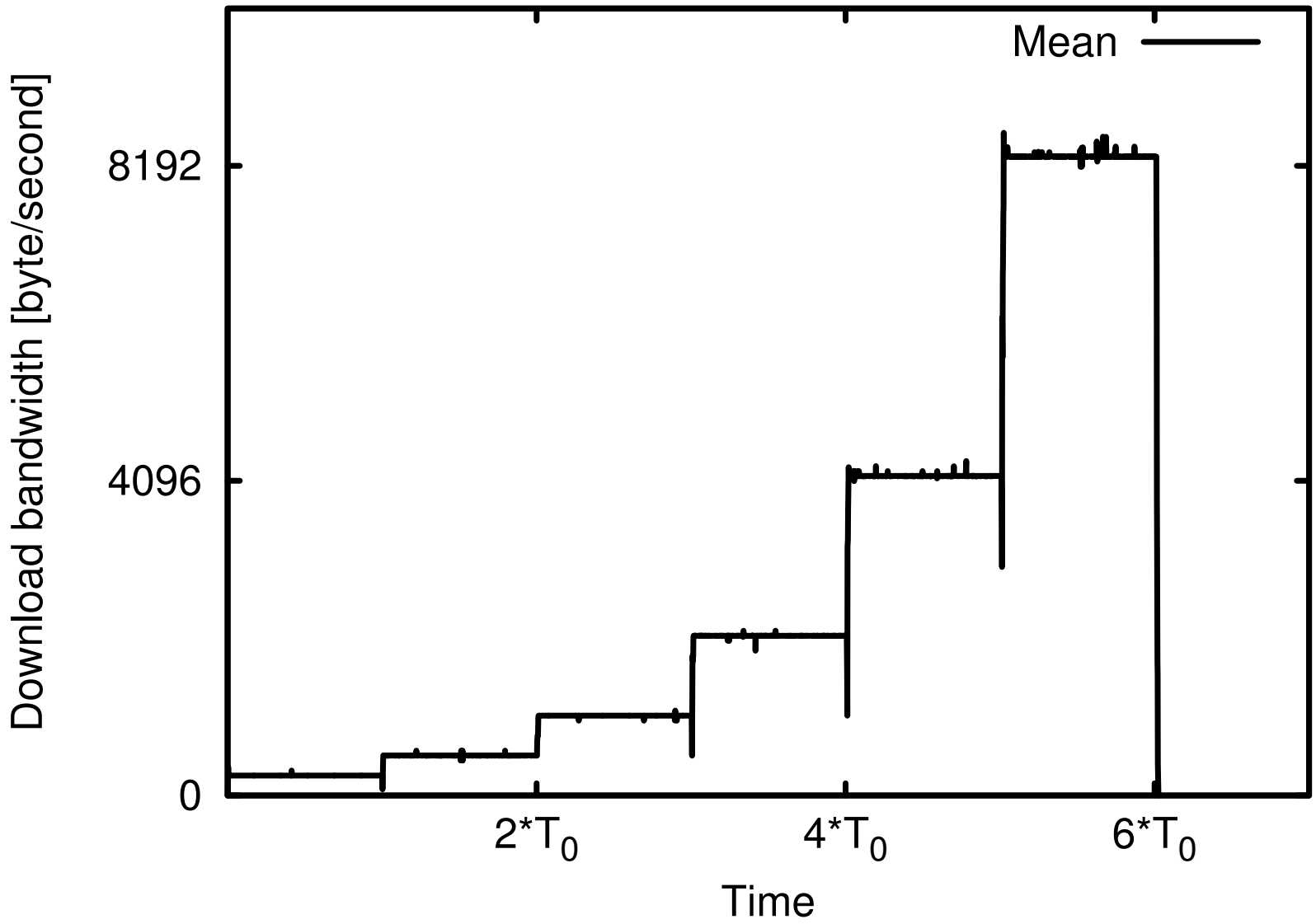
Anhang - Szenario Logarithmic

Peer Upload Bandwidth:



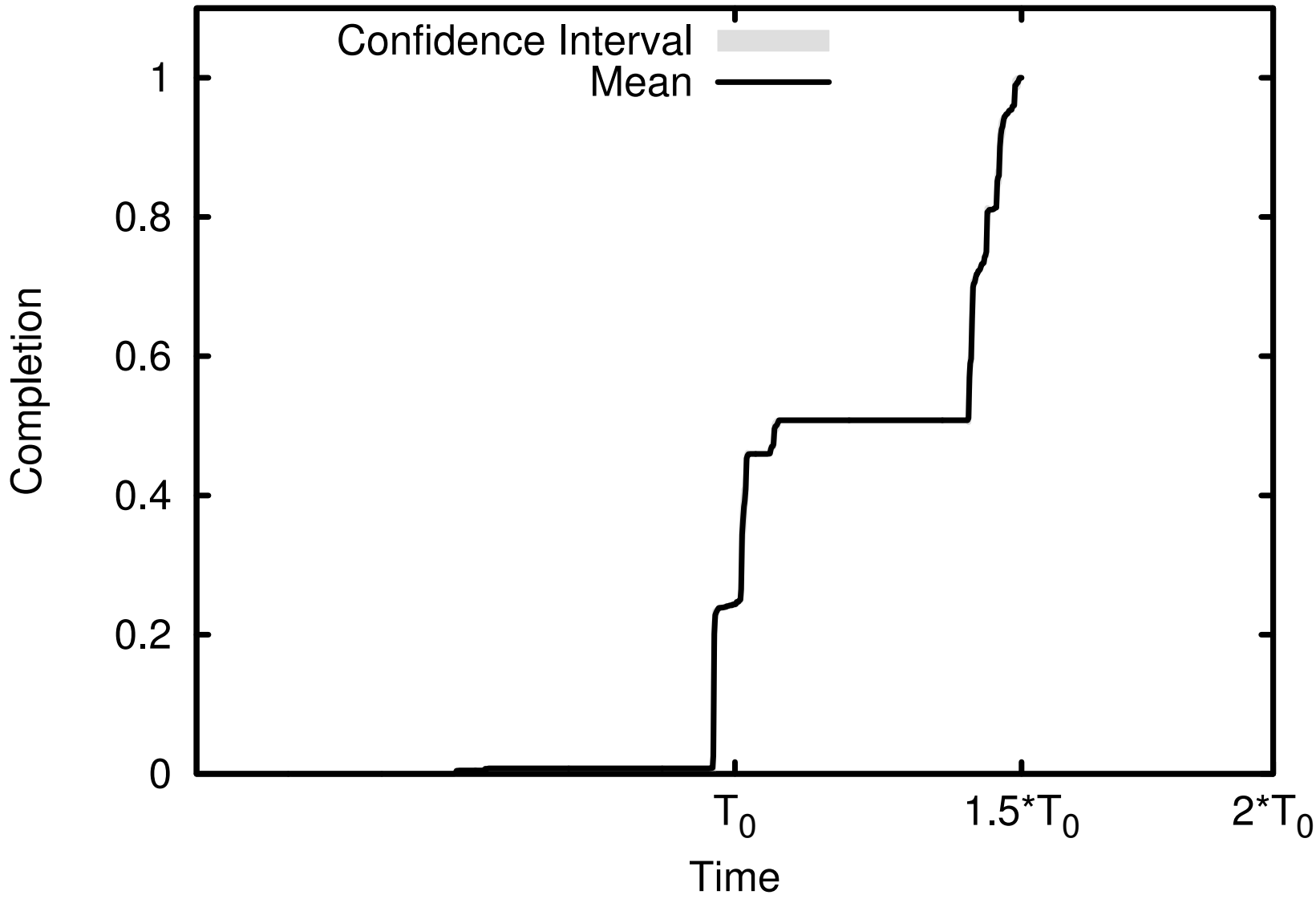
Anhang - Szenario Logarithmic

Peer Download Bandwidth:



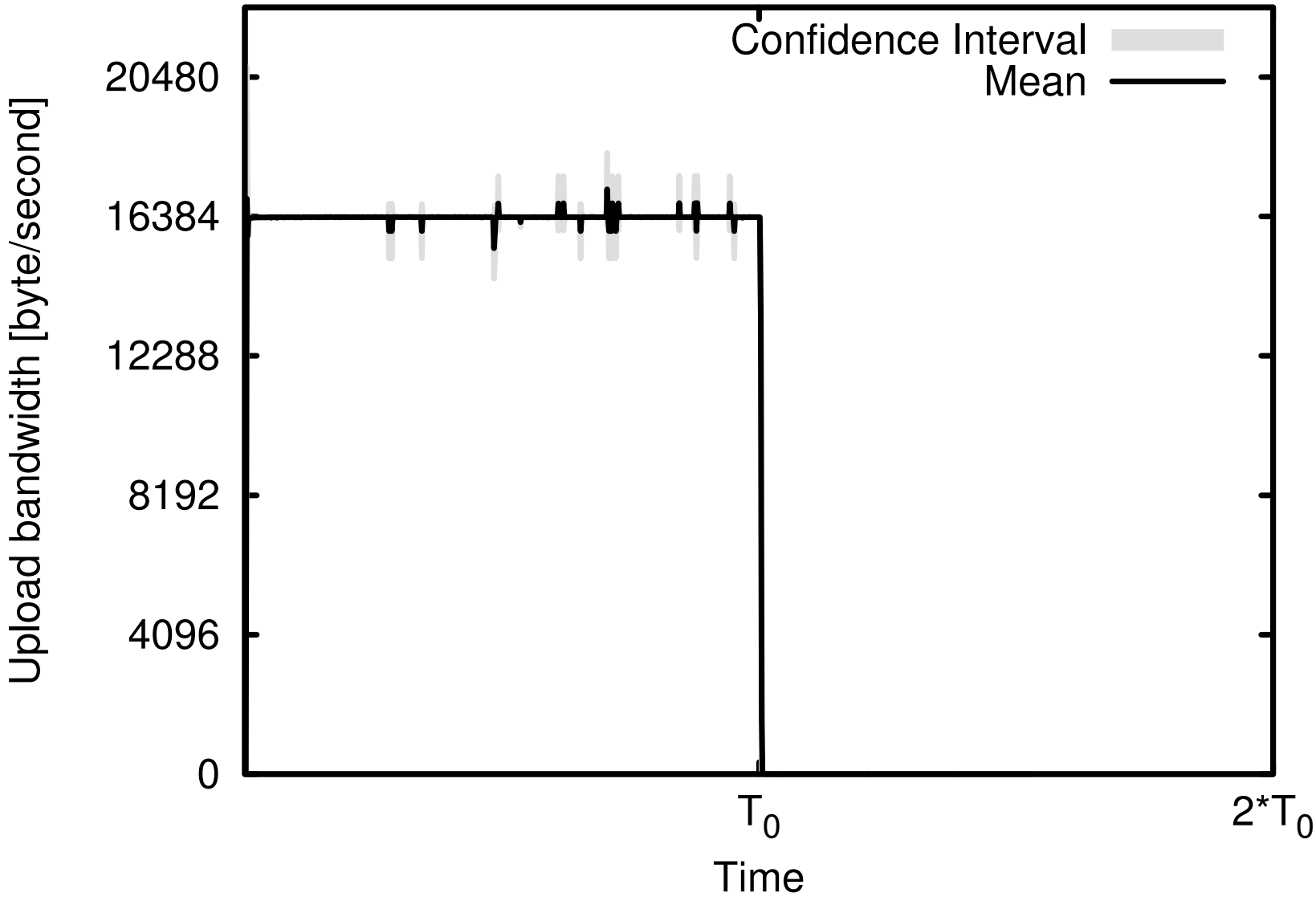
Anhang - Default Szenario mit Meta-Daten 0

Completion Graph:



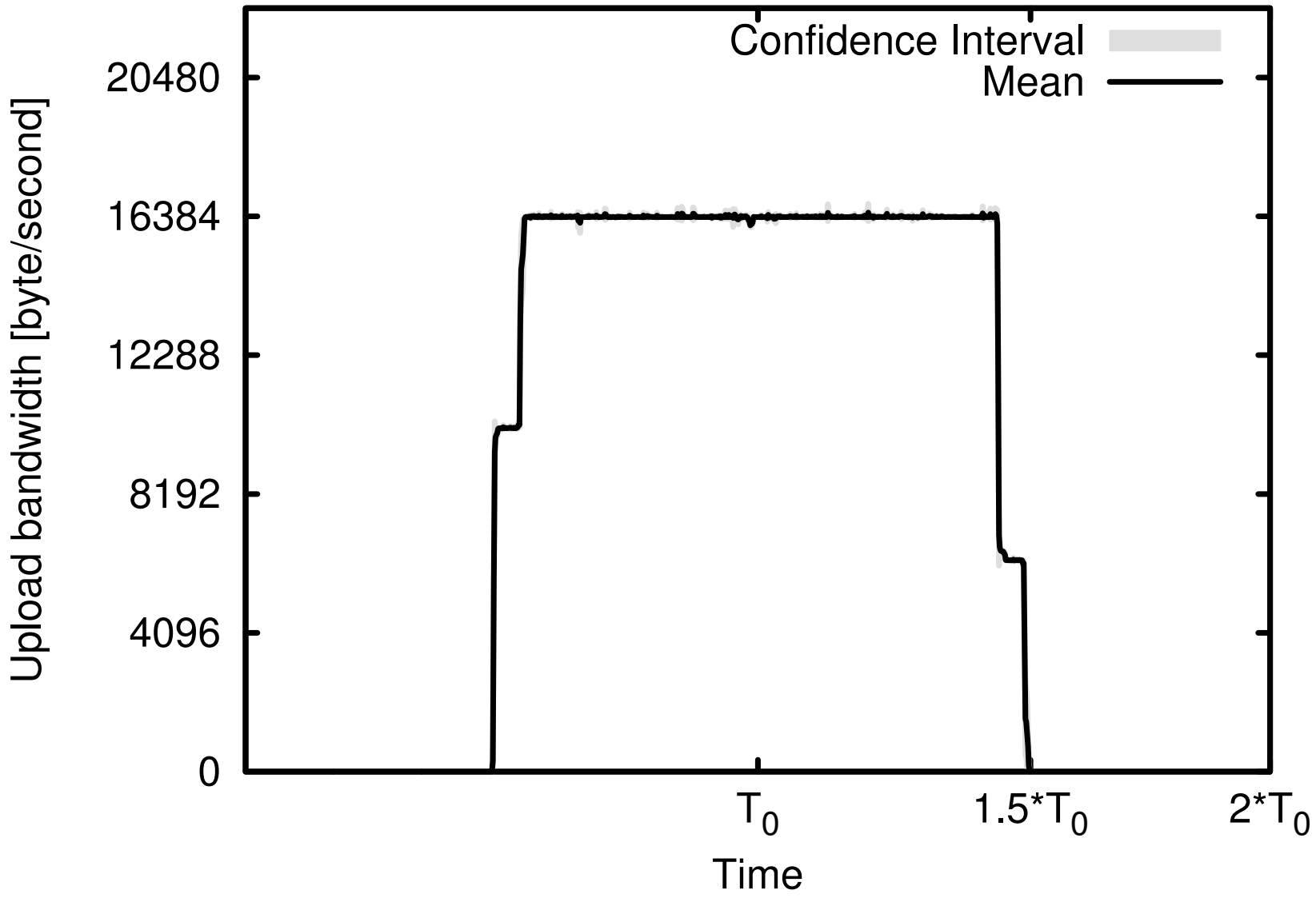
Anhang - Default Szenario mit Meta-Daten 0

Super-Peer Upload Bandwidth:



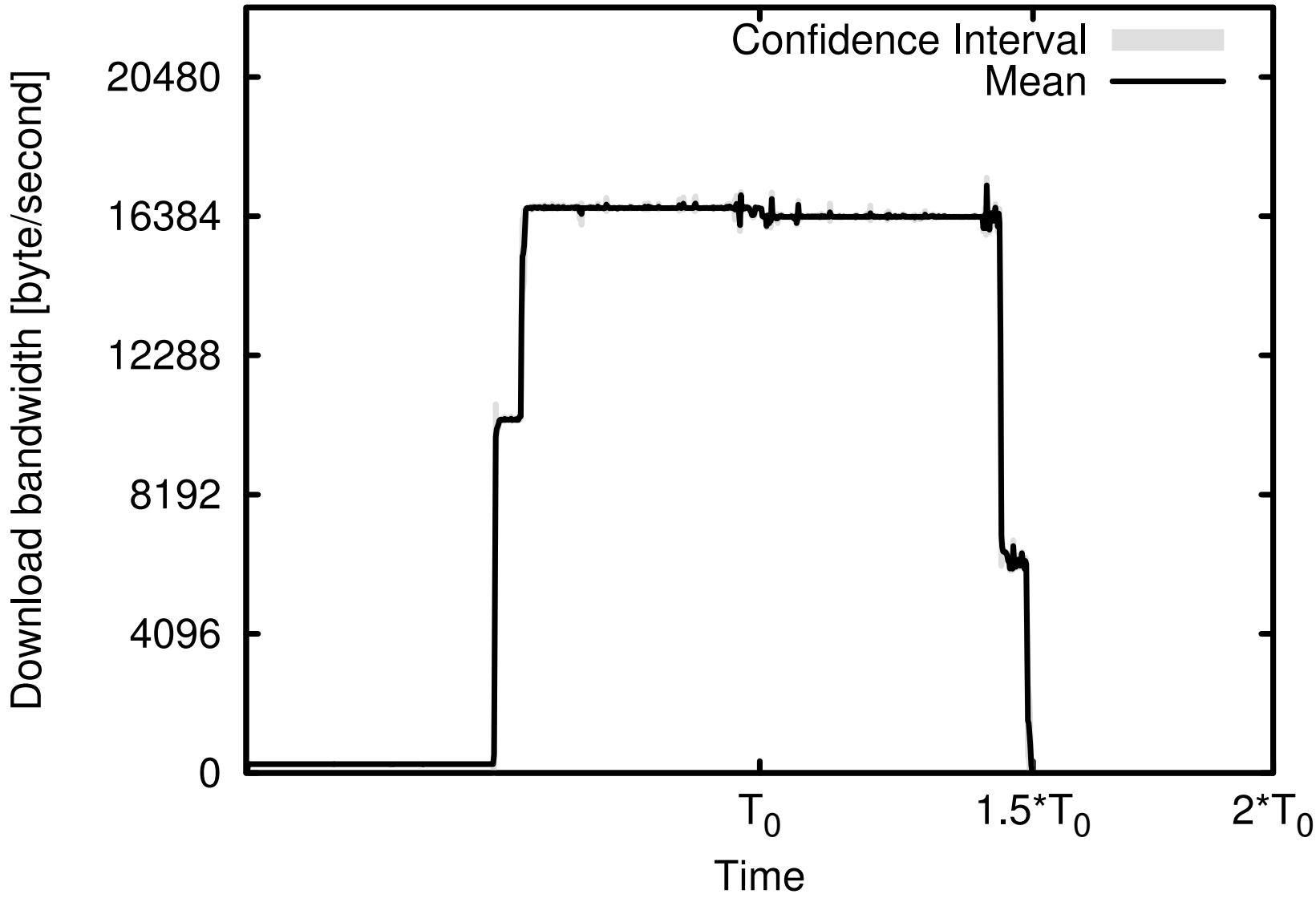
Anhang - Default Szenario mit Meta-Daten 0

Peer Upload Bandwidth:



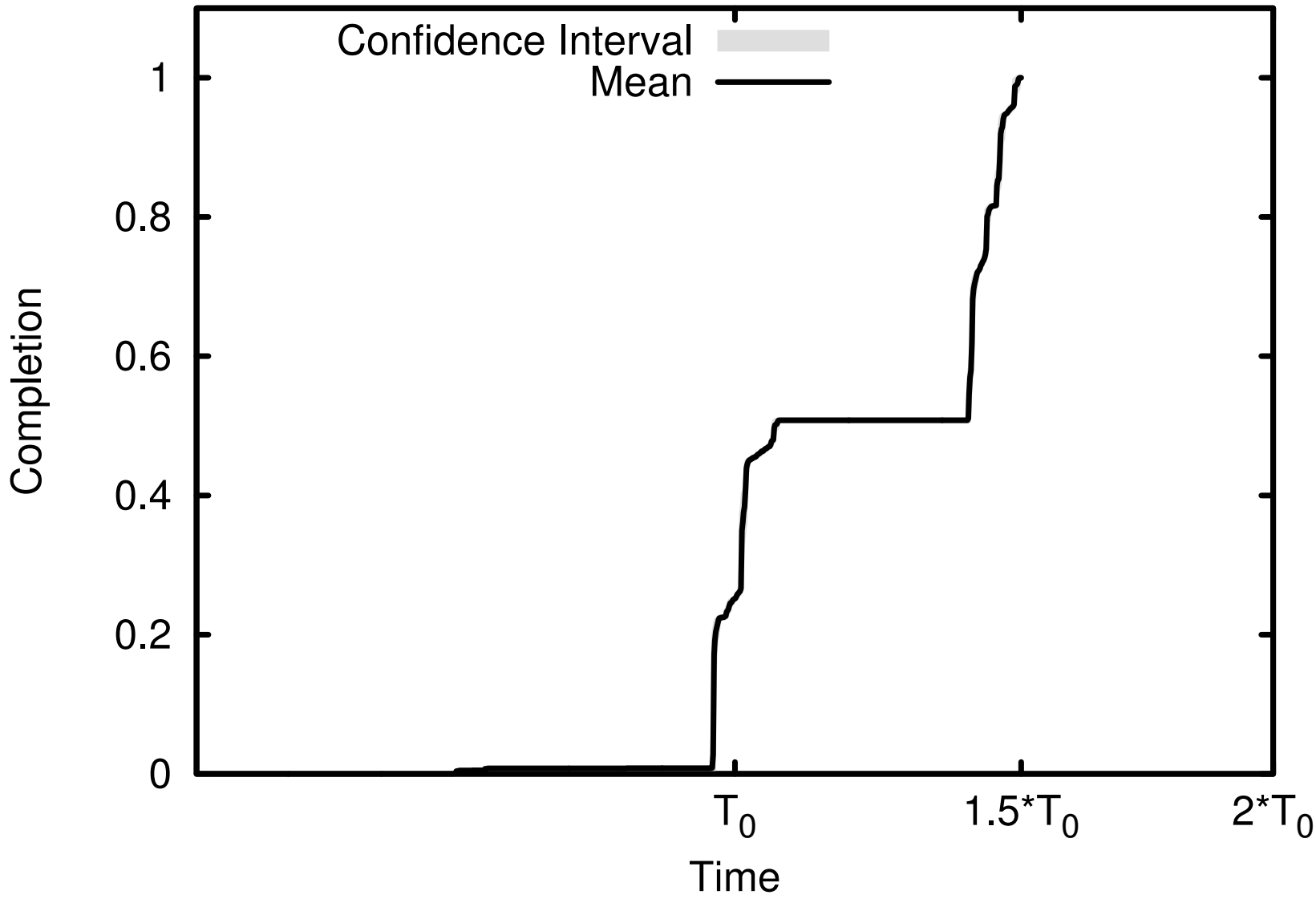
Anhang - Default Szenario mit Meta-Daten 0

Peer Download Bandwidth:



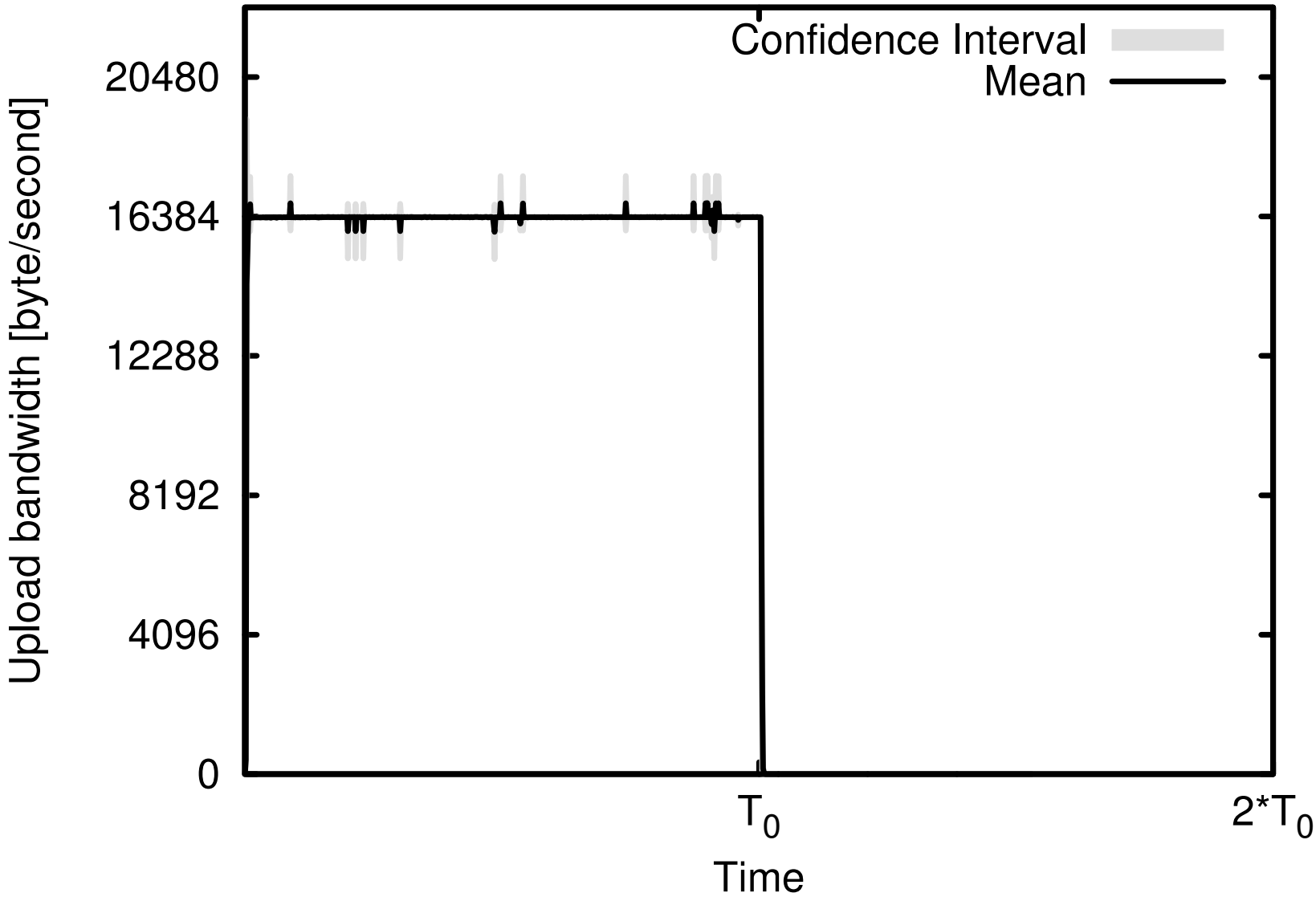
Anhang - Default Szenario mit Meta-Daten 10

Completion Graph:



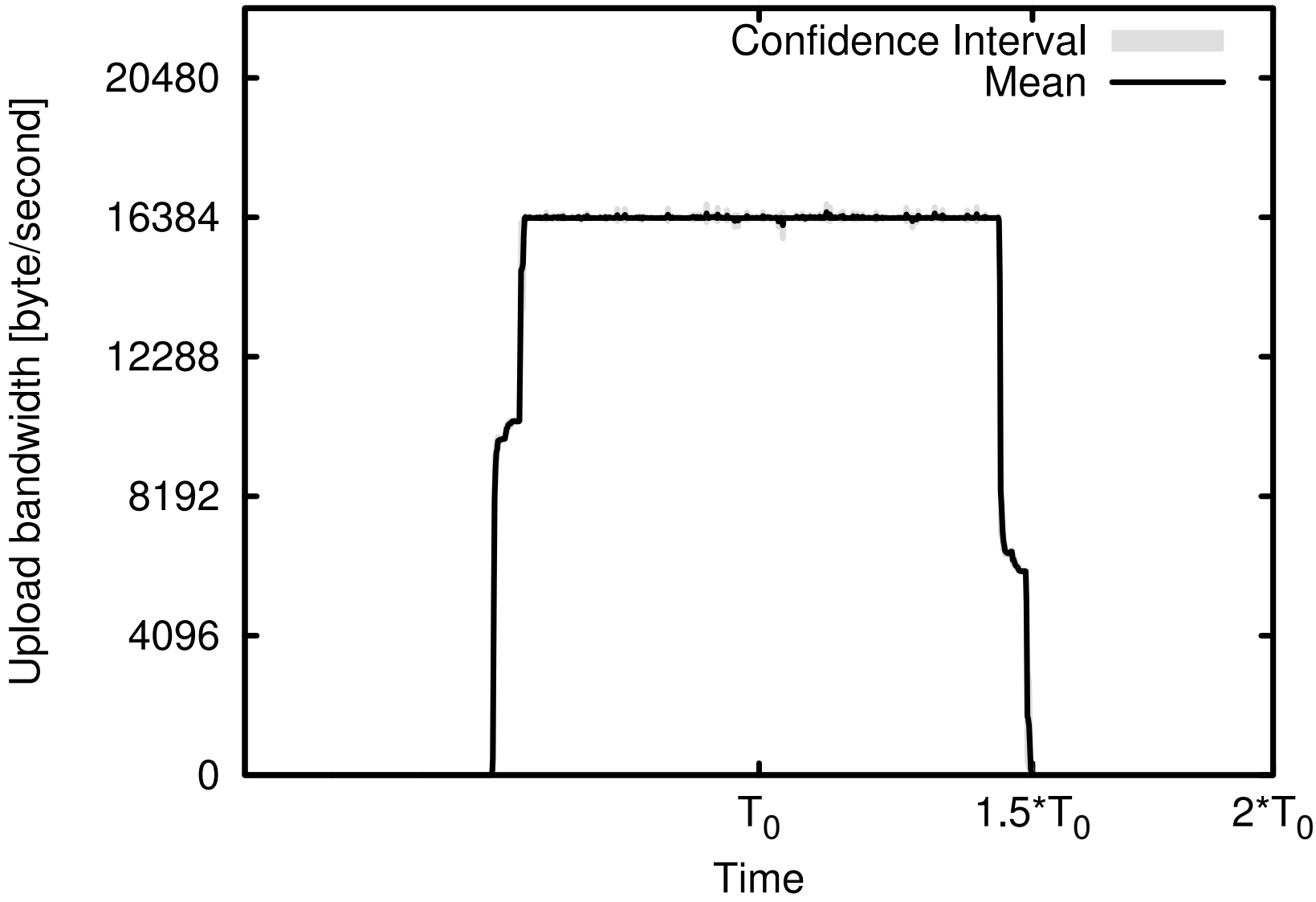
Anhang - Default Szenario mit Meta-Daten 10

Super-Peer Upload Bandwidth:



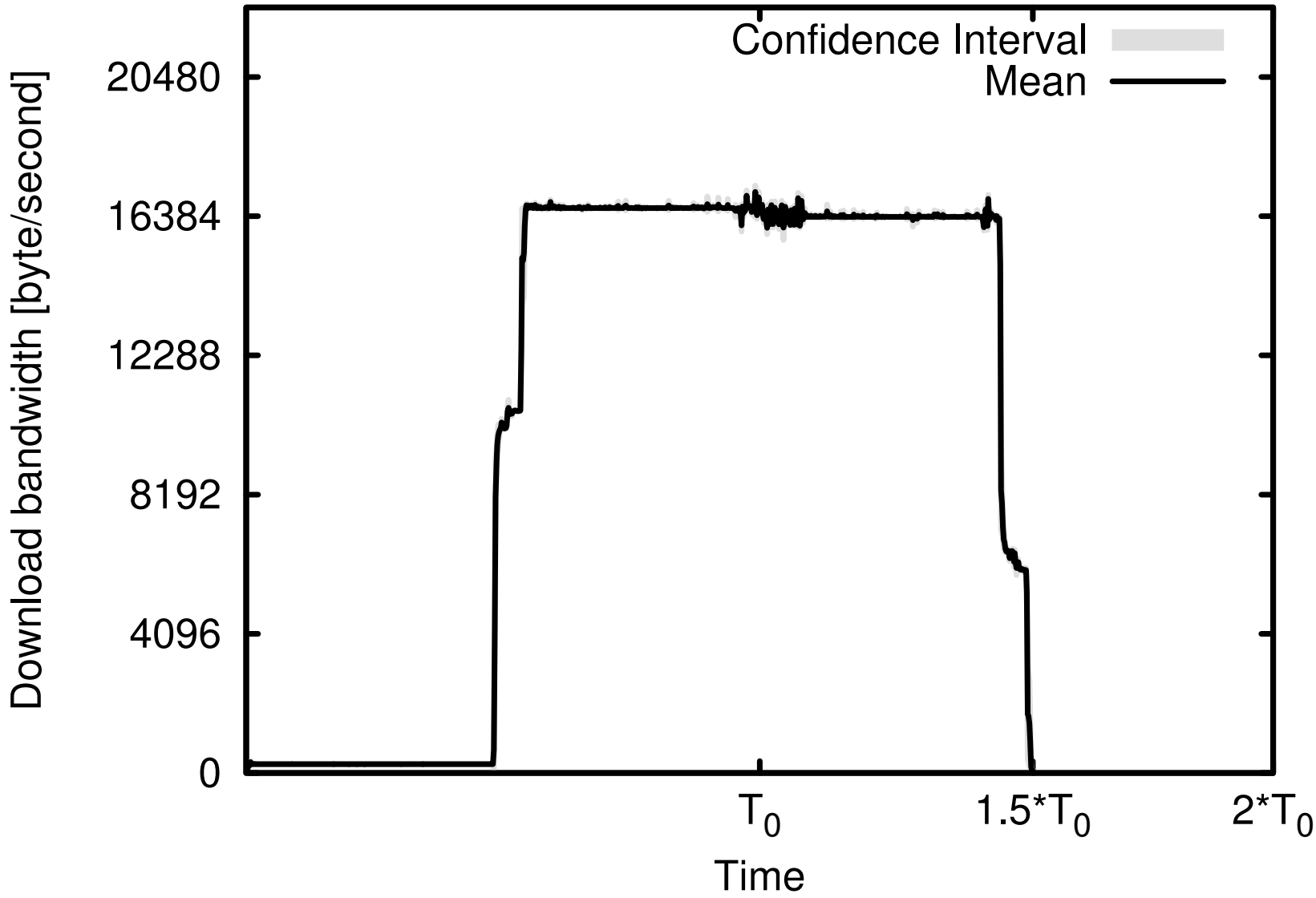
Anhang - Default Szenario mit Meta-Daten 10

Peer Upload Bandwidth:



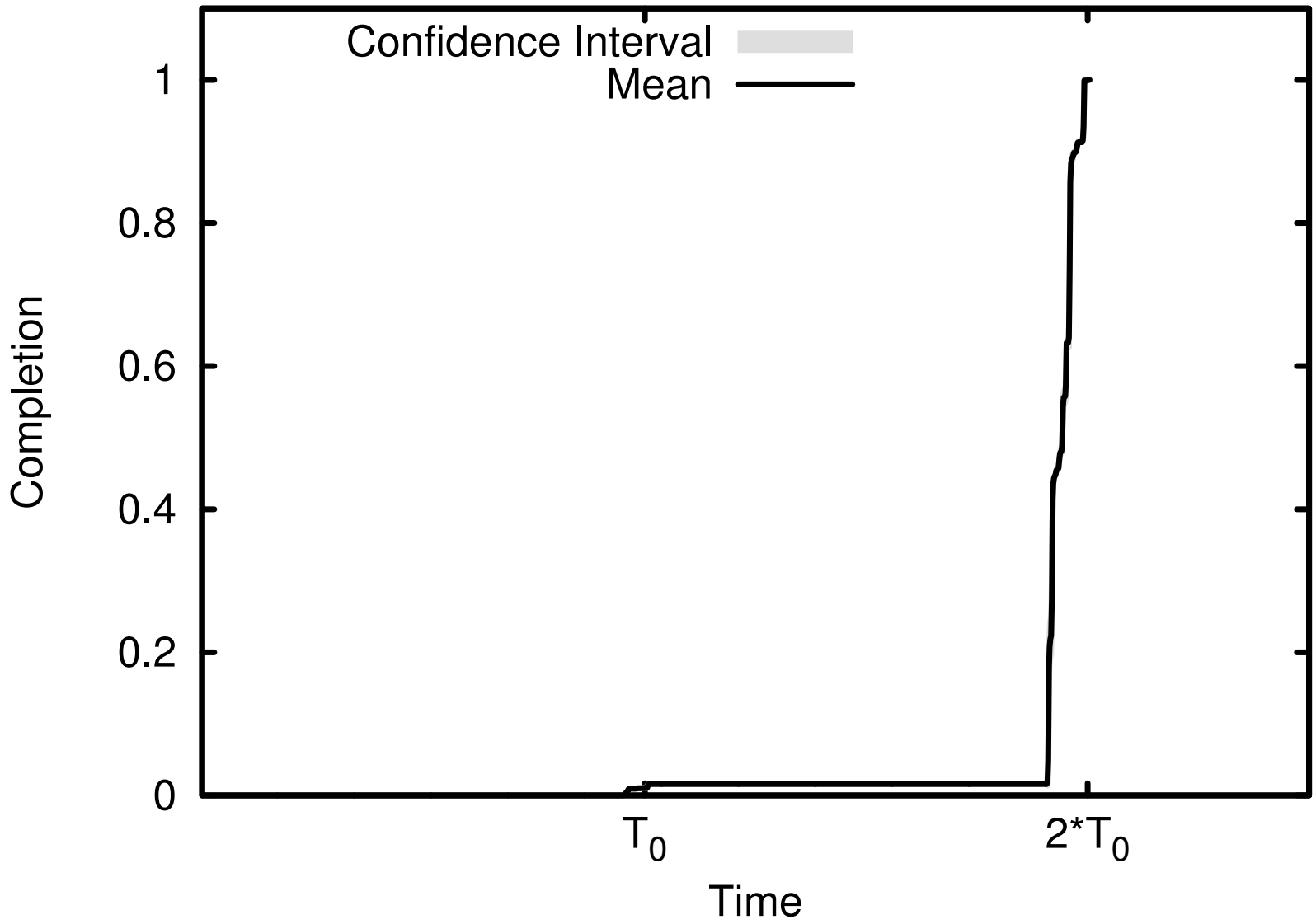
Anhang - Default Szenario mit Meta-Daten 10

Peer Download Bandwidth:



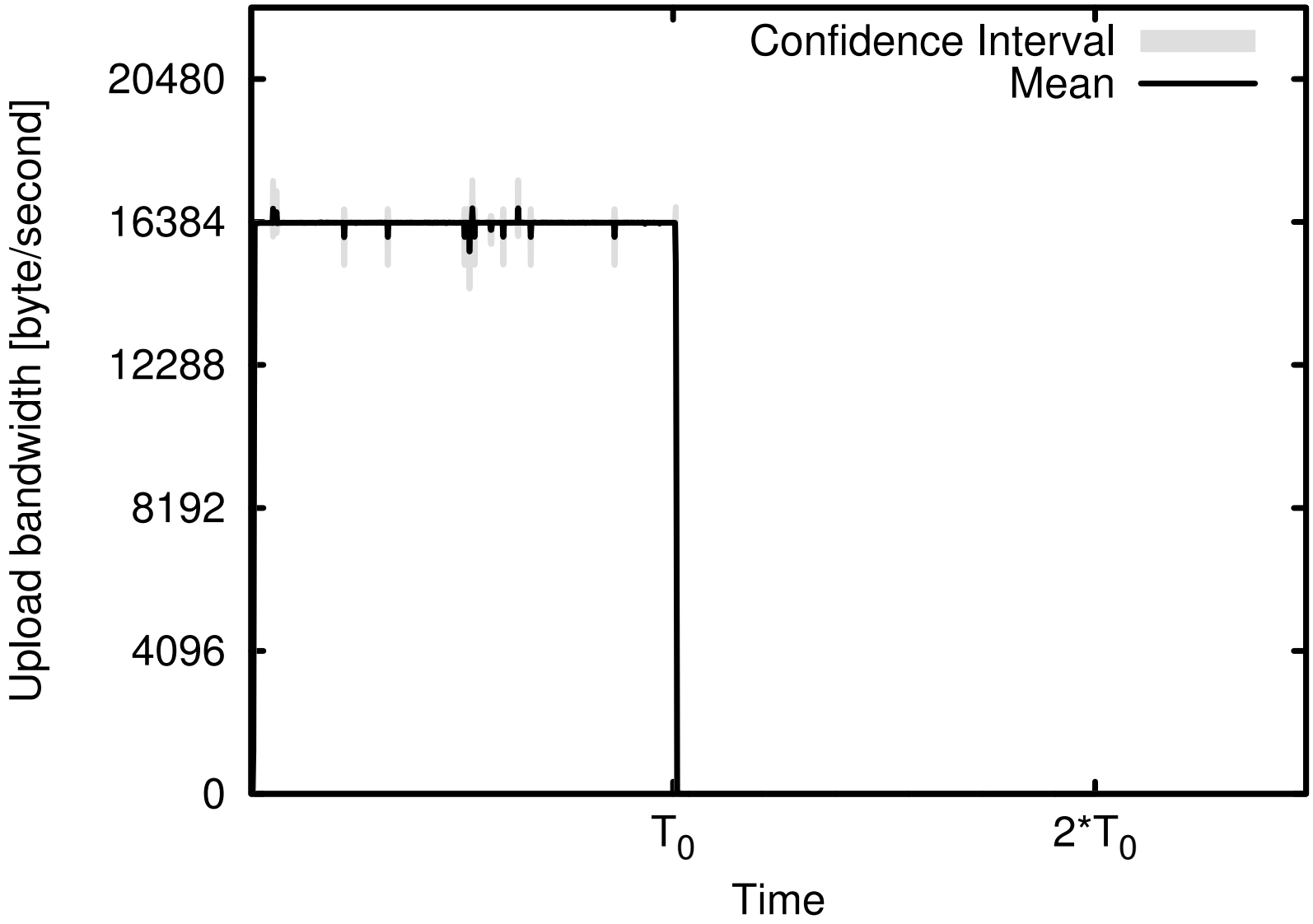
Anhang - Default Szenario mit 1x Chunkanzahl

Completion Graph:



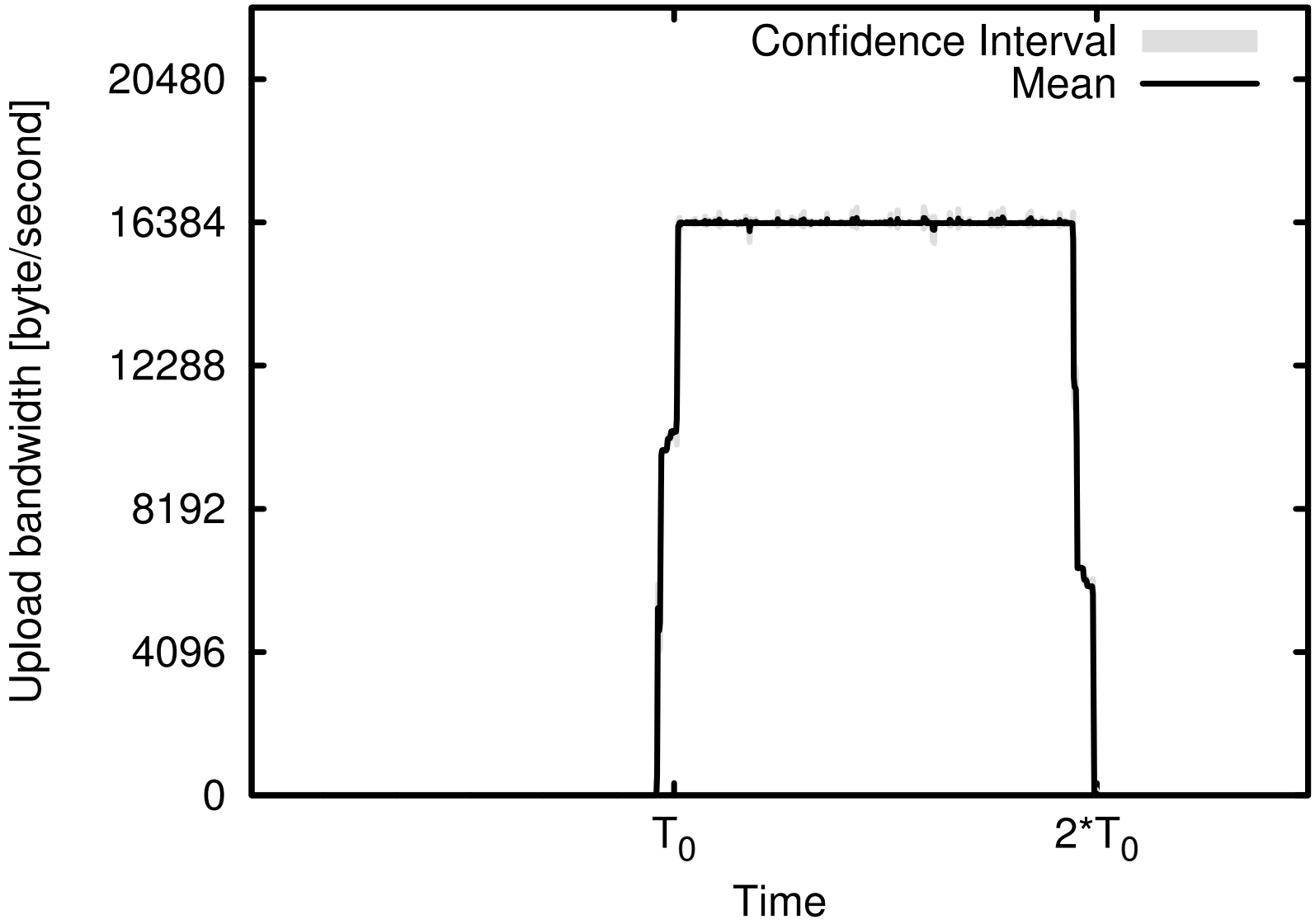
Anhang - Default Szenario mit 1x Chunkanzahl

Super-Peer Upload Bandwidth:



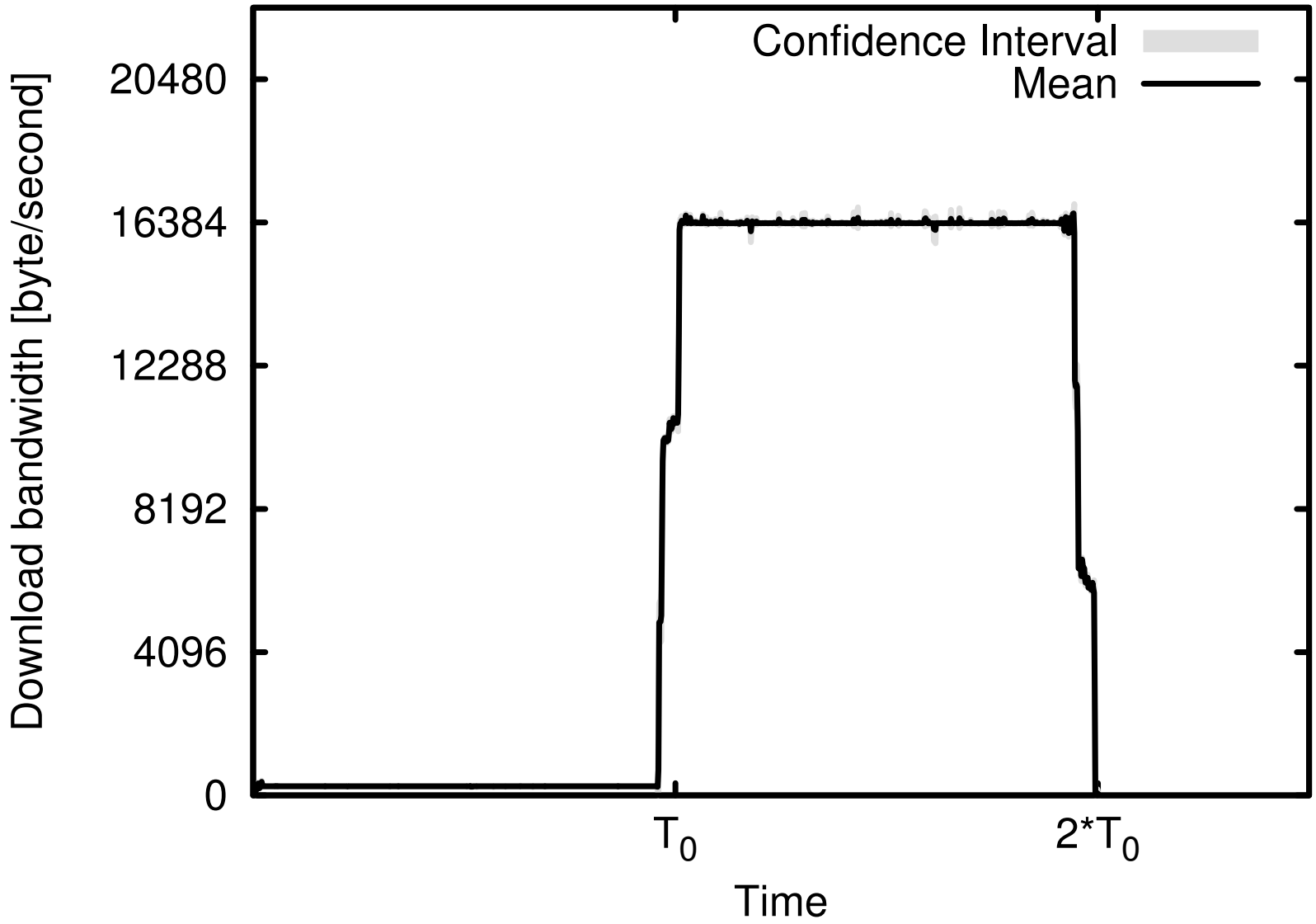
Anhang - Default Szenario mit 1x Chunkanzahl

Peer Upload Bandwidth:



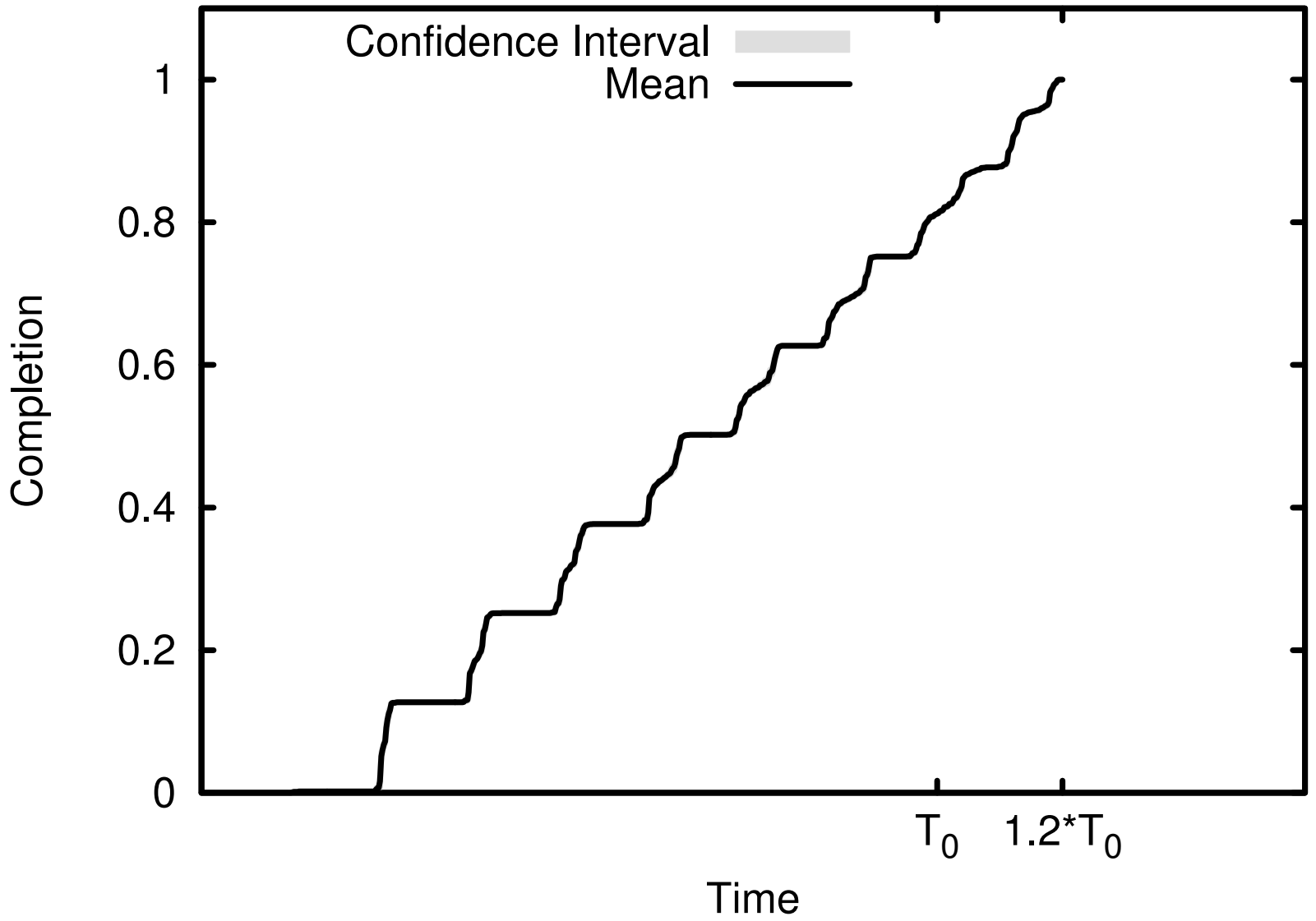
Anhang - Default Szenario mit 1x Chunkanzahl

Peer Download Bandwidth:



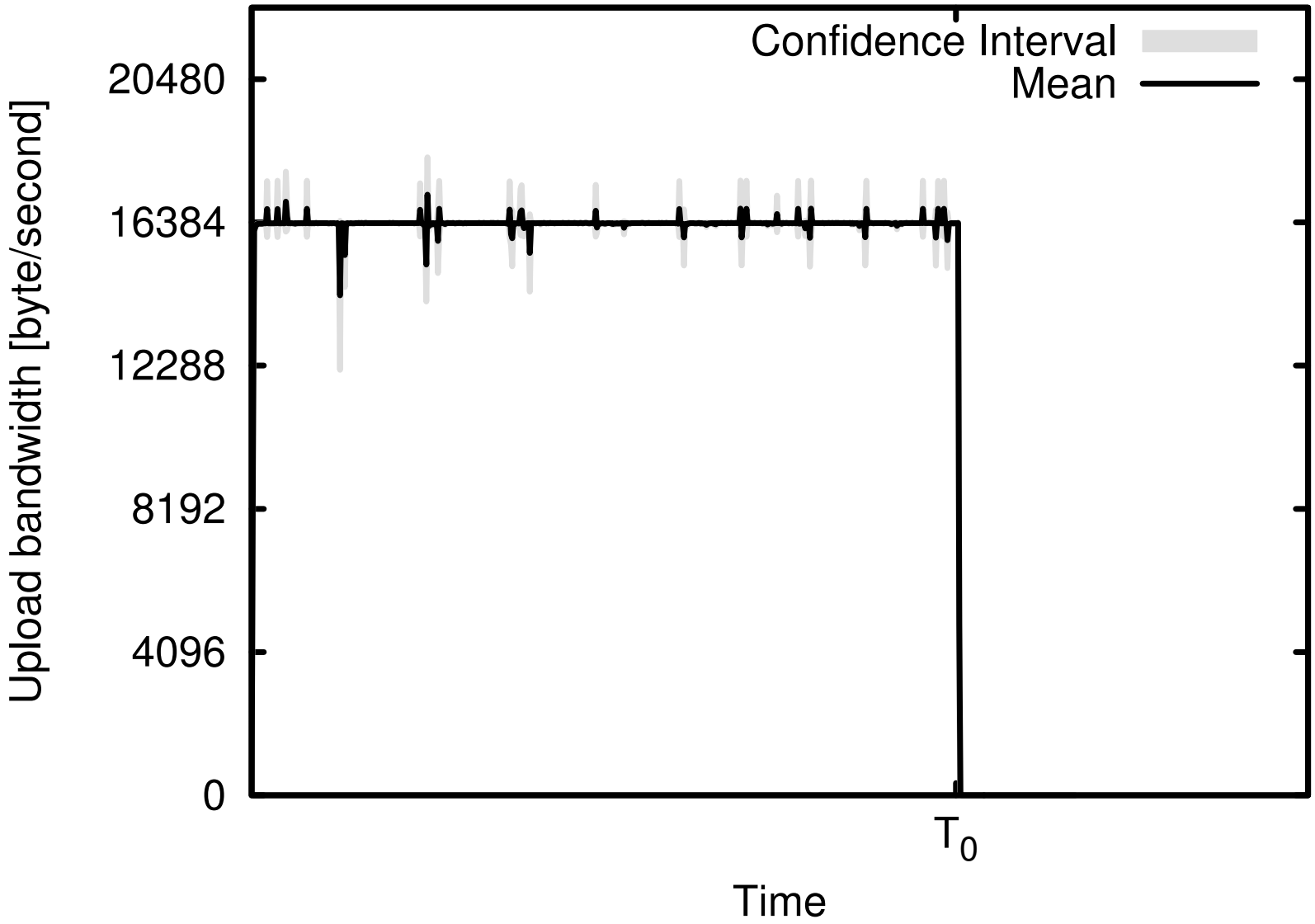
Anhang - Default Szenario mit 8x Chunkanzahl

Completion Graph:



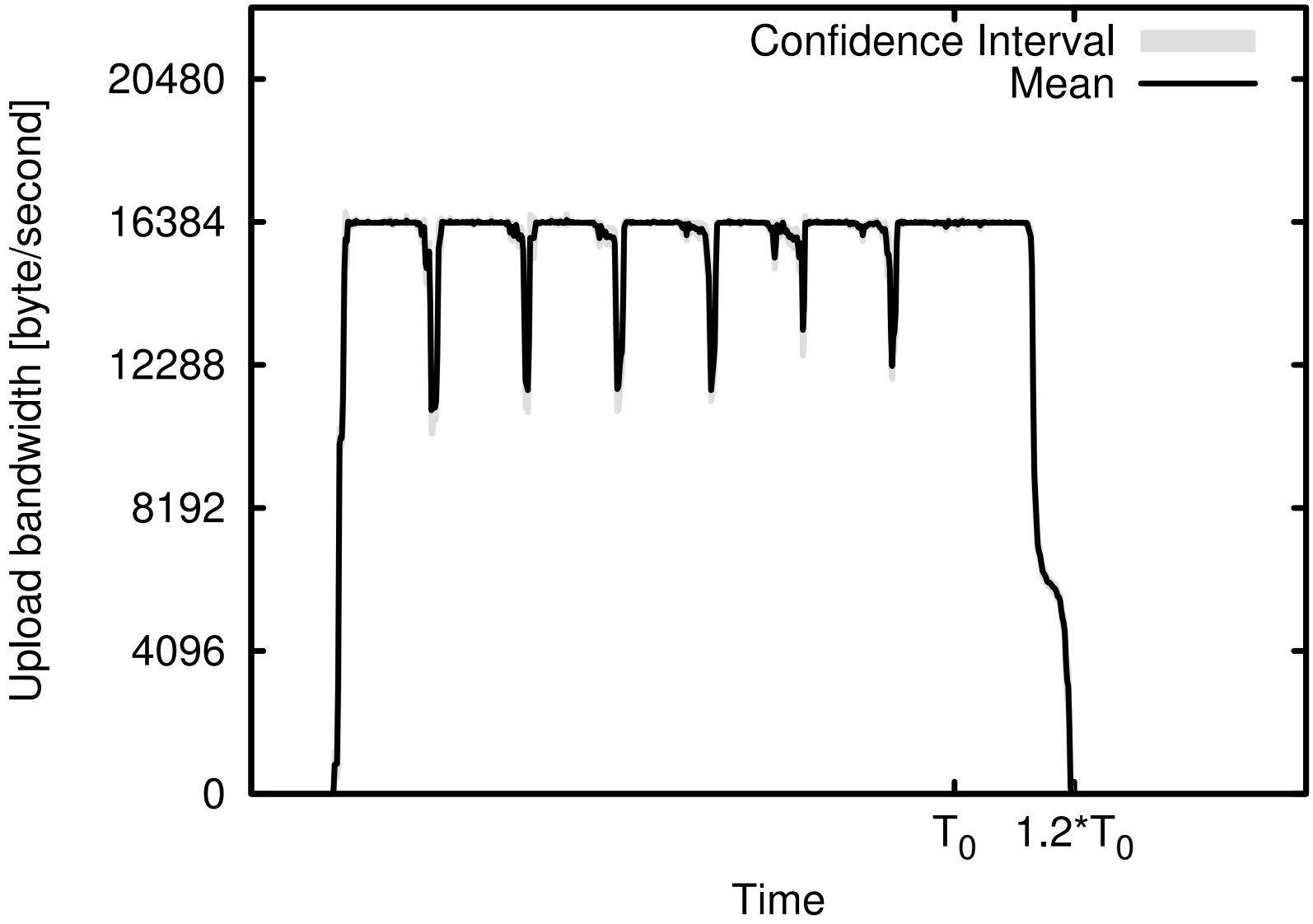
Anhang - Default Szenario mit 8x Chunkanzahl

Super-Peer Upload Bandwidth:



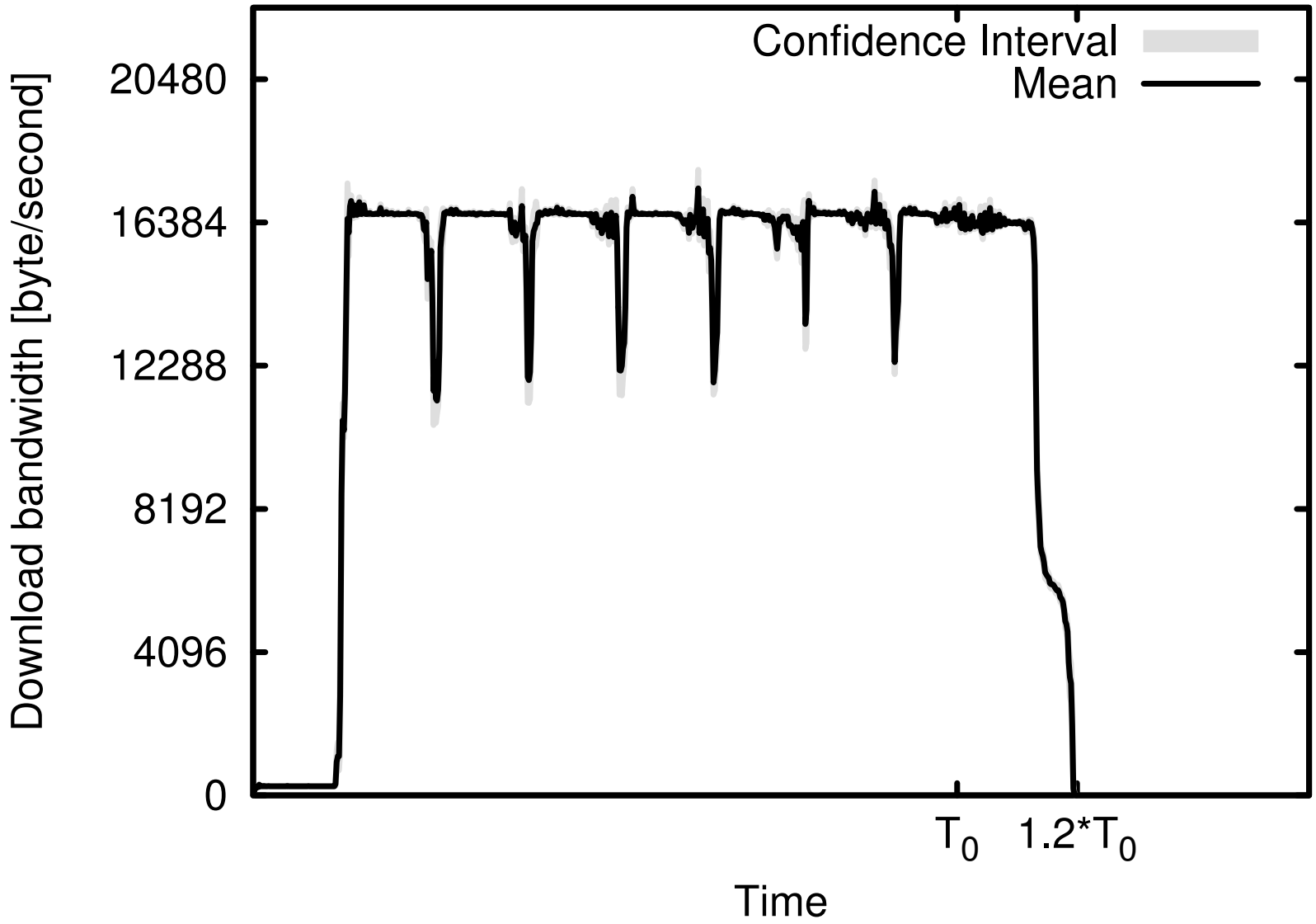
Anhang - Default Szenario mit 8x Chunkanzahl

Peer Upload Bandwidth:



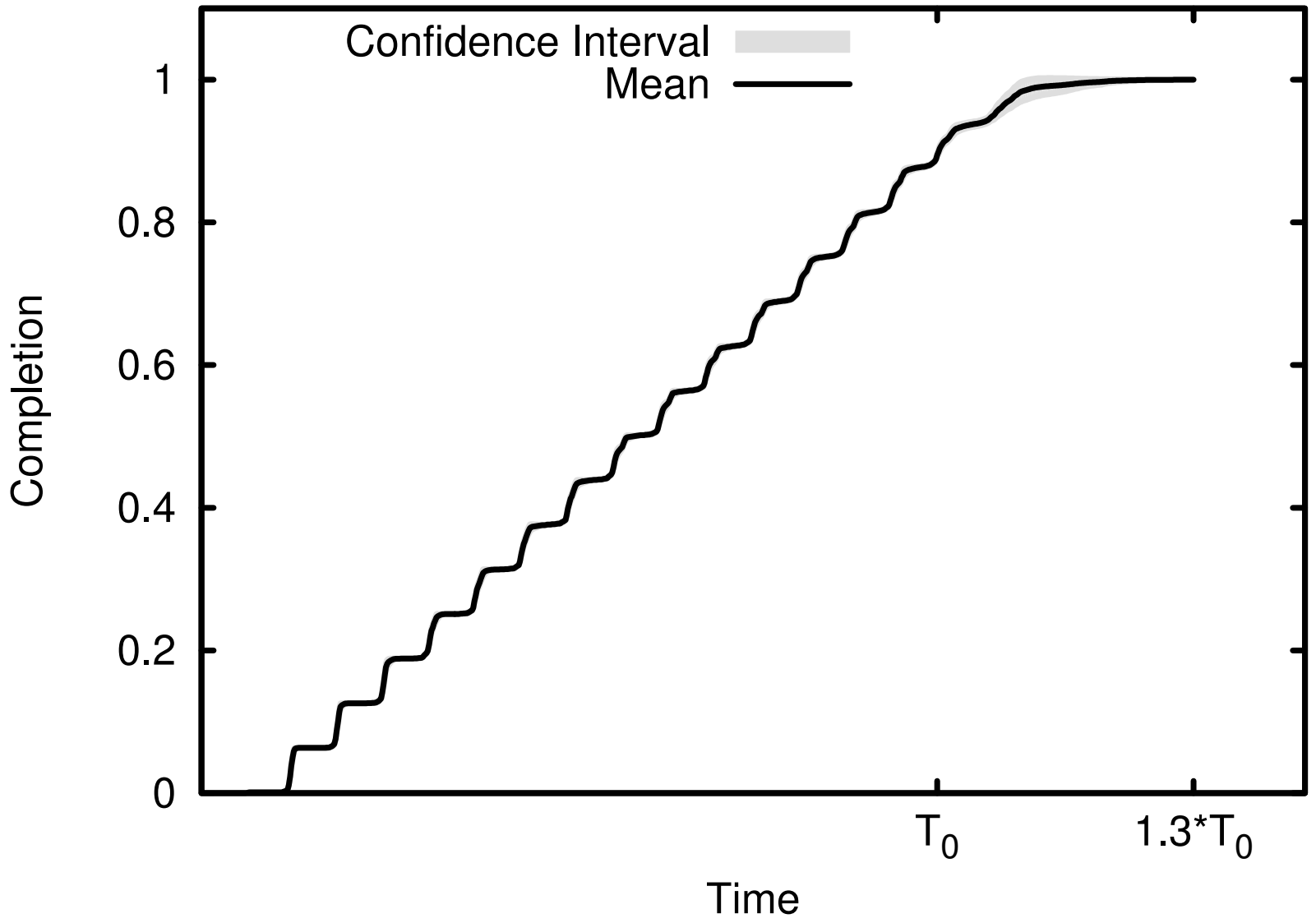
Anhang - Default Szenario mit 8x Chunkanzahl

Peer Download Bandwidth:



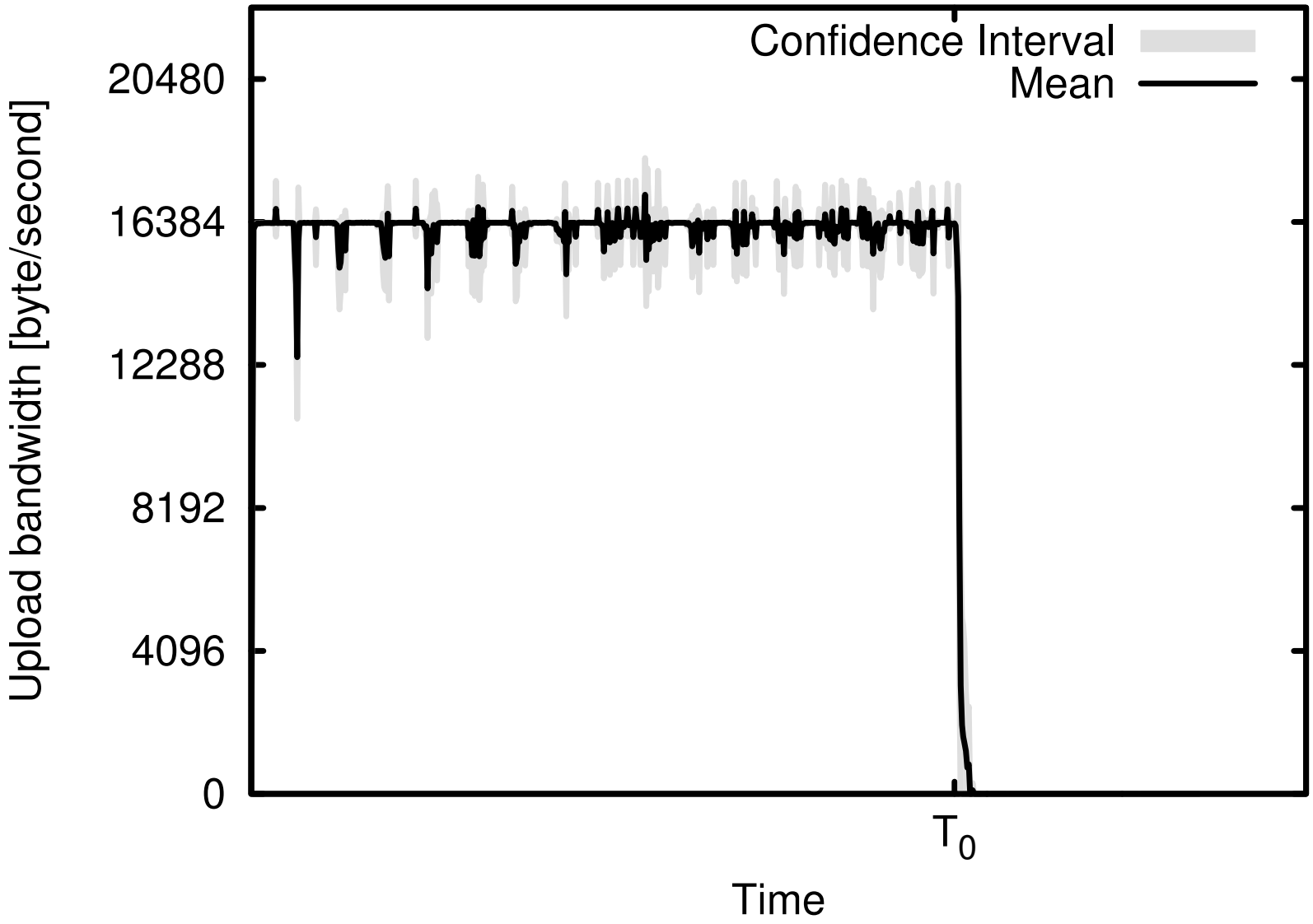
Anhang - Default Szenario mit 16x Chunkanzahl

Completion Graph:



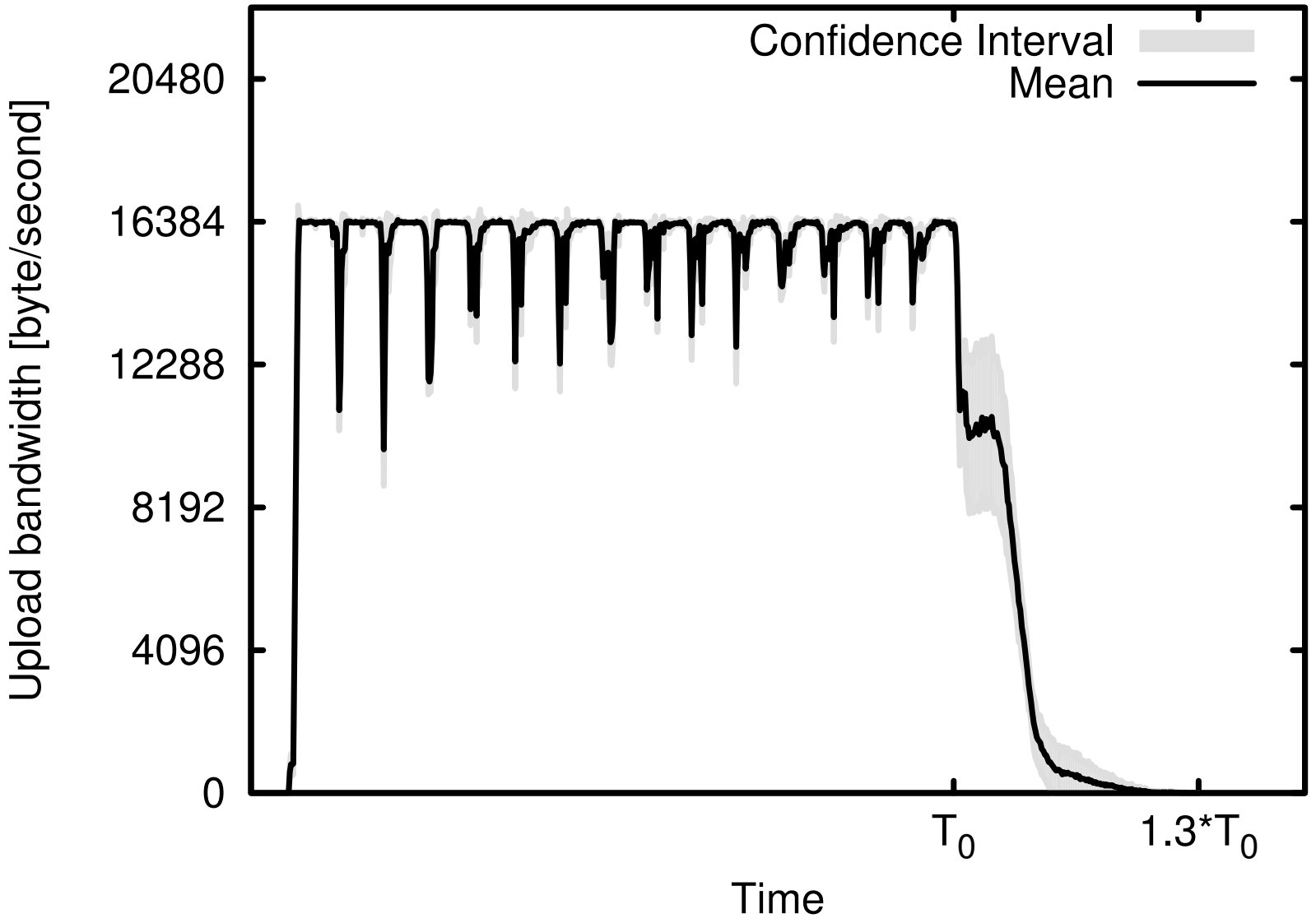
Anhang - Default Szenario mit 16x Chunkanzahl

Super-Peer Upload Bandwidth:



Anhang - Default Szenario mit 16x Chunkanzahl

Peer Upload Bandwidth:



Anhang - Default Szenario mit 16x Chunkanzahl

Peer Download Bandwidth: